UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE AUTOMÁTICA

"A PhD Dissertation on Road Topology Classification for Autonomous Driving"

Author

Álvaro Hernández Saz

Supervisors

Dr. D. Miguel Angel Sotelo Vázquez
Dr. D. Ignacio Parra Alonso

2021

Doctoral Thesis

Muad'Dib podía realmente ver el futuro, pero hay que comprender que su poder era limitado. Pensad en la vista. Uno tiene los ojos, pero no puede ver sin luz. Si uno está en el fondo de un valle, no puede ver más allá de ese valle. Igualmente, Muad'Dib no podía mirar siempre en el misterioso terreno del futuro. Nos dice que cualquier oscura decisión profética, tal vez la elección de una palabra en vez de otra, puede cambiar totalmente el aspecto del futuro. Nos dice: "La visión del tiempo se convierte en una puerta muy estrecha." Y él siempre huía de la tentación de escoger un camino claro y seguro, advirtiendo: "Este sendero conduce inevitablemente al estancamiento." - De *el despertar de Arrakis* de la Princesa Irulán
(Frank Herbert, Dune)

Agradecimientos

Muchas gracias a Miguel Ángel y a Nacho por ser unos estupendos directores de tesis. Han sabido guiarme en este mundo de la investigación y del Deep Learning en el que entré siendo un completo novato y del que salgo, de momento, siendo un poco menos novato. Muchas gracias a mis padres por darme insistentemente la turra durante el ultimo año con el estado de mi tesis. Muchas gracias a Adriana por no dármela y soportar muchas cosas sola mientras yo estaba escribiendo delante del ordenador. También quiero dar las gracias a Augusto, por ser un trabajador incansable y un gran compañero. Mucha de la investigación de esta tesis doctoral no habría podido salir adelante sin su colaboración. Muchas gracias a Noelia por su ayuda con las figuras y la maquetación, dada mis cualidades artísticas no habría sacado ni una sin sus consejos. Muchas gracias a Scrat, siempre es de reconocer encontrar a alguien con quien hablar y resolver dudas mutuamente con toda la confianza del mundo. Por último, muchas gracias al núcleo duro del laboratorio, MalHector, Cuñao, AQP, y LSD. Sin ellos esto tampoco habría sido posible, porque tan importante es el trabajo duro como el tiempo de esparcimiento sacándonos los ojos con las nerfs.

Resumen

La clasificación de la topología de la carretera es un punto clave si queremos desarrollar sistemas de conducción autónoma completos y seguros. Es lógico pensar que la comprensión de forma exhaustiva del entorno que rodea al vehiculo, tal como sucede cuando es un ser humano el que toma las decisiones al volante, es una condición indispensable si se quiere avanzar en la consecución de vehículos autónomos de nivel 4 o 5. Si el conductor, ya sea un sistema autónomo, como un ser humano, no tiene acceso a la información del entorno la disminución de la seguridad es crítica y el accidente es casi instantáneo i.e., cuando un conductor se duerme al volante.

A lo largo de esta tesis doctoral se presentan sendos sistemas basados en deep leaning que ayudan al sistema de conducción autónoma a comprender el entorno en el que se encuentra en ese instante. El primero de ellos 3D-Deep y su optimización 3D-Deepest, es una nueva arquitectura de red para la segmentación semántica de carretera en el que se integran fuentes de datos de diferente tipología. La segmentación de carretera es clave en un vehículo autónomo, ya que es el medio por el que debería circular en el 99,9% de los casos. El segundo es un sistema de clasificación de intersecciones urbanas mediante diferentes enfoques comprendidos dentro del metric-learning, la integración temporal y la generación de imágenes sintéticas. La seguridad es un punto clave en cualquier sistema autónomo, y si es de conducción aún más. Las intersecciones son uno de los lugares dentro de las ciudades donde la seguridad es crítica. Los coches siguen trayectorias secantes y por tanto pueden colisionar, la mayoría de ellas son usadas por los peatones para atravesar la vía independientemente de si existen pasos de cebra o no, lo que incrementa de forma alarmante los riesgos de atropello y colisión.

La implementación de la combinación de ambos sistemas mejora substancialmente la comprensión del entorno, y puede considerarse que incrementa la seguridad, allanando el camino en la investigación hacia un vehículo completamente autónomo.

Palabras clave: Aprendizaje profundo, Redes Neuronales Artificiales, Conducción Autónoma, Segmentación Semántica, Clasificación.

Abstract

Road topology classification is a crucial point if we want to develop complete and safe autonomous driving systems. It is logical to think that a thorough understanding of the environment surrounding the ego-vehicle, as it happens when a human being is a decision-maker at the wheel, is an indispensable condition if we want to advance in the achievement of level 4 or 5 autonomous vehicles. If the driver, either an autonomous system or a human being, does not have access to the information of the environment, the decrease in safety is critical, and the accident is almost instantaneous, i.e., when a driver falls asleep at the wheel.

Throughout this doctoral thesis, we present two deep learning systems that will help an autonomous driving system understand the environment in which it is at that instant. The first one, 3D-Deep and its optimization 3D-Deepest, is a new network architecture for semantic road segmentation in which data sources of different types are integrated. Road segmentation is vital in an autonomous vehicle since it is the medium on which it should drive in 99.9% of the cases. The second is an urban intersection classification system using different approaches comprised of metric-learning, temporal integration, and synthetic image generation. Safety is a crucial point in any autonomous system, and if it is a driving system, even more so. Intersections are one of the places within cities where safety is critical. Cars follow secant trajectories and therefore can collide; most of them are used by pedestrians to cross the road regardless of whether there are crosswalks or not, which alarmingly increases the risks of being hit and collision.

The implementation of the combination of both systems substantially improves the understanding of the environment and can be considered to increase safety, paving the way in the research towards a fully autonomous vehicle.

KeyWords: Deep Learning, Neural Networks.

Table of Contents

List of Figures

List of Tables

List of Acronyms

3D-BEV	3D-Generated Bird Eye View.
3DMASKED-BEV	Masked 3D-Generated Bird Eye View.
ADAS	Advanced Driver Assistance Systems.
AI	Artificial Intelligence.
ASSP	Atrous Spatial Pyramid Pooling.
BDSCE	Built-in Depth-Semantic Coupled Encoding.
BEV	Bird's eye view.
CBAM	Convolutional Block Attention Module.
CDC	Correlation Distance Calculation.
CDSC	Compression and Depth-Semantic Coupling.
CNN	Deep Convolutional Neural Network.
CRF	Conditional Random Field.
DGT	Direción General de Tráfico.
EDI	Is an image in which a point cloud is projected and the intensity value of each pixel that is different to 0 represents is elevation..
ELU	Exponential Linear Unit.
FCNN	Fully Convolutional Neural Network.
FL	Focal Loss.
FMS	Feature Map Shifting.
FP	Feature Pyramid network.
GAN	Generative Adversarial Network.
GRU	Gated recurrent unit.
K-NN	k-nearest neighbors.
LiDAR	light detection and ranging.

LM	Landscape Metric.
LSTM	Long Short-Term Memory.
MAP	Mean Average Precision.
MAPR	Mean Average Precision@R.
MBEV	Model-Based Bird Eye View.
MLP	Multilayer Perceptron.
MSE	Mean Squared Error.
NHTSA	National Highway Traffic Safety Administration.
PA1	Precision-at-1.
ReLU	Rectified Linear Unit.
RGB-D	Is an image in which the three first channels are the red green and blue values and the fourth channel is the corresponding disparity map..
RGB-E	Is an image in which the three first channels are the red green and blue values and the fourth channel is the corresponding *EDI* image..
RNN	Recurrent Neural Network.
RP	R-Precision.
SAE	Society of Automotive Engineers.
SVM	Support Vector Machine.
T-SNE	T-Distributed Stochastic Neighbor Embedding.
VRU	Vulnerable Road Users.
WARPING	Warping with Homography.

Chapter 1

Introduction

Artificial Intelligence (AI) has been one of the significant technology disruptions of the twentieth century, and it is still a revolution of the twenty-first century. This researcher believes the multiple uses given to it have always been looking for the benefit or the greater comfort of the human being. Its applicability is practically infinite, and only the future will tell us how far humanity will be able to go thanks to all the theoretical and practical research carried out with increasing assiduity.

Somehow, we all have all imagined AI as Ava in Ex-Machina or Roy Batty (Figure 1.1) in Blade Runner. However, far from science fiction, AI appears in many fields that ordinary people are unaware of. The more its use spreads, the less conscious our minds are of it, ceasing to consider some AI methodologies as such.

If someone asks laypeople about Google's online translator, it is sure that most of them would not consider it AI. Nevertheless, natural language processing is one of the most contemporary research topics studied. It has reached outstanding achievements, such as Google's assistant Duplex presented in 2018 with the famous and controversial two automatic phone calls [Leviathan and Matias, 2018].

Figure 1.1: Roy Batty, a replicant in the film Blade Runner. [Bernard Goldbach, 2008]

If we look to something closer to us, every citizen in developed and developing countries has access to smartphones. These devices are capable of performing tasks that only personal computers were once competent in, and because of that advancement, AI has also been used to improve them. Nowadays, most of them come with facial or fingerprint recognition devices that allow secure access or storing protected content. Countless also comes with built-in assistance into their photographic software/hardware, as Huawei first announced in its P20 model. This assistance uses AI-based methodologies to, for example, obtain better photos in low light conditions or with an unsteady hand.

Along with smartphones, facial recognition software is also used in many other cases, such as for real-time population recognition (Figure 1.2). Countries such as China or the United Kingdom are already applying this methodology to detect suspects, although it is a much more controversial use of facial recognition due to the well-known trade-off between security and privacy.

Figure 1.2: Face recognition in live CCTV.[1]

In conjunction with personal computers, smartphones are also responsible for generating large amounts of data to be analyzed, the so-called Big Data. This amount of data is usually processed by AI techniques with the intention of obtaining helpful information for companies and people. The irruption of artificial intelligence and big data in marketing is perhaps one of the most distinctive approaches that may be taken by the private sector [Forbes, 2019]. An example that the reader has probably experienced is personalized advertisements on the Internet. These ads try to anticipate the user's tastes more or less successfully by studying their digital footprint by distilling data into useful information using the process mentioned above.

Industrial production is also one of the fields in which AI has developed most. In China, for example, there is a company that produces a traditional medicine remedy by breeding cockroaches that are subsequently processed. The production facilities have an AI system that, through the reading of the configured variables, searches for the optimal environment to maximize the reproduction of the insects, thus maximizing the possible benefits. [Chong, 2018]. Another example of its use can be in chain manufacturing, such as using automated systems to validate product specifications and discarding those that do not meet the minimum quality requirements. This automated system usually allows a higher production level than by-hand review due to the increased efficiency. This method is also commonly used in parcel delivery companies or logistics warehouses, such as Amazon. In some of the corporation warehouses, computer vision software monitors product movement within the storage during order generation. The company also uses computer vision in its Go stores, where the software monitors all purchases made by a user by recognizing the products and assigning and charging them directly to the user's account [Bond, 2019].

Moving to a more idle field such as video games, AI also has its place. Gamers are increasingly looking for a more immersive and realistic gaming experience, which AI is trying to achieve. For example, DeepMind, an AI that plays Starcraft II, has reached the rank of Grand Master, the grade where the best players in the world compete against each other [Kelion, 2019]. Playing against it is currently just as challenging as playing against any of the best players in the world. This type of AI development is not limited to real-time strategy games. For example, Electronic Arts have tried to train an agent to

[1] Image obtained in www.biometricupdate.com

play Battlefield I, a first-person shooter game, who can "impersonate" a real player in a real game without being noticed by the rest of the players [Nordin, 2018].

AI plays a crucial role in autonomous driving, a field closer to this thesis's scope. Also, one of the areas is currently gaining the most notoriety both inside and outside AI research circles. This research domain seeks to develop a self-driving car that might be defined as a ground vehicle capable of sensing its environment and moving safely with little or no human intervention. The ultimate goal of the research is to achieve an *Society of Automotive Engineers (SAE)* level 5 or fully autonomous driving vehicle [International, 2014]. This automation level would make it possible to somehow convert vehicles into passenger transport capsules in which total responsibility for driving lies in the car, with the multiple benefits that this would bring. Increased transport efficiency and vehicle safety are two excellent examples of these benefits. However, to reach this level 5 of automation, there is still much research to be done.

1.1 Motivation

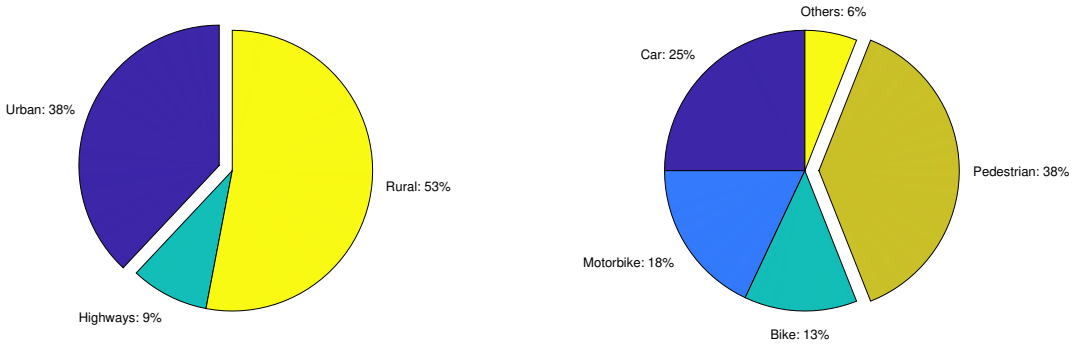
As mentioned before, despite being one of the hottest branches of artificial intelligence, autonomous driving still requires a great deal of research, and part of it is what we are trying to develop in this doctoral thesis. Among the multiple approaches that can be taken in autonomous vehicles, in this case, research will focus on safety and its implementation through scene understanding in urban environments. Even though the performance and availability of scene understanding systems increased over the past years, technology seems to be far from the SAE full automation level requirements. This statement is particularly true regarding urban areas and contexts without a strict Manhattan-style city plan, i.e., oversize building blocks surrounded by simple straight avenues. The transition towards full automation requires reliability under all circumstances, including ordinary middle and small cities where narrow and strangled roads are the most common.

During the year 2019, an estimation of 22.700 people died throughout Europe due to traffic accidents, according to the European Commission's latest report [Commision, 2020]. Furthermore, according to the last Annual Accident Report of 2018 [Commission, 2018], 19.4% of EU road fatalities come from *at-grade* intersection areas. Similar results emerge from the reports of the United States *National Highway Traffic Safety Administration (NHTSA)*, which show us that in the 2015-2019 period, *at-grade* intersections areas concentrate more than 40% of motor vehicle accidents and 25% of fatal crashes [Administration, 2019].

If the European data are broken down, as shown in Figure 1.3a, it can be seen that 38% of the deaths occurred in urban areas. Most of them were pedestrians within these fatalities, accounting for 38% of the total (Figure 1.3b).

Going towards a more local level, such as Spain, the number of deaths in 2019, the last published official report on its website, is 1755 [DGT, 2019]. As in the European statistics, these can be broken down according to where the accident occurred, as shown in Table 1.1. If we compare both statistics, it can be seen that the percentages are pretty similar: 30% urban environment, 51% rural environment, and 19% on highways.

If we want to break down more precisely the urban figures in Spain, the September



(a) Percentage of road fatalities by road network.

(b) Percentage of road fatalities by means of transport.

Figure 1.3: Percentages about road fatalities in urban roads [Commision, 2020].

Localization	Number	Percentaje
Interurban	1236	70%
Highway	91	5%
Motorway	249	14%
Rural road	896	51%
Urban	519	30%
Crossing	43	2%
Street	473	27%
Urban highway	3	0%

Table 1.1: Number and percentage of road fatalities in Spain.

2018 *Direción General de Tráfico (DGT)* report on the figures in 2017 [DGT, 2017] should be referred to. If the report is analyzed, as shown in Figure 1.4, 96% of the accidents and 90% of the fatalities took place on urban streets. These extremely high percentages put the spotlight on urban lanes as the place where safety must be improved. In Figure 1.4, we can learn a little more about the casuistry of accidents. In it, we can see that 20% of the accidents are pedestrian run overs, another 20% are rear-ended or multiple collisions and 35% are lateral or frontal-lateral collisions. However, there is a striking change in the data percentages within the fatality graph. 47% of the fatalities are due to pedestrian run overs, 16% to lateral or frontal-lateral collisions, and another 16% to run-off-the-road collisions, which surprisingly only account for 4% of the total number of accidents. Some quick deductions can be made after the detailed analysis conducted in the figures.

- The vast majority of accidents and fatalities in urban environments occur on streets, and they are pedestrian collisions and lateral or front-lateral collisions.
- Run-off-the-road accidents, even though being one of the least frequent, have a high fatality rate.

These particular circumstances highlight the lines on which a scene recognition and safety system for autonomous vehicles should focus for the time being. Therefore, this thesis will address the objectives stated in the following point to contribute to improving the performance of the existing systems under the above assumptions.

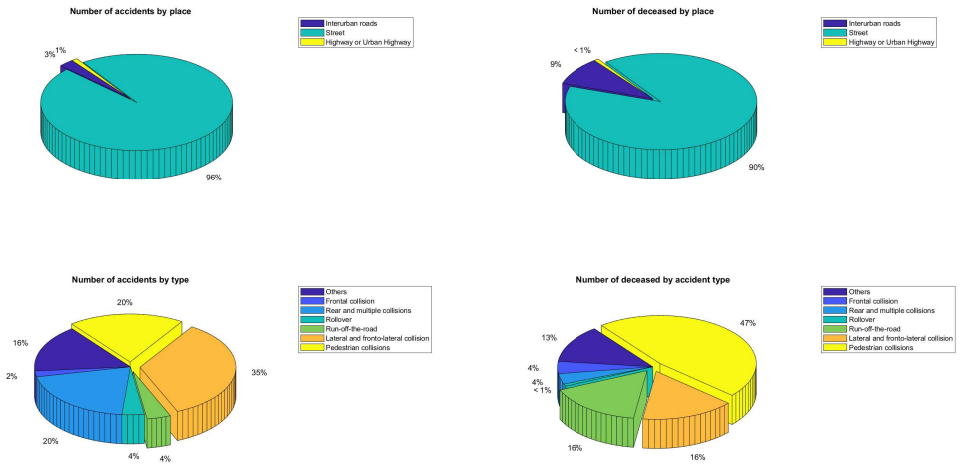


Figure 1.4: Comparison of accidents and fatalities by type of road and type of accident.

1.2 Scope of this thesis

After analyzing the causes and particularities of accidents and fatalities in the urban environment, the next step is to establish the objectives. This thesis aims to implement, employing deep learning, optimization, and data pre-processing techniques, surrounding identification systems that allow an autonomous car to navigate in an urban environment safely.

One of the most striking figures presented in point 1.1 is the fatality rate of runoff-the-road accidents. The number of fatalities during 2017 due to this case study is 80, practically the same as in lateral and frontal-lateral collisions, being the latter practically ten times more frequent.

Therefore, we can deduce that it is vital for an autonomous vehicle to recognize the road on which it must and can circulate with total accuracy. This problem can be addressed through a deep learning technique called semantic segmentation or scene interpretation (Figure 1.5). *Semantic segmentation* is a supervised learning technique that classifies each of the pixels of an input image according to the required information, so it can be used to recognize each road's pixels, thus obtaining the road surface.

On the other hand, the other two types of accidents with the highest fatalities are pedestrian run overs and lateral and front-lateral collisions. That brings us to another of the troublesome points within the cities, the intersections. It seems clear that most

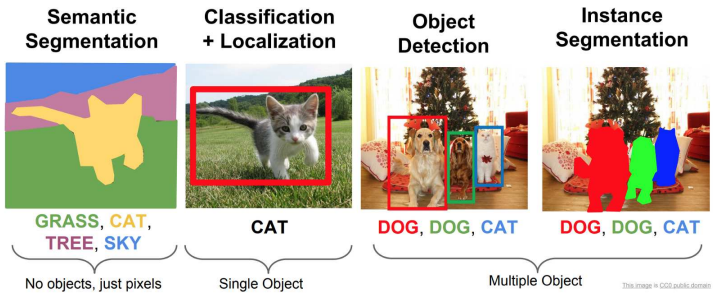


Figure 1.5: Computer vision tasks.

lateral and front-lateral collisions might occur at intersections between two roads, where the vehicle's paths will cross at some point. It is also at intersections where crosswalks are most often encountered and where it is, therefore, more likely that a vehicle could accidentally collide with a pedestrian. For example, a pedestrian on the sidewalk must cross the street in two of the three possible paths at a cross intersection (Figure 1.6).

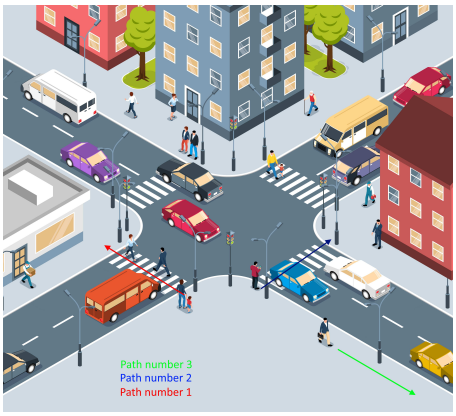


Figure 1.6: Possible trajectories within an intersection.[2]

Consequently, within the navigation of an autonomous vehicle, it can be considered very important to have a system that recognizes the scene and identifies the type of intersection. If the vehicle recognizes the upcoming intersection, it will deduce helpful information. An example of this valuable information could be the number of entrances to the intersection, thus knowing the topology of the intersection. Consequently, it will know from which trajectories another vehicle or pedestrian on a collision course with the ego-vehicle may come from and imply higher risk. In addition, and outside the field of safety, by knowing the type of junction where the vehicle is located, it will be able to perform navigation maneuvers without the need for a complete and detailed map of the area, as a human being would do when asking for directions in an unfamiliar city.

Deep learning methodologies can also solve this casuistry, but in this case, with a set of different types of techniques. In this thesis, we propose that the classification task (Figure 1.5) is necessary to solve the problem adequately. This technique tries to obtain as relevant features as possible from the input data to return the input data category within the ones we are interested in.

Figure 1.7 shows the workflow of this thesis. It can be seen how these problems have common points that can be integrated into a single system whose usefulness would undoubtedly allow it to be integrated into even more complete future developments.

[2]Image obtained in freepik.com

Figure 1.7: Thesis workflow diagram.

1.3 Document structure

After the introduction in Chapter 1, Chapter 2 contains a brief review of the most significant research on this work's three main points:

1. Review work on semantic segmentation and road segmentation will be conducted.
2. Classification and intersection classification researching will be reviewed.
3. A discussion on papers on the optimization of deep learning architectures will be developed.

Chapter 3 will discuss the different methodologies that have been used for road segmentation during the research process of this doctoral thesis:

1. Starting with basic implementations in CAFFE and add complexity until the data integration led to the new 3D-Deep architecture.
2. Addressing 3D-Deep architecture's optimization to single-task targeting led to the update of the same in 3D-Deepest.

In Chapter 4 we will discuss the different training strategies implemented to solve the intersection classification.

1. A teacher/Student approach with single-frame data will be addressed.
2. Switch the previous approach to a metric learning approach with triplet loss.
3. Finally, the input information is extended by temporal integration of the data.

Chapter 5 contains the conclusions and main contributions in road segmentation intersection classification, network optimization, and future research lines.

Finally, Appendix A describes all the other things vaguely related to THE TOPIC, and Appendix B summarizes the leading publications derived from this Ph.D. dissertation.

Chapter 2

State of the Art

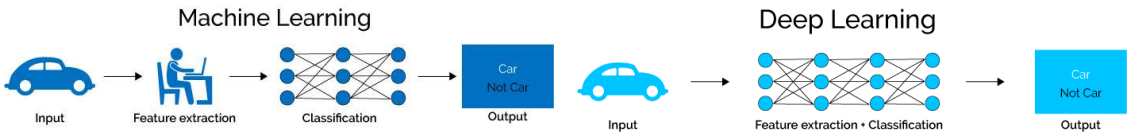
As seen in the previous chapter, the way to face and solve the problems proposed in this doctoral thesis will be through semantic segmentation and classification techniques. These procedures are usually based on *Deep Convolutional Neural Networks (CNNs)*, one of the most common methodologies within deep learning and computer vision.

This chapter will conduct an exhaustive review of the research topics that have brought these deep learning methodologies to lead within computer vision. Remarkably, the search will be focused on those currently in the state-of-the-art.

The relevance of computer vision inside the field of artificial intelligence rocketed in the last few years. The amount of research found in almost any specialized search engine is the most substantial evidence. This significant increase is mainly due to the deep learning revolution led by the success of previously named CNNs (Figure 2.1b).

However, this is not to say that there was no active research before and after the advent of CNNs. Usually, proposed none deep learning methods rely on hand-built features, machine learning techniques, geometrical transforms, or any combination of them (Figure 2.1a). Random Decision Forest [Scharwächter and Franke, 2015] or Boosting [Sturgess et al., 2009] are two examples of it.

For example, in [Oniga et al., 2008] and [Siegemund et al., 2010], they use digital elevation maps along Hough transformation or *Conditional Random Field (CRF)* to detect and reconstruct curbs. Going further back in time, [Felzenszwalb and Huttenlocher, 2004] uses a graph-based representation to segment the regions of an image, and [Otsu, 1979] employs an optimal threshold as the discriminant for segmentation from the grey-scale histograms.



(a) Traditional machine learning approach.

(b) Novel deep learning approach

Figure 2.1: Learning approaches in machine learning.

2.1 Deep learning for semantic segmentation

Moving away from the field of traditional computer vision methods and into deep learning, CNNs, as previously stated, boosted the research in computer vision. Notably, since the unmatched effectiveness in 2012 in the ImageNet classification challenge [Russakovsky et al., 2015].

Networks like Alexnet [Krizhevsky et al., 2012], VGG [Simonyan and Zisserman, 2014], and GooleNet [Szegedy et al., 2015] and their mentioned effectiveness are the ones to blame for this significant increase in machine vision research in general and in CNNs in particular. These architectures focus on feature extraction from images, using sequential convolutional layers for subsequent classification. The training images are obtained from massive datasets specially generated for the task, in which usually efficiency rankings are established, e.g., [Everingham et al., 2010], [Russakovsky et al., 2015] and [Geiger et al., 2012]. This end-to-end automated supervised learning methodology applied to massive amounts of data usually outperforms hand-crafted features in most of the tasks.

Introduced in 2015, *Fully Convolutional Neural Networks (FCNNs)* [Long et al., 2015] proposes a method based on an encoder-decoder pipeline (Figure 2.2) to segment pixel-wise an image in each of its classes. The encoder part of the network is in charge, as in traditional CNNs, of obtaining context and spatial information. This operation is performed by means of gradually down-sampling the images and synthesizing them in a set of meaningful features. After that codification, the encoder's classification layers are discarded, and the set of features is delivered to the decoder part of the network. Subsequently, the decoder applies up-sampling operations on the supplied features through a series of sequential deconvolutional layers [Zeiler et al., 2010] until a probability map of the same input image size is obtained for each of the classes. The parameters of the deconvolutional layers can be learned during the training process, as in convolutional layers. Alternatively, they can be fixed, for example, in a bilinear interpolation, depending on what the researchers consider most appropriate for the system's best performance.

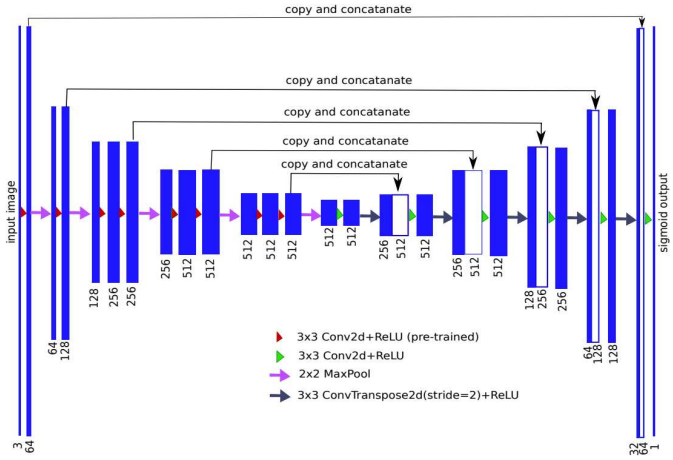


Figure 2.2: Encoder-decoder architecture [Iglovikov and Shvets, 2018].

FCNNs research has significantly advanced since 2015, introducing new improvements in both efficiency and effectiveness. Simultaneously, the significant increase in the number of datasets [Cordts et al., 2016, Wang et al., 2019b, Xie et al., 2016, Mnih, 2013] provided for semantic segmentation tasks noteworthy facilitates the validation and testing of new methodologies. Some examples of this assertion are listed below.

[Ronneberger et al., 2015], whose work implements a new architecture with a contracting path and a symmetric expanding path to facilitate precise localization in biomedical image segmentation. [He et al., 2016] proposes ResNet, an architecture with shortcut connections between layers to create a residual learning framework to learn residual functions referenced to layer inputs in order to avoid the vanishing gradient problem. This approach facilitates the optimization of deeper networks, generally more accurate than those previously implemented, which could be used in segmentation and semantic classification tasks. [Badrinarayanan et al., 2017] introduces SegNet, an encoder-decoder architecture in which the encoder part is a VGG network, and the decoder part uses the pooling indices computed in the max-pooling step of the corresponding encoder to perform non-linear up-sampling of the features. [Chen et al., 2017] proposes the dilated/atrous convolution (Figure 2.3) to enlarge the field-of-view without no resolution loss of the feature descriptors. [Chen et al., 2014a] uses CRFs as a post-processing feature classifier to improve location capability to increase the precision in the segmentation of the boundaries of the object. ParseNet, architecture defined in [Liu et al., 2015], uses global average pooling to entirely use global and local image features. In [Zhao et al., 2017], the aggregation capacity is exploited, through the pyramid-pooling module, of the different region-based context information to obtain global context information.

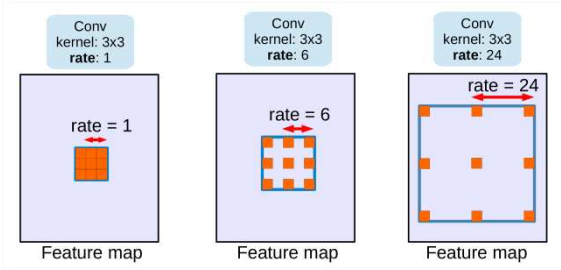


Figure 2.3: Dilated/Atrous convolution kernels example [Chen et al., 2017].

Newly, some research that utilizes depth and *light detection and ranging (LiDAR)* information for scene understanding has arisen. These methods usually extend the RGB images with a fourth channel that contains the three-dimensional information transforming them into RGB-D images. This new data source can be just concatenated to the RGB channels, like in [Couprie et al., 2013], or pre-processed and encoded to extract richer features as can be with HHA images [Gupta et al., 2014].

Moving to 2020, in [Liu et al., 2020b], a new module and some new network architectures that benefit from it are proposed to solve the scene understanding task. The proposed module, *Built-in Depth-Semantic Coupled Encoding (BDSCE)* (Figure 2.4), adaptively extracts and fuses RGB and depth information inspired by binocular stereo vision methodology. The module is divided into three sequential parts that act on two input

feature maps: *Feature Map Shifting (FMS)*, *Correlation Distance Calculation (CDC)*, *Compression and Depth-Semantic Coupling (CDSC)*. The first one, FMS, shifts the right feature maps to match them with their corresponding left feature maps. The CDC module then calculates the difference via the $\mathcal{L}_1$-norm between the left and already shifted right features. The third one, the CDSC module, compresses the third channel's results to acquire the positional deviation of the corresponding points and obtain the geometric information. Then *Depth-Semantics Coupling* sub-module matches the complementary features between the RGB input and the new depth information extracted.

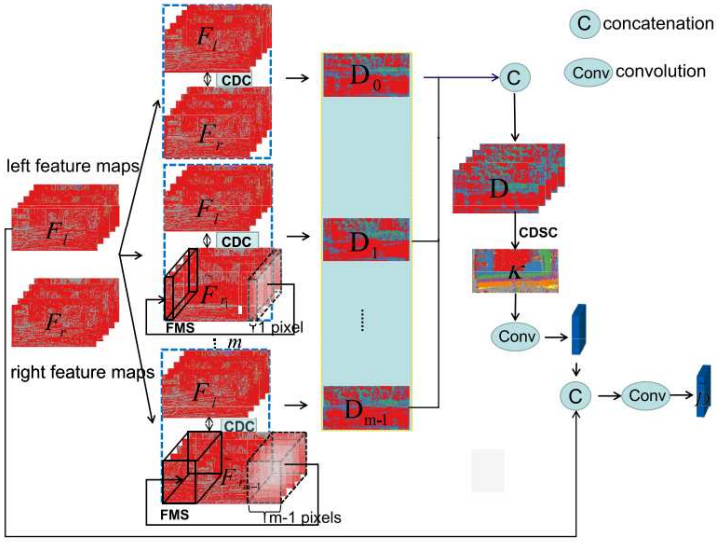


Figure 2.4: BDSCE workflow example [Liu et al., 2020b].

The researchers suggest four network models, *CEncNet-FCN, CEncNet-Deeplab and CEncNet-PSP, CEncNet-Multinet*, *CEncNet-FCN*, based on the proposed module introduced to validate its improvement in each assigned task. In the CEncNet-FCN, a BDSCE module is positioned at specific points to calculate the corresponding features' depth information and integrate it with the information coming from lower layers of the network to provide a correct scene understanding. *CEncNet-Deeplab and CEncNet-PSP* are two minor actualizations on the Deeplab and PSP architectures that introduce the BDSCE module on top of the conv5 layer. Its results are fed to the *Atrous Spatial Pyramid Pooling (ASSP)* module to obtain the final prediction. *CEncNet-Multinet* is an architecture actualization of Multinet, a network designed to perform parallel work, road segmentation, and vehicle detection. The BDSCE module is placed in the network's encoder part, between all the three convolutional layers, and its results are fed to the two decoder branches, which remain unchanged. Each decoder branch is in charge of each different task: road segmentation and vehicle detection.

Furthermore, during 2019 was published [Deng et al., 2019] proposing a new convolution type to deal with training images with significant distortion, *Restricted Deformable Convolution* (Figure 2.5), a version of the deformable convolution introduced in [Dai

et al., 2017]. This particular convolution is based on correcting the lack of spatial information that the deepest layers of CNNs suffer from. This augmentation of spatial information is led by a set of offsets applied to the sampling locations. These offsets' values are learned simultaneously as the convolutional kernel weights during the training process, which does not substantially increase the difficulty of the training. Knowing this, the idea behind *Restricted Deformable Convolutions* is fixing the kernel's central location since the authors believe that model transformations strongly rely on the outer sampling locations.

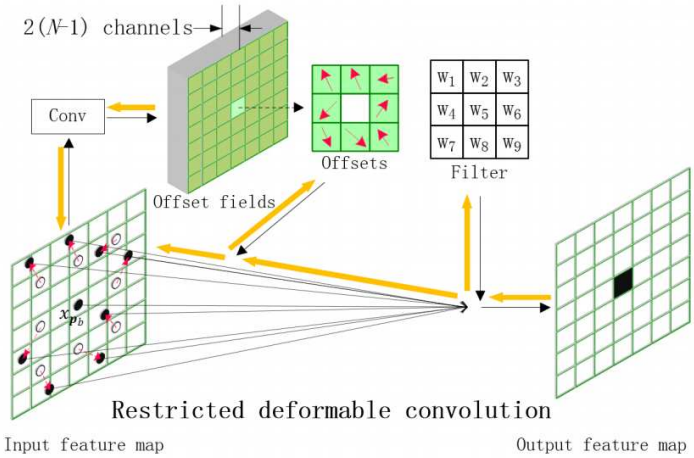


Figure 2.5: Restricted deformable convolution example [Deng et al., 2019].

A regular convolutional layer with fixed channels is used since the offsets' values should be learned together with the weights' values. This convolutional layer establishes a bidirectional relationship between its weights and the offset values that will then be applied to the original kernels.

Also, for training purposes, the work uses a data augmentation method called Zoom Augmentation that transforms the input images from panoramic view to fish-eye view. This transformation is made to supply the need for a large-scale dataset of context images, usually provided from surround camera systems that typically produce fish-eye images.

It is worth commenting that other research areas exist inside the semantic segmentation field, as can be the instance-aware semantic segmentation. Instance segmentation focuses on target objects within the provided data and labels only those objects and not all the image pixels. One of the most remarkable examples of this approach is proposed in [He et al., 2017].

2.1.1 Deep learning for road semantic segmentation

So far, the growing importance of scene understanding has been seen within the field of deep learning and, in particular, in autonomous vehicles. However, understanding the whole scene is not always necessary to have enough information for proper driving safety. As will be seen, road segmentation is a particular field within semantic segmentation

that is very useful and in which the amount of research has been increasing exponentially over time. Perhaps, when one thinks of road semantic segmentation, one has in mind a panoramic image in which all the road pixels are correctly labeled. However, as will be shown below, in recent years, other approaches may also substantially influence the field of autonomous driving.

In [Henry et al., 2018], the input data is not an RGB image obtained from a car's dashboard; instead, it is synthetic-aperture radar satellite images. The research focuses on extracting roads from this particular type of image since this information has a great value in keeping maps up-to-date and automatic map building. Three FCNNs were proposed to be trained end-to-end to solve the task: FCN-8s, U-Net, and DeepLabv3+. Also, because of the class imbalance in the training data, a spatial tolerance parameter is set to transform the task from a binary classification to a binary regression to avoid spatially small mistakes. Finally, pre-and post-processing of the data, such as non-local filtering and fully connected CRFs, are studied to improve the results. These last steps did not work well in this precise task, validating something prevalent in deep learning: the improvements that worked in specific research will not always work in similar other ones.

If we continue with the remotely-sensed images, [Panboonyuen et al., 2017] proposes a methodology following similar ideas to the previous research: road extraction in very-high-resolution images or satellite images, with a specific conformed network made for the task. The network's adaptations can be labeled into two categories, post-processing, and processing.

Starting with the processing upgrade, this consists of modifying the base architecture, in this case, SegNet [Badrinarayanan et al., 2017], to introduce a new activation function, the *Exponential Linear Unit (ELU)* [Clevert et al., 2015] (Figure 2.6), as opposed to the *Rectified Linear Unit (ReLU)* based on its performance. In addition, the number of parameters is optimized by discarding the three fully connected layers between the encoder and decoder.

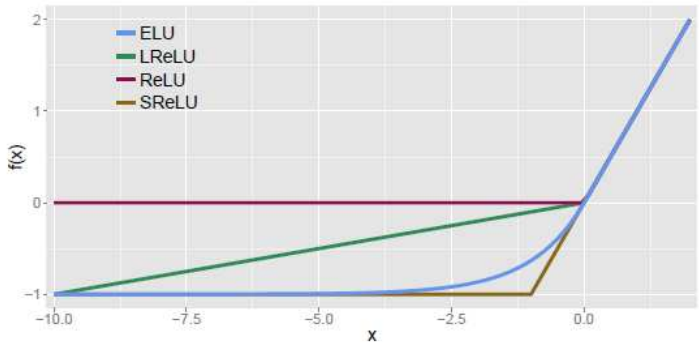


Figure 2.6: ELU activation function comparison [Clevert et al., 2015].

Following with post-processing upgrades, CRFs must be mentioned again, in this case, together with *Landscape Metrics (LMs)*. This last method is focused on false object

removal based on the shape complexity of previously detected objects, followed by a filtering process whereby any object with a complexity index lower than 1.25 is discarded as a candidate. In order to increase the correctness of the LMs method, two previous steps are done. The first is a *Gaussian smoothing* to remove unnecessary details. The second is a *connected component labeling* to combine and group all the pixels of object candidates. After all, this initial post-processing CRF comes into play. A fully connected CRF is used to sharpen segmentation results by adding explicit dependencies among the neural network outputs.

A significantly different approach is taken in [Shamsolmoali et al., 2020], where a *Feature Pyramid network (FP)* combined with domain adaptation with adversarial networks is used to segment the road in remotely-sensed images.The adversarial training aims is to minimize the gap between the sample images and the target domain by generating realistic synthetic images based on authentic images.

Generative Adversarial Networks (GANs) [Goodfellow et al., 2014] generally are composed of two modules: the generator in charge of creating samples of a domain from random noise and a discriminator that tries to recognize the fake samples from the real ones (Figure 2.7). These two modules compete with each other to obtain equilibrium between is losses. In this work, the generator aims to produce artificial feature maps similar to the real ones that can minimize the domain gap between them. The generator is a specially designed multiconnected FP to extract multiscale pyramid features from noise and specific features of processed synthetic images. That module is followed by fusion and concatenation to produce additional descriptive features, which are in turn merged to finally obtain different scale feature maps that are then collected and processed to assemble a prediction, in this case, an artificial feature map. The discriminator, in this case, is a *Multilayer Perceptron (MLP)* that uses as inputs the feature maps vectorized.

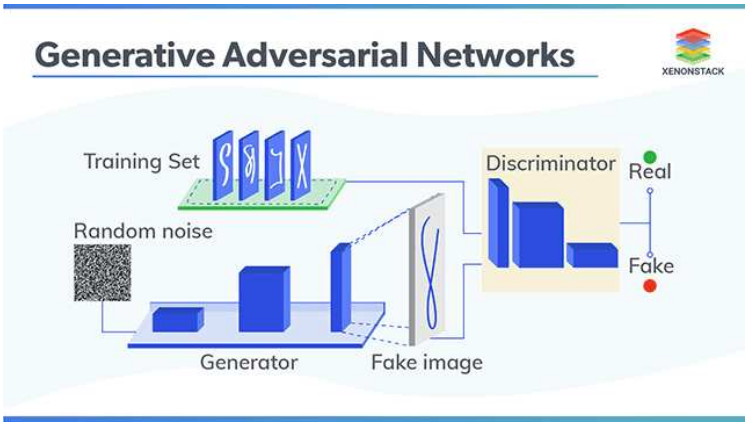


Figure 2.7: GAN workflow [xenonstack, 2019].

Moving on to other approaches, transfer learning is a training technique widely used in deep learning research and, therefore, road segmentation. [He et al., 2019] uses this technique to solve generalization problems under specific domain data changes. The system workflow runs as follows. First, a U-net is trained with easy data, such as, in this case, daylight RGB remote-sensing images that are easily accessible. Then the lower

layers of the architecture are frozen, which are those that are in charge of learning deep semantic information, and the higher layers are re-trained, which should extract the basic local features of the new type of data, cross-modal images.

The U-net input is limited to three channels since it is trained with RGB images. However, cross-modal images are not limited by channel number, so they would not be compatible. To overcome this, the authors propose a three-layered autoencoder module that transforms the multi-band input data through convolutions to thirty-two channel data and subsequently to three-channel data. This autoencoder is trained before attaching it to the baseline net.

Focusing on road segmentation in panoramic images research has increased significantly in the last few years, especially with the appearance on the scene of the KITTI specific benchmark [Fritsch et al., 2013]. It is possible to fall into the trap that deep learning has taken over all the research in road segmentation due to its excellent results. However, work continues outside of it, such as in [Fan et al., 2018]. In that research, only three-dimensional information is used to obtain a segmentation map, specifically disparity data. The process follows three basic steps:

1. The estimation of roll angle γ by least-squares fitting to rotate the dense disparity map to its ideal position where each row's disparity values are similar while they change in the vertical axis.

2. The projection of the road pixels from the v-disparity map by dynamic programming and minimum energy search. This part is focused on fitting a road model parabola.

3. The final transformation of the disparity map, where the function previously calculated is applied to each disparity value and added a constant to maintain all values in the positive range.

Then the map is rotated by $-\gamma$ previously calculated, obtaining a disparity map where the road's disparity values become very similar.

If we return to the use of deep learning for the task of road semantic segmentation, it should be said that the research purpose is not always focused on autonomous vehicles. Other devices, such as mobile robots, need to know which areas of their environment will be capable of circulating. In turn, and as seen above, the conditions under which the input data are obtained can affect the results. Different luminosity or rain are conditions that can negatively affect the quality of the data. That assertion is discussed in [Zhang et al., 2018], where the main idea is to construct a deep learning architecture to segment roads in all-day outdoor data.

The network is divided into two phases. The former one is the forward model, composed of a generative network and a segmentation network. The latter one is a mirror of the former one (Figure 2.8).

The main idea behind this architecture is that the forward model is the one that is in charge of learning the proposed problem and the backward model supervises the learning process of the forward one. The forward phase of the generative model tries to map from the input data, which are poor quality images, to an image domain with richer semantic information. A GAN makes that process in which, as mentioned above, the generative

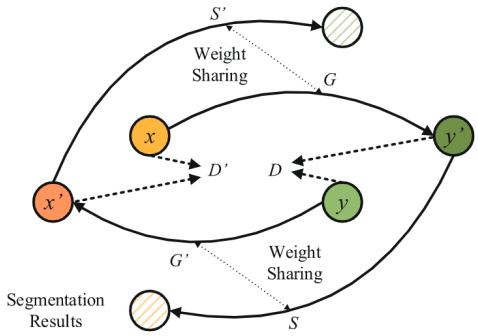


Figure 2.8: Workflow of segmentation architecture [Zhang et al., 2018].

model competes with a discriminator. Then the segmentation model, an FCNN, returns the segmentation result. As said above, the backward model is only active during the training process and supervises the forward model since it has difficulties segmenting road boundaries correctly. The generative model of the backward phase also makes a mapping, but in this case, it is in the opposite direction, from a rich semantic information image to a poor quality image. This generative model shares the weights with the segmentation module of the forward model. The segmentation module of the backward shares the weights with the generative module of the forward and segment the road in poor quality images. This weight-sharing methodology supervises the whole training focusing on maintaining the road borders information to secure it for the subsequent testing phase, where only the forward model is used.

Generally, the road semantic segmentation task focuses on labeling each visible pixel at the input image. That approach can be advantageous in ideal conditions or clean environments like highways. Anyhow, these ideal conditions are not often fulfilled since heavy traffic is expected in an urban environment. To the extent of the researcher's knowledge, cars around the vehicle-ego lead to poor results due to the lack of information about the road situation.

[Wang et al., 2019a] addresses this problem by inferring the occluded road by an extensive insight into the location's geometry and semantic. Researches proposed a new architecture based on the previously mentioned encoder-decoder method as the baseline and a loss function with spatially-dependant weights to pay attention to road edges. Unlike other works, the input data are not raw data, such as RGB images or point clouds, but instead previously obtained semantic representations of the image.

The encoder task comprises several block types, the context downsampling block, the factorized block, and the dilated block. The context down-sampling block combines a regular convolution block and an attention branch that bypasses the main branch to prevent the vanishing gradient and add the attention capability. The factorized block is applied to extract dense features and the dilation block, as has already been reported, enlarges the field of view of the encoder. The decoding task comprises two modules: the joint context up-sampling block and the residual bottleneck block. The first one merges two feature maps of different encoder stages, therefore of different sizes. That operation

is made in order to group the spatial information, which comes from the higher resolution map, and the context information from the lower ones, and is performed employing convolutional blocks and bilinear up-sampling. Therefore, this module is in charge of the up-sampling operations inside the decoder. The bottleneck residual blocks are located between the encoder and the decoder and deal with the vanishing gradient problem. In the early stages of the encoder, the factored residual blocks are used to extract dense features to send them directly to the decoder phase. In the final stages of the encoder, dilated convolutions are used to increase the receptive field, and the extracted features are also sent directly to the decoder. A continuous regular residual block is inserted inside the decoder branch itself, every joint context up-sampling block.

The other innovation proposed in the last work is the loss function. Based on the authors' hypothesis that the pixels closest to the road edge are the most difficult to detect, they propose modifying the cross-entropy function by adding distance-dependent weights to a set of pixels located around the road boundaries. The distance used for the calculation of these weights is the Manhattan distance.

Other works, such as [Liu et al., 2020a], focus on integrating different data sources to improve the segmentation performance. The network architecture follows the already known encoder-decoder scheme, but unlike other works, it is proposed to merge the RGB data and the LiDAR data during the decoder stage. The Refine Fusion Unit performs this data fusion (Figure 2.9).

This unit aims to refine the score maps obtained at the encoding stage with the LiDAR maps generated at various scales. This strategy makes it possible to avoid the drawbacks of both early and late fusion strategies and not require considerable structural modifications to prevent the use of pre-entrained networks. Its construction consists of two phases. In the first one, the LiDAR information is fused with the feature maps of the same stage and the previous stage. Convolution operations make the data fusion, while upsampling operations are made for resolution adaptation and addition from different stage maps. A chained residual pooling is applied in the second one, similar to the one used in RefineNet [Lin et al., 2017a], which tries to capture as much important information as possible from the combined LiDAR feature data.

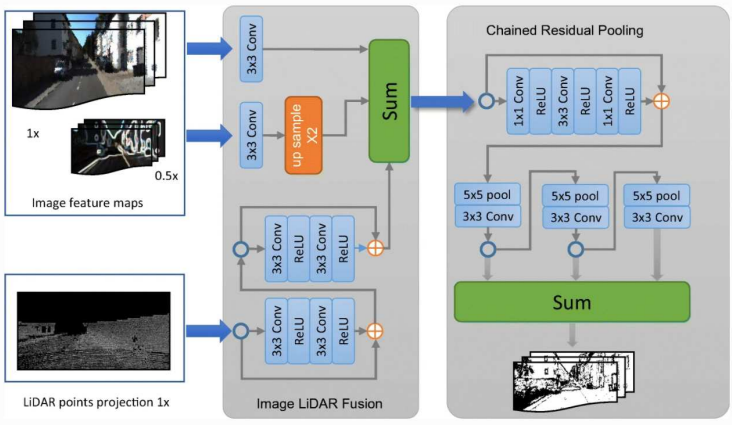


Figure 2.9: RFU Unit architecture [Liu et al., 2020a].

The LiDAR data is obtained by projecting the 3D points in the RGB image plane. Since the authors' idea is to use the complete projected data in all the stages of the deconvolution process, they used a multiscale reprojection by introducing a scale factor in the camera's intrinsic parameters to obtain a set of pseudo-parameters that will do the scale reprojection.

In 2020, [Yan et al., 2020] presented a work in which they disregard RGB images and work directly with LiDAR point clouds to obtain a road segmentation in bird's eye view, eliminating possible occlusions of the road.

Their work's main contribution is the multitasking network architecture, which can segment the road and obtain a crossing typology classification and a dense picture of the road height estimation (Figure 2.10). The backbone network architecture is based on MixNet [Tan and Le, 2019b] and follows the encoder-decoder pattern proposed in many other works. A named Joint Up-sampling Module is added to the backbone to implement a skip pathing between the encoding and decoding stages. This module's function is to up-sample the smaller feature maps with the large resolution ones to concatenate them. A cascade convolution and squeeze and excitation blocks are then used to wisely adapt the feature sources into a unique one that models its interdependencies. Then, this new and better feature map is sent to the corresponding decoder stage.

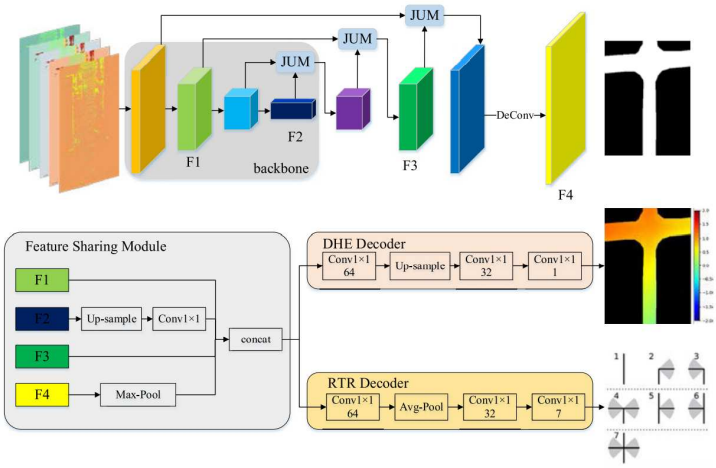


Figure 2.10: Multitasking architecture [Yan et al., 2020].

In order to fulfill the multitask objective, the architecture proposes a feature-sharing module. This module selects significant features of different stages of the main task and merges them to provide convenient features to satisfy the other tasks. The corresponding decoding phase gathers these features for each corresponding task. Each decoder processes features individually to return the height estimate image, or the crossover classification, in the form of a probability vector, as appropriate to each.

As can be seen, this last work shares many of the intentions of this thesis and links to the next section of this chapter, the classification.

2.2 Deep learning in classification

As discussed in chapter 1, intersections are interesting for vehicle navigation. Therefore, they must also be a point of interest in the field of autonomous driving. Recognizing the environment of the autonomous vehicle, as discussed in the previous section, is crucial. However, to recognize intersections, it does not seem necessary to know which pixels belong to which branches of intersections since the road have been previously wholly segmented. Therefore, this problem can be approached differently through classification.

Classification within deep learning is one of the oldest and most researched tasks. Since its beginnings as MLPs as universal approximators [Hornik et al., 1989], it has reached heights of success that had never been achieved before. As discussed earlier in the first paragraphs of Section 2.1 above, the introduction of CNNs was a significant boost to deep learning research and, therefore, in classification tasks. This type of model's effectiveness in image classification ultimately displaced the previous ones, [Russakovsky et al., 2015]. However, MLPs have not been completely discarded. For example, [Mohsen et al., 2018] uses such architecture to classify feature vectors previously extracted from MRI scans to identify and classify three types of brain tumors. Alternatively, in [Qi et al., 2017], a set of MLPs combined with max-pooling and feature transformations is used to label point sets directly and even segment them from raw point clouds.

Nevertheless, returning to CNNs, medicine is one of the fields they have come to be used very effectively. In [Stephen et al., 2019], they are trying to classify lung X-ray images according to whether the patient is affected by pneumonia or not, and in [Tan and Le, 2019b] blood cell images, classification is used to differentiate the healthy ones from the abnormal ones present in cases of leukemia.

In general, image classification using CNNs is performed in two stages, the encoder stage and the classification stage (Figure 2.11). During the encoder stage, successive convolutions are applied to the input image until the image's relevant characteristics are extracted. After that, in the classification stage, generally, one or more fully connected layers are applied to those features to transform them into a probability/confidence vector. That is the basic structure followed by models explained in the previous section to introduce FCNNs, such as *Alexnet* [Krizhevsky et al., 2012], *VGG* [Simonyan and Zisserman, 2014], and *GooleNet* [Szegedy et al., 2015].

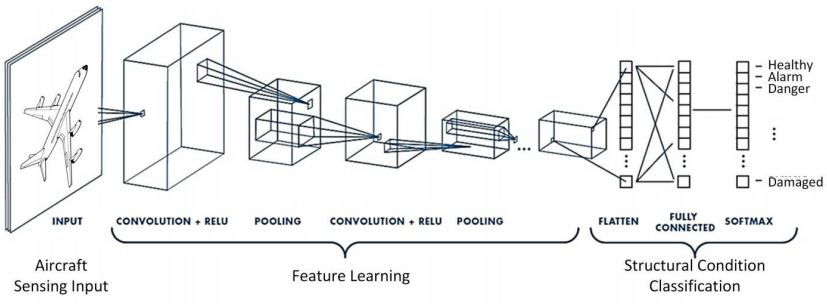


Figure 2.11: CNN architecture example [Tabian et al., 2019].

This type of network's versatility can be almost infinite, as long as a data set supports the training. For example, [Kagaya and Aizawa, 2015] classify images as food or non-food or in [Kim, 2014] for sentence classification within natural language processing. [Nakazawa and Kulkarni, 2018] uses a CNN to classify the twenty-two possible defects that can appear in wafer maps during semiconductor wafer manufacturing. [Saleem et al., 2019] provides an overview of deep learning classification techniques for plant disease detection. Mainly, this last one shows the multiple ways of approaching the same problem using deep learning, which means that the versatility of CNNs affects the multiple problems that can be solved and the multiple ways of solving the same problem.

As discussed in section 2.1, remote imagery is also one of the fields in which deep learning is most widely used. In [Kussul et al., 2017], the aim is to segment crop types from satellite imagery. However, it does not use an FCNN approach with encoder-decoder but only an encoder and classifier approach to categorize image patches according to the classes provided. In [Chen et al., 2014b], they also try to classify hyperspectral images, but in this case, using stacked autoencoders and a logistic regression layer employing the column vector of each hyperspectral image pixel as input data.

Out of what can be considered standard, it is also possible to classify something as mathematical as time series utilizing CNN, as shown in the paper [Gamboa, 2017] in section 4, where he reviews ways to transform the time series into an image and then classify them.

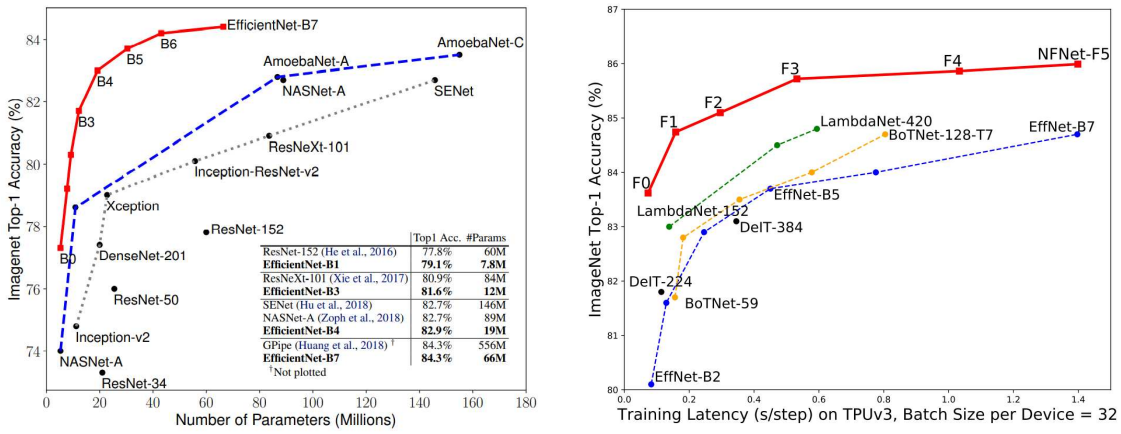
One of the most outstanding examples that classification through deep learning is still on the cutting edge is the recent publications of the Efficient-Net and NFNet architecture families [Tan and Le, 2019a, Brock et al., 2021], which beat the Top-1 accuracy metrics in ImageNet in their respective years of publication (Figure 2.12). The former is oriented to model scaling in order to optimize model size, training, and performance. This model balance is performed by the compound scaling technique, which establishes a ratio coefficient applied to the network's depth, resolution, and width. The latter, which has far surpassed the first, focuses on eliminating batch normalization layers through adaptive gradient clipping while maintaining the instability control benefits they provided and eliminating the drawbacks such as batch size dependency.

2.2.1 Deep learning for intersection classification

One of the latest works that can be found regarding the classification of intersections is the one carried out in [Kuo and Tsai, 2021]. In this work, images obtained from Google Maps roadmaps are used as a data source. They try to classify within five different typologies all existing intersections. For this purpose, the authors implement a multi-model system based on convolutional neural networks and combination rules.

The multi-model system consists of two convolutional neural networks that work together to classify the intersections. The first is a VGG16 model that performs a binary data classification between foreground and background images. Authors consider as foreground images all those in which there is a road intersection and as background images the rest.

[1] Image obtained from the paper [Brock et al., 2021].



(a) Efficient net family models Top-1 Accuracy. (b) NFNets family models Top-1 Accuracy.

Figure 2.12: Top-1 accuracy comparison between NFNet and Efficient Net.[1]

The second one takes the data provided by the previous binary classification model and tries to classify all those positive samples in five different types of intersections. For this purpose, an InceptionResnetV2 model is used, followed by a 6-layer fully connected classifier. This process is repeated three times taking three different pre-set image sizes, small, medium, and large.

Finally, for the consolidation of results, the authors propose filtering and combining occurrences of each one through rules that allow refining each one of the detections. Since the classification is performed using three pre-established sizes, these three sources of information are used to eliminate duplicates. For example, the detection is filtered when the same type is detected in the three sizes. However, a combination rule comes into play when a specific type is detected using a size window, and after enlarging the size window, a different type is detected.

More focused on the scope of this thesis, there are also publications and research on road crossing detection using data obtained from the ego-vehicle perspective. However, given the importance that, in our opinion, the subject deserves in view of the accident rates indicated in section 1, it seems surprising not to have found much more documentation.

One of the earliest works on the subject can be found at [Kushner and Puri, 1987], where two old-fashioned computer vision methods are presented. The first one extracts the road boundaries and tries to match them to a predefined intersection model using heuristic methods. Rather than recognizing the image road intersection, the second method finds free space corridors through which it is safe to drive.

Later, in 2013, Andreas Geiger published his Ph.D. [Geiger, 2013] in urban scene understanding. In his thesis book, two new approaches to intersection detection are proposed that, in many cases, can be considered the kick-off of the recent research on this field. The former is a new intersection modeling with flexibility in the number of intersecting streets and the location, orientation, and width of crossing branches. The latter efficient learning and inference algorithms based on Markov Chain Monte Carlo

Sampling and belief propagation are developed to infer scene design.

In different contexts to the above but with a similar approach, authors in [An et al., 2016] and [Ballardini et al., 2017] exploited RGB images from vehicle front-facing cameras and standard computer vision techniques to create temporally integrated occupancy grids. In turn, they compared them to predetermined shapes to assess the presence of upcoming intersections.

The approach proposed in the [Habermann et al., 2016] work is substantially different from the above ones. The data input is solely based on point clouds obtained by LiDAR. These raw data clouds go through three independent processes that will generate a binary classification of whether they belong to an intersection or not. The first process extracts the most relevant features from the point clouds. These features capture appropriate geometric and statistical properties for further classification using three different classifiers, an MLP, an *Support Vector Machine (SVM)*, and the AdaBoost algorithm at the second step. Since these methodologies produce a single frame classification, which can produce much noise, the third step of the methodology includes Hidden Markov's Models and CRFs that will model the sequence of observations.

[Baumann et al., 2018] work is closer to the scope of the present study and uses transfer learning techniques in CNNs to classify intersections. The idea behind transfer learning is using the learned feature representations from a network to solve a similar problem, usually training only the classifier part of the architecture (Figure 2.13).

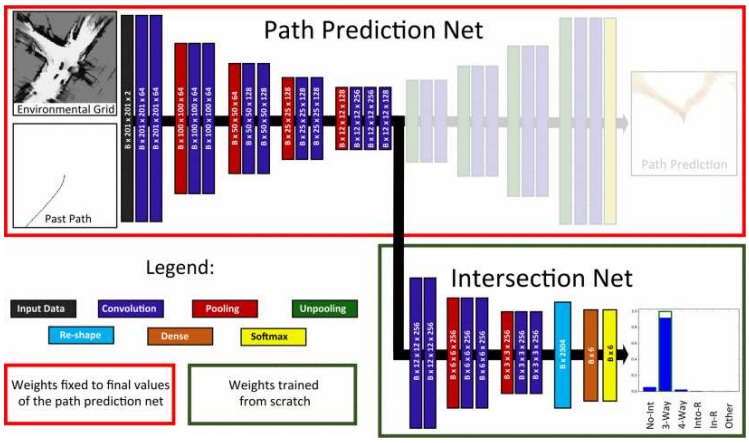


Figure 2.13: Transfer learning methodology example [Baumann et al., 2018].

In this case, researchers start from an FCNN trained to predict the future path of the ego-vehicle. The decoder is discarded, and all the encoder weights are frozen and will not be modified during training. Then the encoder will be in charge of extracting the feature sets from the input data. After that, the proposed architecture called Intersection Net, composed of a series of convolutional and pooling layers together with a fully connected layer at the end comes into play. This last architecture is the one that is trained from the obtained encoder's features. Its function is twofold, making the obtained features more specific and classifying them using the fully connected layer.

The work proposed in [Tümen and Ergen, 2020] also uses a personalized deep con-

volutional model to classify five types of multiple lane roads images. Three intersection types can be found within the labels: left separation, right separation, and multiple intersections. In addition, without being of interest to this particular thesis, crosswalks and empty roads can be found within the labeling.

The classification method is quite similar to other state-of-the-art approaches. However, the previous stages of the classification process are pretty interesting. Assuming that road classification from scratch is a difficult task, and there is a lack of disponibility of images to be implemented, the authors propose a two-stage methodology to ease the problem.

This procedure focuses on obtaining images in which the information not needed for the task has been cleaned, hypothesizing that this will increase the classification performance. For that hypothesis, the methodology focuses on extracting the image's RoI, in which most of the non-road elements are discarded. The first stage of the process consists of calculating Cr_1 and Cr_2, the levels from which the image will be vertically cropped. The authors calculate these two points by the vanishing point method. After that step, a canny axis detection filter is applied to the image to delimit approximately the road in parallel to this calculation. The combination of this delineation together with the applied cropping provides an RoI with much more valuable data for classification.

So far, the already research presented based on deep learning has focused on detecting or classifying intersections once they have been reached. However, as can be seen in the work [Habermann et al., 2016] that employs old-time computer vision, temporal integration can be an asset to be considered. In [Bhatt et al., 2017], this approach is given into consideration, and an *Long Short-Term Memory (LSTM)* [Hochreiter and Schmidhuber, 1997] is added to the well-known CNN for classifying intersections in RGB images. The difference between LSTMs and other types of architectures is that they can perform temporal integration since they have a "short-term memory" and a "long-term memory" that allows them to remember what they have processed previously to decide what they are processing now (Figure 2.14). The workflow of the final architecture is, therefore, as follows. First, a sequence of ordered images is supplied to the CNN, which will, in turn, return a sequence of feature vectors. These vectors are packed and sent to the LSTM, which integrates them temporarily to extract information to classify the sequence as an intersection or not.

[Koji and Kanji, 2019] work maintains a similar approach about temporal integration, but instead of detecting whether or not intersection exists, it tries to classify them into seven basic typologies.

For this purpose, it uses two data sources that the authors call Input-F and Input-T. Input-T is an RGB image taken just before reaching the intersection, and Input-F is a sequence of images taken while crossing the intersection. To process these two data sources, the authors use an architecture composed of three different modules, F-Net, TNet, and I-Net.

T-Net is a VGG16 network in which the transfer learning paradigm is applied. The classifier is replaced by two fully connected layers that are then trained to distinguish between the seven classes, freezing all the other weights. F-Net is a combination of an Inception-V3 and an LSTM architecture. Inception-V3 extracts the optical flow from the image sequence, and the LSTM classifies between straight, right-facing, and left-facing

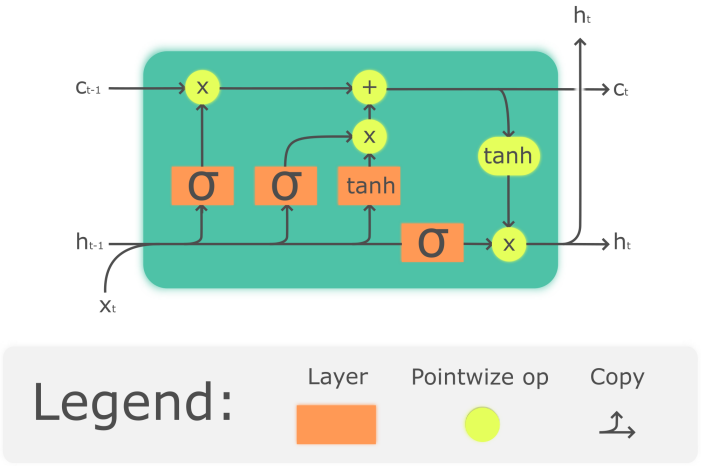


Figure 2.14: LSTM Cell architecture [Guillaume Chevalier, 2018].

crossings. I-Net is the architecture in charge of integrating previous results to return a final classification using Bayesian multimodal information fusion [Xu et al., 2016].

Reliable intersection detection is only possible if there are sufficient data sources from which to extract the features. However, it has been even more challenging to find labeled datasets with which researchers can carry out their work.

In [Ballardini et al., 2018], the authors propose a methodology to build datasets for intersection classification from existing datasets using GPS positioning and Open Street Maps. This combination of data sources allows the authors to establish a semi-automatic labeling methodology that significantly speeds up the dataset generation process and calculates the distance to the center of the crossing for each frame.

It is possible to mention that along with the arduous task of searching for data sets commented on before, the task of finding papers intersection classification has been no less complicated. Our opinion is that these difficulties are intrinsically intertwined and of notorious importance. Therefore, their causes and consequences for the research during this dissertation are discussed further during the presentation of the results in chapter 4.

2.3 Conclusions

As a conclusion of this brief investigation of the classification and segmentation techniques that are considered relevant for the development of this doctoral thesis, it can be observed that although their origin dates back to the dawn of artificial intelligence research, they are still valid today. It is an great research effort to minimally improve methodologies that have already been developed for so long. However, based on the accident data presented in Chapter 1, their improvement is still vital if the autonomous car is to become a real thing in the future. Both self-localization and free driving space identification remain critical points, and any slight improvement in the procedures capable of carrying out

these tasks is essential.

One of the approaches observed in work investigated focuses on the search for new proven techniques, such as novel architectures or training methods that improve the results of a particular problem.

Two clear examples of this are papers [Wang et al., 2019a] and [Tümen and Ergen, 2020] that use architectures wholly designed from scratch for the particular job. These approaches tend to work very well as a rule for particular tasks, and there is the benefit of gaining complete control over the architecture. However, they generally require a very high cost in resources, both computational and research time.

In [Yan et al., 2020], a slightly more straightforward approach is used. It employs an existing network architecture and applies it to a particular problem. This procedure facilitates the research by significantly shortening the design time, which is a great benefit. However, no pre-trained weights are used for the proposed architecture. That fact, as a rule, causes the training convergence to require substantially longer times. Another possible weakness is the data sources, consisting of point clouds, discarding RGB images. The implementation in a real system becomes much more complex due to the high costs of LiDAR devices.

The use of methodologies that have been used previously for other problems, adapting them to the specific matter to be solved or improved, is also a common approaches that can be found. Using that type of approach, the authors try to rely on development that has already proven its good results as the foundations of the new methodology that tries to resolve the proposed problem, which can also yield excellent results without the need for a large number of resources involved in research from scratch.

[Kuo and Tsai, 2021], for example, uses two already designed architectures such as VGG16 and InceptionResnet v2 and their pre-trained weights. This method makes the training process for both architectures much less time-consuming. However, given the nature of the input data, the final processing requires establishing a set of combination rules. This latter process must be specifically designed for the problem, and we believe it is not very useful for our particular problem. A very similar approach is used in [Koji and Kanji, 2019] work on the Inception V3 + LSTM and VGG16 architectures. It combines transfer and fine-tuning learning approaches, both focused on reducing training and development costs and can extract temporal information from the data. However, the whole system becomes slightly complex and cannot be easily trained end-to-end due to the *Bayesian Multimodal Information Fusion* method for merging the data from both architectures.

[Habermann et al., 2016] also uses temporal data integration, showing its usefulness in problems whose data can be easily obtained as ordered sequences. Despite using computer vision techniques outside of deep learning, he obtains excellent results. However, these are only for detection and do not fall within the classification field, which limits the information obtained and, like [Yan et al., 2020], only uses point clouds as input data.

[Baumann et al., 2018] uses the transfer learning methodology that has already been discussed and can give outstanding results and shorten development and training times. However, in this work, the input data are occupancy grids, which require a pre-processing that can be costly and not always accessible since three-dimensional measuring devices are used to obtain them.

[Bhatt et al., 2017] used the already debated temporal integration and fine-tuning of pre-trained weights seeking to find good results in sequence classification. It is an approach that is widely agreed upon since it is simple and can be trained end-to-end even though LSTMs are usually not pre-trained. However, one disadvantage is that it only focuses on RGB images, which are much more accessible data as a general rule, but usually will not be the only data source in autonomous systems. The results obtained are very encouraging; however, as in [Habermann et al., 2016], they refer to detection and not to classification.

Curation, pre-processing and data domain adaptation are also commonly used in the state-of-the-art researching. Most of the deep-learning-based investigations require a massive amount of data for the results to be optima, as has been seen by the importance that researchers tend to give to the dataset they are working with. However, this amount alone does not necessarily achieve good results. It is vitally important that the data provide as much information as possible to solve the problem, discarding as much noise as possible.

The [Zhang et al., 2018] and [Shamsolmoali et al., 2020] work focuses on the input data's domain changes through adaptations with GANs and Spatial Laplacian Pyramid Networks. These approaches are very valid and can obtain excellent results in generalizing the training data sets in order to obtain much more robust systems in the face of changes in the environment. However, the convergence of generative models tends to be very complicated and usually requires a long-term commitment.

[Liu et al., 2020a] work brings together several exciting approaches. It uses an already known and proven architecture, such as Resnet50, which allows pre-trained weights in the ImageNet dataset, with the benefits already mentioned. At the same time, a data fusion is implemented at the decoder level, which allows the use of several sources to solve the problem. Moreover, since the fusion is implemented at the decoder level, the modification of the network architecture does not affect the pre-trained weights, and the whole process can be trained end-to-end. The only drawback of the work is that the Resnet50 architecture is capable of extracting outstanding context features, but it is not specific for extracting spatial features, which are very important in some tasks.

As a final though, obtaining results is not the only thing that must be taken into account when approaching deep-learning research; efficiency is crucial. Before initiating the development of the methodology, or during the same one, it is necessary to establish the training process's optimization methods, as it has been observed. The proposed solution's efficiency, whether it is a new architecture, a new data pre-processing system or a new training methodology, is critical to obtain results in a reasonable time.

Table 2.1. below shows the results of the papers analyzed during the research process of this doctoral thesis. One thing to note with respect to the classification of interactions, in particular, is the lack of existing data sets. Of the explored methods shown in the table, five of the seven works use the KITTI dataset, which is not even specific for the task.

Reference	Methodology	Dataset	Results
Road semantic segmentation methods			
[Shamsolmoali et al., 2020]	Domain adaptation and Feature Pyramid Network	DeepGlobe Road Extraction, Massachusetts Road, EPFL Road Segmentation	69.58%, 78.85%, 81.68% (IoU)
[He et al., 2019]	Domain adaptation and Transfer learning	WorldView-3	46.8%, 43.2%, 36.1% (IoU)
[Fan et al., 2018]	Roll angle estimation	EISATS	$\approx 0.0647°$
[Zhang et al., 2018]	Semantic information complement with GAN	UAS, Kitti-ROAD, Cityscapes	98.48%, 97.19%, 98% (PA)
[Wang et al., 2019a]	OFSRNet, Spatially dependant loss function	Kitti-OFRS	92.2% (F1)
[Liu et al., 2020a]	Encoder-Decoder, Data-fusion	Kitti-ROAD	93.98% (F1)
[Yan et al., 2020]	Encoder-MultiDecoder LMRoadNet	MultiRoad based on SemanticKitti	94.2% (F1)
Intersection classification method[a]			
[Habermann et al., 2016]	Feature extraction + ANN | AdaBoost | SVM + HMM | CRFs	Carina 2, Kitti	87.89%, 91.04% (Acc)
[Baumann et al., 2018]	Encoder + Classification & Transfer learning	Own	72% (Acc)
[Bhatt et al., 2017]	CNN + LSTM	Oxford Robot Car, Lara	72.05%, 78.25% (Acc)
[Koji and Kanji, 2019]	CNN + LSTM + Bayesian multimodal information fusion	Kitti	42% (Acc)
[Ballardini et al., 2017]	Stereo vision, CRF, TextonBoost, Visual odometry	Kitti	39% (Acc)
[Kuo and Tsai, 2021]	CNN + LSTM + Bayesian multimodal information fusion	Kitti	42% (Acc)
[Tümen and Ergen, 2020]	Stereo vision, CRF, TextonBoost, Visual odometry	Kitti	39% (Acc)

Table 2.1: A comparison of deep learning segmentation and classification methods.

[a]The results presented are not entirely comparable since they depend on the number of classes proposed in each investigation.

2.4 Objectives

After the review of the state-of-the-art, and considering the discussion presented in the introduction and the conclusions exposed in the last point, the objectives of this thesis are as follows:

1. To build a road segmentation system that efficiently recognizes all the space where an autonomous vehicle can drive without risk for both the driver and the surrounding users.
   (a) Constructing a network architecture based on an already developed and tested network architecture.
   (b) The developed architecture must be able to integrate various types of data efficiently without limiting the end-to-end training of the architecture or the possibility of using pre-trained weights.
2. Implement an intersection classification system in the urban environment that allows an autonomous vehicle to obtain sufficient information for self-localization and safe navigation of intersections.
   (a) Exploring existing temporal integration methods and training techniques that allow faster convergence without loss of efficiency for the particular task.
3. Establish improvement methods that optimize the systems used to achieve the above objectives and achieve the most outstanding possible efficiency in extracting the dataset's relevant information.
   (a) The development of guided optimization techniques allowing the use of the proposed systems more efficiently.
   (b) Using data pre-processing techniques for three-dimensional information data sources that improve the information extracted from it focused on resolving the road segmentation problem.

Chapter 3

Road Semantic Segmentation

As previously stated, one of the main objectives of this thesis is the detection of the entire surface over which an autonomous vehicle can circulate. With such purpose in mind, this thesis research intends that once implemented on a large scale, the autonomous vehicle can reduce the large number of accidents that occur by a departure from the road. The exhaustive research carried out in chapter 2 has allowed us to conclude that the best way to approach this task is to employ semantic segmentation techniques through deep learning, which is one of the most widespread and best-performing techniques.

3.1 Introduction

Semantic segmentation is a specific computer vision technique for classification. Unlike standard classification, which tries to assign a label to one or multiple objects in an image, segmentation tries to classify each image's pixels by assigning them a label and forgetting about the object that those pixels may form themselves. With that in mind, it is possible to know which part of the entire image's pixels set belongs to a label and its occupation within the image and, therefore, the environment.

Usually, as seen in Chapter 2, semantic segmentation is carried out through deep learning techniques, explicitly using FCNN. Briefly introduced earlier, this type of networks usually have an encoder-decoder structure, (Figure 3.1).

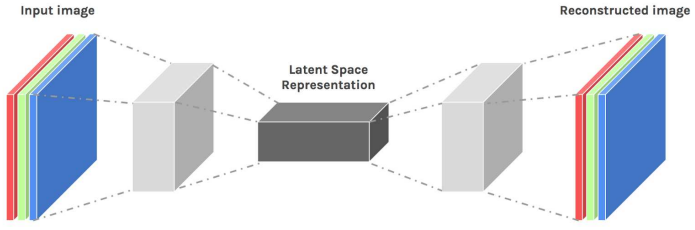


Figure 3.1: Encoder-Decoder example.[1]

The encoder is the part of the network in charge of encoding the input data to obtain a set of features as representative as possible of the input data. As in classification networks,

[1]Image obtained in https://hackernoon.com/autoencoders-deep-learning-bits-1-11731e200694

that extraction is usually done by convolution layers. Convolution is the process of adding each element of the image to its local neighbors, weighted by the kernel, and can be denoted by the mathematic expression 3.1 where $g(x, y)$ is the filtered image, $f(x, y)$ is the original image, and w is the kernel.

$$g(x, y) = w * f(x, y) = \sum_{dx=-a}^{a} \sum_{dy=-b}^{b} w(dx, dy) f(x + dx, y + dy) \tag{3.1}$$

The kernel weights are responsible for extracting or enhancing certain image features, which will be highlighted in the output image. An example of this would be the *sobel operator* [Sobel, 2014], which applies two kernels to an image to obtain the gradient intensity at each pixel and, therefore, the probability that an edge exists.

During the training process of a CNN, the weights of the kernels of each convolutional layer are modified so that, employing convolution operations, they can extract the necessary information to solve the problem. The information throughput is usually done sequentially so that the deeper convolutional layers extract the information from the features that have been extracted by the layers immediately above. During this process, the encoder usually seeks to compress more and more features so that the final result is grouped into a single feature vector or a series of small two-dimensional arrays. This compression is usually done utilizing variables such as stride or padding in convolution and pooling layers, eq 3.2 and 3.3.

$$H_{out} = \frac{H_{in} + 2 * padding[0] - dilation[0] * (kernel\_size[0] - 1) - 1}{stride[0]} + 1 \tag{3.2}$$

$$W_{out} = \frac{W_{in} + 2 * padding[1] - dilation[1] * (kernel\_size[1] - 1) - 1}{stride[1]} + 1 \tag{3.3}$$

The decoding part of the network is in charge of transforming the set of features extracted by the encoder into a two-dimensional matrix of the same size as the input image and with a number of channels equal to the number of classes into which it is being classified plus one, which represents the empty class. Each channel will contain a confidence map indicating the confidence that each pixel belongs to the class that this channel represents. The opposite operation to convolution, deconvolution, or transposed convolution, usually performs that process denoted by the expression 3.4 As in the encoder, the weights of the deconvolution layers can be trained, but in this case, they can be fixed, e.g., as a bilinear filter.

$$g(x, y) = w * f(x, y) \tag{3.4}$$

As in the convolutional layers, the deconvolutional layers are usually sequential in the decoder, so the input data usually comes from the decoded features obtained from the previous layer. Some more complex architectures add shortcuts between the encoding and decoding phase or between different layers of the same phase that seek to avoid gradient vanishing and improve results, as seen in some working examples in Chapter 2.

3.2 Method

With the idea of how semantic segmentation usually works in mind, during the following section and its corresponding subsections, it will be explained the proposed methodology and its evolution from the beginnings through a simple encoder-decoder architecture to the final 3D-Deep and 3D-Deepest proposals.

3.2.1 Road Segmentation

The beginnings of this doctoral thesis are based on the work by Jesús Muñoz-Bulmez about road segmentation in 2017, [Muñoz-Bulnes et al., 2017]. That research implements a road segmentation system using an FCNN architecture based on ResNet101 and random data augmentation.

The first point of enhancing this work to improve the results obtained is in the selected dataset. The dataset used to implement the work is the KITTI dataset for road segmentation. This dataset contains a total of 579 road images divided as follows: 289 for training, with their respective labels, and 290 for testing. This number of images is clearly far from optimal, which is why data augmentation is so necessary, so as a first approximation, image increase is a clear choice.

For this purpose, the chosen dataset is Cityscapes [Cordts et al., 2016] that contains 5000 images with fine annotations and 20000 images with coarse annotations. Unlike KITTI, the Cityscapes dataset is labeled with thirty classes, so in order to use it, the labeling had to be modified. Following the KITTI labeling policy, the road class is kept, the other classes are considered suitable terrain (not road), and the void classes are considered invalid terrain.

After grouping both datasets' training sets, several trainings are performed using thirty random images of the KITTI dataset as validation. These trainings obtained substantially worse results than those achieved in the initial work [Muñoz-Bulnes et al., 2017]. The trainings were performed using the same methodology used in the original work, CAFFE, together with its python interface. The Data Augmentation used is also the same, as it is considered perfectly suitable.

3.2.2 Intersection Segmentation

In turn, as an improvement of the initial road segmentation work, the possibility of training an intersection segmentation network was considered. The purpose of this network is that when arriving at an intersection, the vehicle would be able to instantiate the different terrains through which it could circulate to be later on used for the vehicle's navigation. A specific example would be a T-shaped intersection. This type of intersection has two possible trajectories: one on the right arm of the intersection and the other on the left arm. The idea behind that research is that a network can segment the road belonging to each of the arms and subsequently understand what would be a left turn.

For this research, and since no available dataset was labeled in the way that was needed, a set of recordings was made with the INVETT laboratory vehicle. Since this work was intended to be a proof of concept to validate whether it is possible to implement a

(a) Image of the segmentation of the main road in a roundabout.

(b) Image of the segmentation of the inner road in a roundabout.

Figure 3.2: Roundabout instance segmentation. The green pixels are true positive samples, blue pixels are true negative samples and red pixels are false negative samples.

system such as the one proposed in the hypothesis, approximately 300 images were labeled from the recorded videos. The labeling is conducted similarly to the KITTI training set, containing various types of intersections, such as T-intersections, cross intersections, or roundabouts. The training results are not as intended, as can be seen qualitatively in the images (Figure 3.2). Unlike the road segmentation without instantiation, the network seems to have severe problems delimiting which part of the road belongs to which section of the junction. These poor results seem to be somewhat logical because using RGB images, in many cases, it is difficult to determine where each section of a junction ends, even for a human.

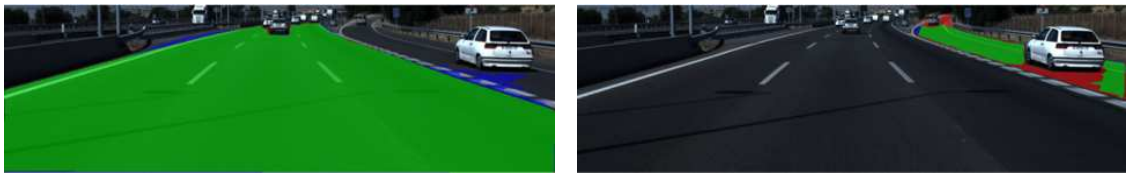
A prominent hypothesis seems to be that urban intersections have a complexity that the network has not been able to deal with. In order to validate this, a way to reduce the complexity of the junctions to be segmented is sought. One of the simplest intersections that can be found on the road is incorporations and detours on a highway. This type of crossing in general always has the same structure: the main road, frequently with several lanes, and a junction or a detour, usually with one or two lanes whose trajectory is secant to the main road.

As with the prior hypothesis, a proof of concept is performed by training two well-established and previously used networks, ResNet 101 and ResNet 50 with FCNN implementation, to segment both the main road and the junctions/turnouts that appear on the road. A series of sequences are recorded to perform the training with the INVETT laboratory car on the Spanish M-40 highway, a Madrid ring road with fine intersection examples. After that, and as with the city segmentation, about 300 examples are labeled in which both the main road and the road belonging to the intersection or detour are marked.

The results seem promising both numerically and qualitatively (Figure 3.3); however, it can be observed that there are still some problems with the delimitation of the respective lanes, which may cast doubt on their use in production. Therefore, the next step decided to go back to basics, implementing a segmentation system that can be considered state-of-the-art.

3.2.3 Working hypothesis

Considering the results obtained in sections 3.2.1 and 3.2.2, and given that the associated working hypotheses could not be validated, it is intended to improve them by means of a



(a) Image of the segmentation of the main road in a highway.

(b) Image of the segmentation of the detour lane in a highway.

Figure 3.3: Highway instance segmentation. The green pixels are true positive samples, blue pixels are true negative samples and red pixels are false negative samples.

new approach.

The previous results have been obtained with a network architecture that only uses RGB images as input data. However, currently, there are many other data sources, such as disparity or point clouds obtained by LiDAR sensors, that we believe can provide information that is much more difficult to appreciate in RGB images. Knowing that precedents and based on them, the following hypothesis is formulated. If a network architecture that gives good results using RGB images is selected and modified to accept other types of information such as disparity maps or point clouds, semantic segmentation results obtained so far can be improved.

3.2.3.1 Backbone architecture

As shown in Figure 3.4 and explained in previous sections, convolutional neural networks used for semantic segmentation traditionally extract all the context from the input data by convolutions to arrive at a score vector. This vector is through which, usually with transverse convolutions, the final probability map is obtained. Occasionally, additions can be made to the architecture to improve its performance, in this case, intermediate connections, which try to solve gradient vanishing.

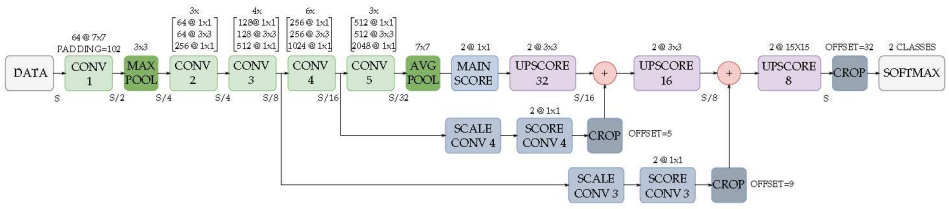


Figure 3.4: Fully convolutional network implemented on ResNet50 architecture.

Although they work reasonably well, these types of architectures tend to forget the input data spatial information, which can be considered a drawback. Some research has found that spatial information obtained through a large receptive field is as key as context information to obtain high-grade results [Zhao et al., 2017, Chen et al., 2017].

This assumption leads us to search for an initial architecture to implement the update

proposed in the hypothesis that can integrate context information and spatial information effectively and efficiently. It is possible to think that developing an entirely new architecture would be an idea to consider. However, building a network architecture from scratch requires a level of both time and analysis that was utterly out of reach when the problem was posed. Therefore, the solution that seemed most reasonable when the decision was made was to select a network that performed well in semantic segmentation and met the necessary prerequisites. These prerequisites are met with the BiSeNet architecture [Yu et al., 2018], designed explicitly for semantic segmentation, and having two distinct branches for each information type: context and spatial.

As shown in Figure 3.5, the network architecture is differentiated into two branches that extract information independently from the images, the context path and the spatial path. The context path is in charge of extracting the context features. These features must be sufficiently rich and high-level to provide the necessary information for image segmentation. In addition, and given its vital importance, a global average pooling layer is included at the end of the branch to obtain a sufficiently wide receptive field from the extracted features. This branch can be built with most of the available architectures of a classical convolutional neural network, such as VGG or ResNet.

On the other hand, the spatial path is responsible for obtaining sufficient spatial information, for which it is essential to maintain as far as possible the original spatial size of the image. For this purpose, a series of convolution operations are performed, followed by batch normalization and activation functions that return a feature map with a size of 1/8 of the original image. The features of both branches are fused using the Feature Fusion Module that will be depicted further.

The results presented by the researchers in the BiSeNet article are sufficiently exciting both in terms of computational speed and accuracy to consider it a more than adequate architecture to take as a reference for the subsequent phases of this research.

3.2.3.2 3D-Deep

Once a valid network architecture has been selected, the main idea is to update it so that it can work simultaneously with image data and three-dimensional data, such as point clouds or disparity maps, as shown in Figure 3.6.

Initially, the most trivial transformation to think of is increasing the number of input channels of the first convolutional layers of both branches. This modification would allow input images to store one or more channels containing the three-dimensional information after the three main RGB channels. However, this approach had certain disadvantages that dismissed it for use. The increase of channels at the input of the convolutional layer allows extracting combined features from all the input data. However, this augmentation, in addition to substantially increasing the number of parameters, limits the possibility of extracting specific features on three-dimensional information. These features can be considered crucial due to the importance of understanding the location of road curbs to segment the road edges accurately, [Wang et al., 2019a]. These city elements usually have a noteworthy difference in altitude with the road but are not always marked in color/-texture. In addition, since BiSeNet implements a specific feature fusion module, it seems reasonable to extract the features by different paths and then merge them accurately.

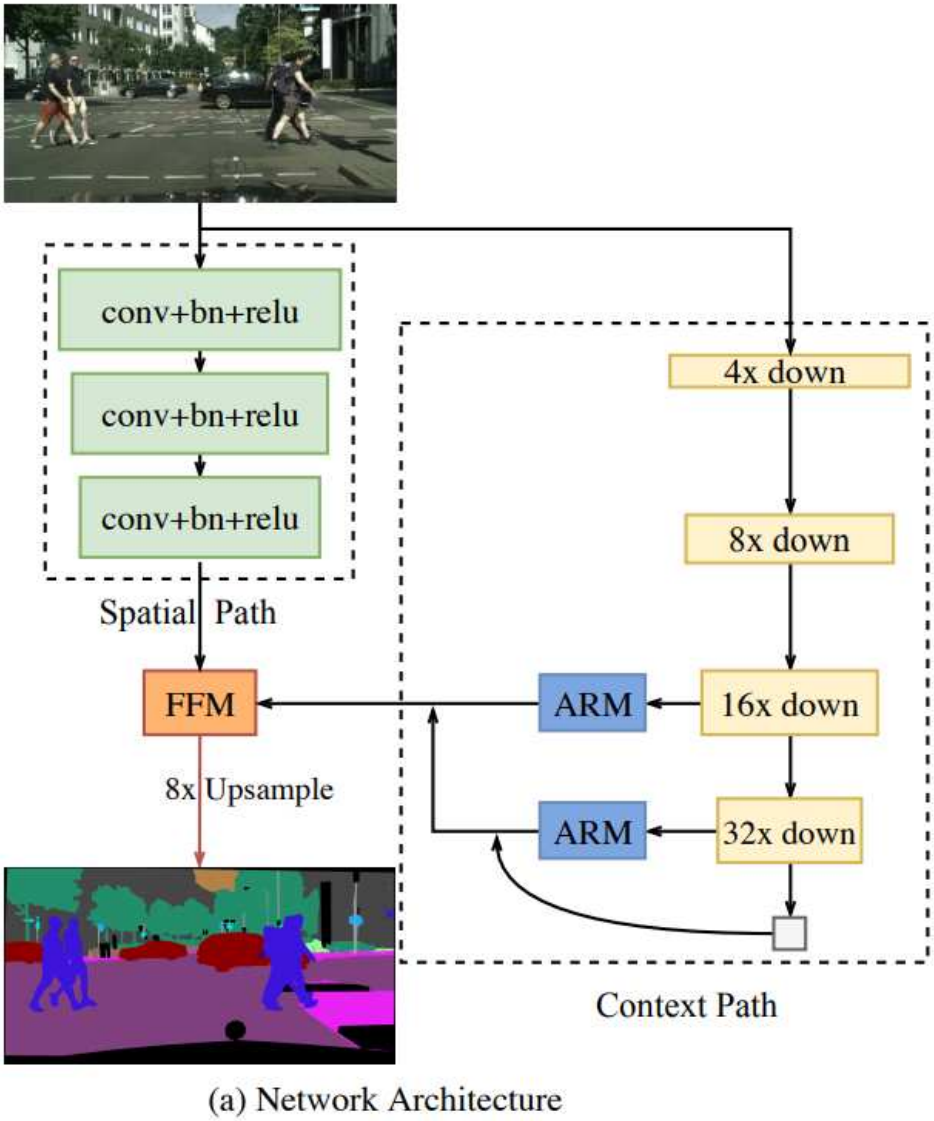


Figure 3.5: BiSeNEt Architecture.

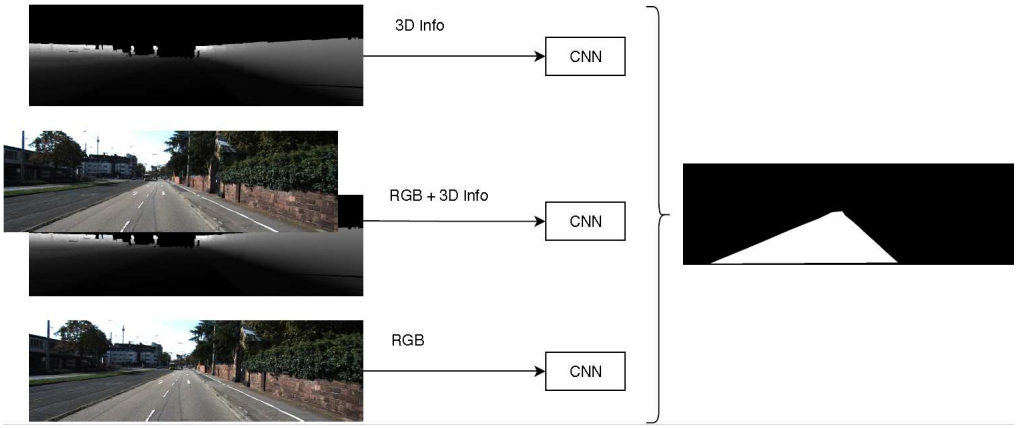


Figure 3.6: Flow diagram of the system.

Therefore, this chapter will present a new branch to be added to the BiSeNet architecture, the deep context path, which will extract all the context information that can be obtained from the three-dimensional data. This modification, however, leaves aside the spatial information that can be obtained from the three-dimensional data. These data can also provide quality information, especially with the scene's geometry. Nevertheless, for the sake of simplicity, and considering in this case that the information is compatible with that obtained from the RGB images, the spatial path is in charge of processing the RGB data and the three-dimensional data together.

These two modifications led to creating a new architecture, shown in Figure 3.7, which will be the focus of our semantic segmentation research throughout this chapter, and which will henceforth be denoted as 3D-Deep. With this architecture, which will be explained more in detail further, a series of tests and experiments were carried out to validate its correct performance and usefulness when used in road semantic segmentation.

3.2.3.2.1 Context path

The context path has the same function as it has in the original architecture, and it is intended to obtain all the context information through convolution operations. If we look at a basic architecture like the one in Figure 3.1, the context path would be the encoder. There are many possibilities within the networks belonging to the state-of-the-art to choose as context path, may consider practically any classification network. In this case, ResNet architectures [He et al., 2016] were used as the initial basis for continuity with previously conducted research.

ResNet architectures consist of 5 convolutional blocks applied sequentially on the data reducing it to a vector of characteristics of size 2048 in its versions 50 101 and 152 and 512 in its versions 18 and 34. These vectors store all the context information and will usually be used to classify the data.

In our case and for the sake of simplicity, the data extracted from the context branch come from the same layers as in the original architecture. The feature maps of convolutional blocks 4 and 5 are extracted and used later in the attention refinement module

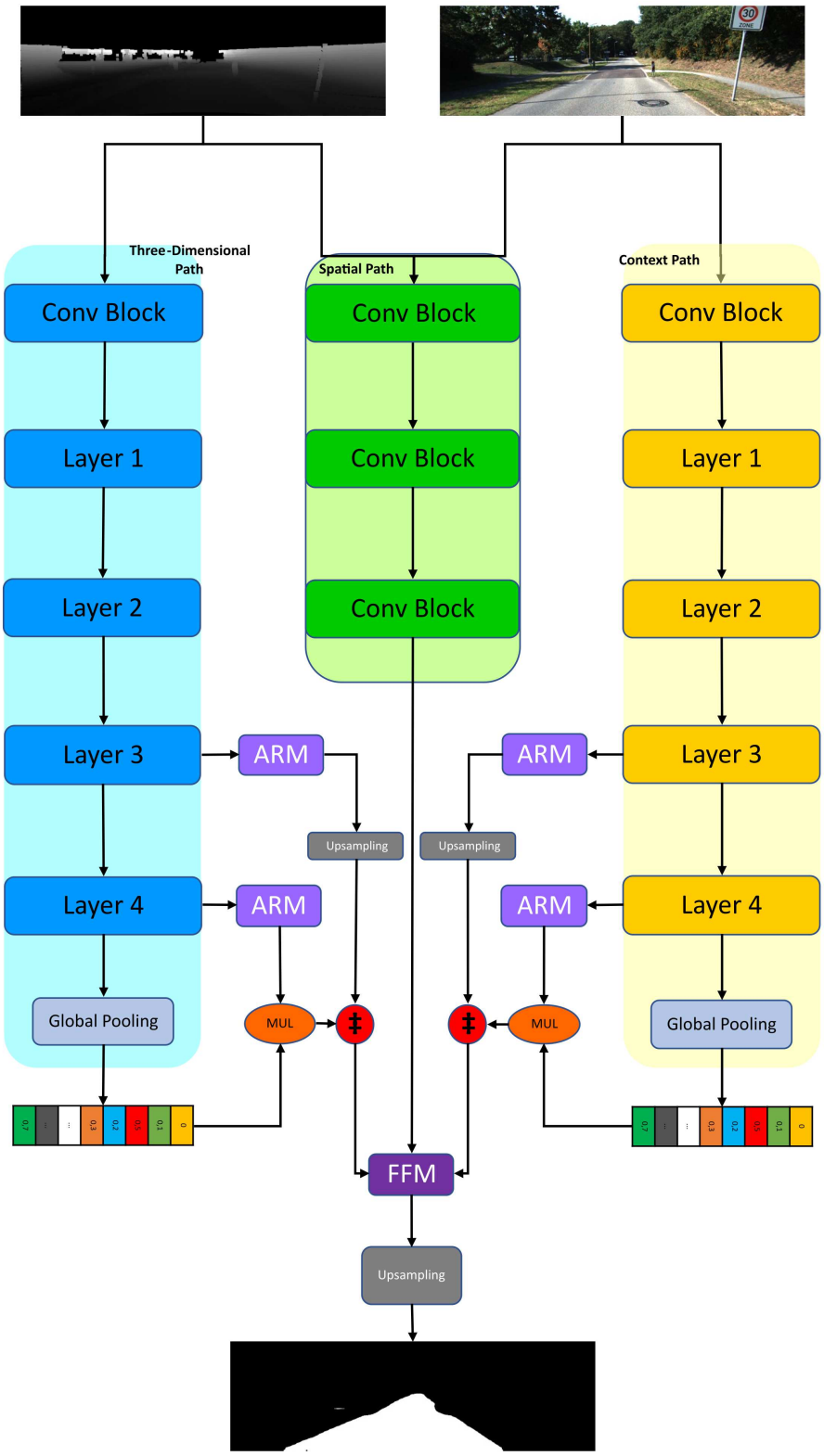


Figure 3.7: 3D-DEEP network architecture.

and the loss function. In addition, a global average pooling operation is applied to the features of block 5, whose results will also be used to obtain the final segmentation.

3.2.3.2.2 Three dimensional data transformation

Before going into detail with the depth context branch, it is advisable to focus on the three-dimensional data. As mentioned above, the idea behind generating a new architecture is to deal with three-dimensional data in order to obtain new sources of information that can improve segmentation results. However, three-dimensional data cannot always be obtained in the same format or by the same means. Point clouds obtained by LiDAR or disparity maps obtained by stereo vision are a clear example of this. The shapes of these data types are entirely different, and it is impossible to process them raw with the same convolutional layer. A three-dimensional convolutional layer would be needed to process a raw point cloud, and the data would have to be adapted to a grid. However, disparity maps are a two-dimensional representation of three-dimensional information obtained by stereo vision, and in general, a two-dimensional convolutional layer will be used for processing it.

It is not always possible to choose the data sources, so the idea is that the proposed architecture should be able to work with both types of data prior to processing. Considering that using three-dimensional layers may cause the number of trainable parameters to be larger than desired and that transforming two-dimensional data into three-dimensional data is not a trivial process, it has been decided to transform point clouds into two-dimensional data. Moreover, it is a process that seems to give good results when segmenting roads, [Chen et al., 2019].

Usually, the road surface does not have significant differences in altitude, especially in short distances, which, however, the margins usually have due to the curbs of the sidewalk. This idea allows us to focus on the altitude as a determinant feature to enhance the value of the data obtained through point clouds through the necessary transformations. The data adaptation process will therefore consist of the following sub-processes:

1. Projection of the points from the LiDAR coordinate system to the two-dimensional coordinate system, in this case, the one belonging to the camera of the RGB images.

2. Use the height (Z coordinate in the LiDAR) to establish a significant difference between the points at road level and those that cannot.

3. Fill in the spaces where the LiDAR resolution has not been able to obtain information.

The first process is relatively trivial. It is necessary to know the rotation and translation matrix between the LiDAR coordinate system and the camera coordinate system and the projection matrix that will convert the coordinates into pixels.

Each of the cloud points is multiplied by the rotation and translation matrix. Then the last row of the result is discarded, and the rest is multiplied by the projection matrix. This last operation returns the coordinates of each point in the camera system (pixels) and a scale factor in the third row. Then it is necessary to divide each point by its

scale factor to obtain the final scaled coordinates. Once all the points are obtained, it is essential to perform filtering to discard all those outside the image frame. All this process is shown in the Eq 3.5

$$\begin{pmatrix} x_v \\ y_v \\ z_v \\ 1 \end{pmatrix} * \big(R|t\big) = \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} \tag{3.5a}$$

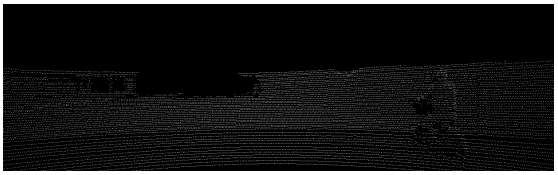
$$\begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} * \big(P\big) = \begin{pmatrix} s * x \\ s * y \\ s \end{pmatrix} \tag{3.5b}$$

This first procedure produces a black and white image in which one white point appears for each projected point of the cloud. However, as mentioned above, it is believed that more helpful information can be extracted. As introduced above, the road, as a general rule, does not have abrupt changes in height. In addition, it is usually located in a low position compared to the rest of the scene. That is why height is the metric chosen to obtain more helpful information. It is represented by one of the coordinates in the reference system of the point cloud. In order to carry this information to the transformed points, each point intensity value is chosen since the generated image is in grayscale. First, the previous filtering must be done to discard those points that are not useful or directly outliers. This filtering will be done by two values: the angle of view and the height value given by the corresponding coordinate. The last is essential since the road will not be much more distant from the center of coordinates than two meters since it is always measured from the vehicle. All points with a lower value are of no interest to us and are discarded and will generally be measurement errors since it is rare that there is anything below the road.

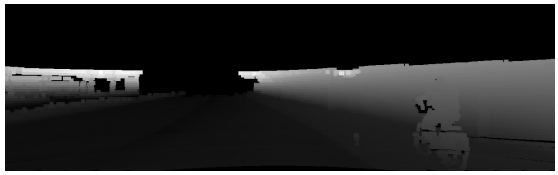
On the other hand, the angle of view is essential to discard other points that generally do not matter to us in the vertical range, such as the sky or excessively high points, such as the top of a building, and at the same time maximize the information in the horizontal angle that allows us to recognize what is to the left and right of the vehicle. A study was conducted to find the optimal values resulting in the ideal horizontal angle ranging from -60º to 60º and the vertical angle ranging from -14º to 3º. All points not in this range are also filtered out.

Once we have selected the points to be worked on, their height value will now be normalized, so in order to transform it into an intensity value. For this purpose, the maximum and minimum height values are selected, and all values are standardized between 0 and 1. These values will be used as intensity when projecting the points in the camera coordinate system. The final result is an image like the one shown in Figure 3.8a, in which the higher points will be brighter, and the lower points will be darker. In Figure 3.8a, the intensity of the projected points was modified for visibility reasons.

As can be seen in Figure 3.8a, the image is essentially black. This lack of data is because the number of projected points will depend on the LiDAR resolution with which the data is being taken, and this is usually not comparable to the resolution of an image. Since it is considered that an image in which most of its points do not provide any



(a) Normalized LiDAR projected points.



(b) Elevation pattern image.

Figure 3.8: Three-dimensional transformation. On the left, LiDAR projected points according to their elevation. On the right, after applying the dilation.

information may not be the best way to help in the segmentation process, it is necessary to extend the information of each point to its neighbors that do not have it. For this purpose, a dilation operation with a kernel of size 9 x 9 completes part of the missing information satisfactorily, as shown in Figure 3.8 image. This set transformation gives us what we called an *Elevation Difference Images (EDI)*, a grayscale image in which the elevation differences obtained from the LiDAR data are adequately represented.

The EDI image, in our opinion, provides more information to the neural network, and these transformations also make the architecture compatible with various three-dimensional data sources, such as point clouds or disparity maps, as intended.

3.2.3.2.3 Deep context path

The depth context branch will be in charge of processing the three-dimensional data in the form of an image, as introduced in Section 3.2.3.2.2. Like the context branch, its purpose is to extract a set of features from the three-dimensional data through a set of convolutional blocks. These features are intended to contain as much context information as possible of the 3D data.

This branch is designed to mirror the context branch, so its structure will be exactly the same. Therefore, the architecture selected from the possible ResNet models for the context branch will be the architecture for the three-dimensional context branch, and the size of the feature vectors will be the same. Accordingly, the features will be obtained from the same points of the ResNet architecture: the feature maps of the convolutional blocks 4 and 5. As in the context path, these features are the ones to be sent to the respective attention refinement module. In addition, and as in its twin branch, global average pooling is applied on the features returned by block 5 to obtain the model's tail that should capture the full global context of the data.

3.2.3.2.4 Spatial path

As in the original architecture that serves as its backbone, the spatial path is the part of the network architecture responsible for having sufficient resolution to obtain spatial information as representative as possible from the data. There are other approaches capable of processing/extracting spatial information in semantic segmentation, like [Zhao et al., 2017] or [Yu and Koltun, 2015]. However, in order to keep the development procedure as simple as possible, try to maintain the processing speed, and not increase

memory consumption, we have tried to keep it as similar as possible to the BiSeNet architecture.

As in the original architecture, the spatial path contains three convolutional layers. However, the ones belonging to the proposed architecture contain slight differences that make them suitable for the new input data type. Each layer includes a convolution with stride 2, followed by batch normalization [Ioffe and Szegedy, 2015] and ReLU [Glorot et al., 2011]. In order to adapt the layers to the *RGB-E* and *RGB-D* data inputs, the number of input channels of the first convolutional layer was increased from three to four.

These minor modifications allow maintaining the initial idea of making prevail the original spatial size of the data and coding sufficient spatial information since the size of the output feature maps of 1/8 of the original size of the image is maintained. In our opinion, these maps are large enough to obtain rich spatial information.

3.2.3.2.5 Attention refinement module

Attention mechanisms have become one of the essential concepts in the field of deep learning. In convolutional architectures, the attention mechanisms are developed to guide the learning process using high-level information that *focuses* on relevant features and filters the excess data. They try to emulate one of the most critical mechanisms of complex living beings, as the name suggests, the attention mechanism. This system is the one that allows living beings to filter the excess of information coming from the environment and focus their attention on the distinctive parts that best allow them to process the situation.

In our case, the attention refinement module is in charge of implementing this function in the 3D-Deep architecture. There are many different attention mechanisms, but for the sake of simplicity, the same as in the backbone has been used, a self-attention variant of the *Convolutional Block Attention Module (CBAM)* methodology [Woo et al., 2018].

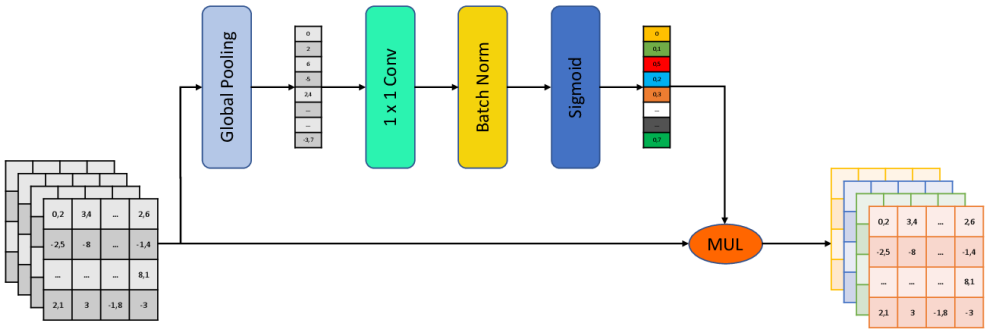


Figure 3.9: Attention Refinement module depiction.

The proposed module only focuses on channels, unlike the CBAM module, which has spatial and channels attention. As can be seen in Figure 3.9, the operation is simple. First, a global average pooling is performed, returning a vector of size C, the number of channels. Then a convolutional layer is applied with a kernel of size 1x1, which will try to detect the most remarkable high-level features. Then, a batch normalization layer and

a sigmoid function are applied to the convolution results. The resulting vector of weights is used to balance the importance of each channel.

3.2.3.2.6 Feature fusion module

Since the proposed architecture processes the different types of data in different paths, it is necessary to integrate all the features obtained by each branch to perform the upsampling process that allows us to obtain the probability maps for each class. For this purpose, the feature fusion module proposed in BiSeNet [Yu et al., 2018] has been used with slight variations in order to be able to work with the features of three branches instead of two.

As shown in Figure 3.10, the structure is practically the same as the original one, with the only difference being that the number of input channels is higher, as they come from three different branches. This increase means that the first convolutional block must be modified to accept a different channel number than the one used in the original version. For example, if the context path chosen is a ResNet50, the first convolutional layer must accept data inputs of 6400 channels. The calculation of this number is given as follows. The fourth convolutional block of the context path provides 2048 channels, and the third convolutional block provides 1024. On the other hand, since it mirrors the context, the depth context path provides the same number of channels. Finally, the spatial path returns 256 channels. Adding them all together gives the 6400 channels that, in this particular case, must be accepted by the convolutional block. These modifications will always depend on the architecture selected for both context branches.

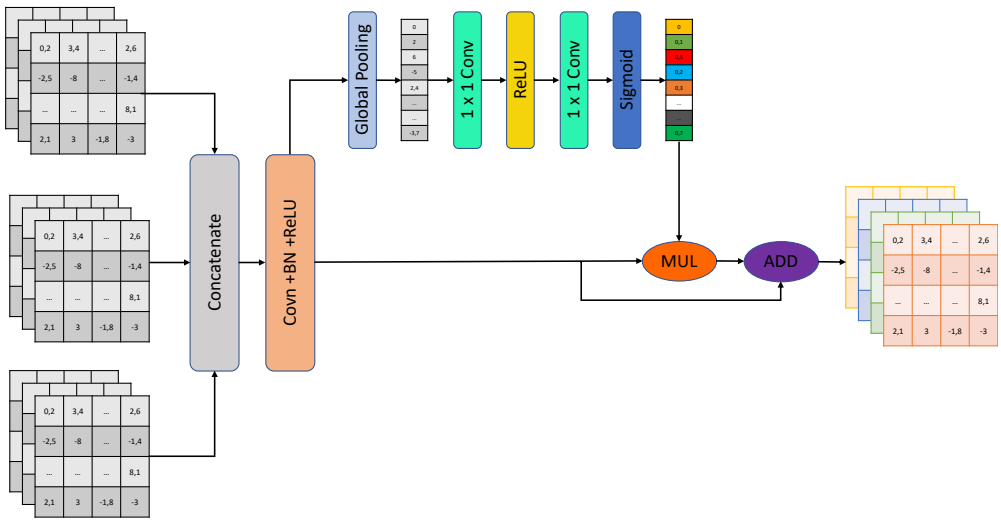


Figure 3.10: Feature Fusion Module depiction.

Going a little more in detail, the idea of the module is the following. Once all the characteristics have been concatenated, the first block tries to re-balance them employing the sequence: convolution + batch norm + ReLU. After them, by means of a squeeze and excitation block, similar to those proposed in [Hu et al., 2018], a vector of weights

is computed that re-balances the features so that they are selected and combined in the most appropriate way possible. This vector of weights is computed using global feature pooling and convolution operations. In the end, a sigmoid function is used in the vector to normalize the weights. These merged and refined features are the ones that will be upsampled to obtain the confidence maps of each class, and therefore the final prediction of the network.

3.2.3.3 3D-Deepest

After the results achieved with the 3D-Deep architecture were presented in [Hernández et al., 2020] and presented in detail in Section 3.3.2, the next logical step was to explore the possibility of updating/optimizing the architecture.

Optimization is one of the most critical issues in neural network development. It is useless to find a model that solves a problem with practically total effectivity if it is necessary to invest an excessive amount of resources to implement it in production. To the extent of our knowledge, most systems based on neural networks require a lengthy training process, especially if the training is done from scratch and transfer learning methods are not used.

There are two primary forms of optimization within artificial neural networks, structural modifications (architecture) and optimized searching to find the best hyper-parameters that allow the model to converge (learning rate, momentum, optimizer). For example, two of the most successful architectures, [Tan and Le, 2019a], and [Brock et al., 2021], focus mainly on optimizing the architecture by reducing the number of trainable parameters. Reducing their number decreases training times substantially, and, in these two particular cases, it also increases the model's accuracy.

Architecture optimizations may also begin in its construction phase, an approach that can be seen in the work of [Barrios et al., 2001] and [de la Hoz Galiana, 2020]. Both investigations try to encode the network architecture so that the new codification can be treated by an optimization algorithm, in this particular case, by genetic algorithms. Another possible option for optimizing architectures is to start with an oversized one and then discard the parts known not to solve the problem, like in [Stier et al., 2018], in which they use a cooperative game to prune the useless nodes of the hidden layer.

Inside the hyper-parameter search field, a learning rate schedule is also a method to avoid stagnation at local minima and erratic and oscillating training behaviors. In [Park et al., 2020], the researchers aim to introduce cost functions into learning rate optimization methods that increase or decrease the learning rate value taking into account the training behavior.

However, since we believe that the hyperparameter search is already well covered by the sweep training configuration that will be discussed later in this section, the most relevant updates on 3D-Deep focus on the architecture.

A restriction of the previous architecture gave the most straightforward idea for architecture optimization: both context branches were a mirror image of each other. It seems logical to think that the complexity of an RGB image does not have to be the same as the complexity of a single-channel image, especially if the latter is a disparity map. This hypothesis allows us to deduce that perhaps both branches do not need the same

architecture.

On the other hand, limiting that context branches can only be ResNet models seems to be a relatively high restriction. Therefore, in this new architecture, the possibility of using ResNext models as context branches has also been introduced. Maybe this variation does not seem very relevant since it only requires to be consequent with the output feature sizes. However, the idea behind this is to verify model independence, letting that more architectures that have previously obtained good results as encoders can be used, permitting in the future to use new state-of-the-art models to extract contextual information.

One of the most effortless updates to test is given by the presentation of the paper [Brock et al., 2021]in 2021 of the Normalize-Free Networks architectures. It explains the improvements that have been achieved by eliminating batch normalization and replacing its function by creating new residual blocks, Scaled Weight Standardization, and Adaptive Gradient Clipping. The upgrade to test the improvements has been straightforward since these refinements can use ResNet as the base model, as in the proposed 3D-Deep model. Also, there is a python library named *nfnets-pytorch* where predefined methods are available to implement these improvements in already loaded models.

The in-depth design analysis gives another more direct update of the 3D-Deep architecture. As mentioned above, the context branches have a CBAM-based attention module that tries to guide the training by extracting high-level features through per-channel reweighting. If we look at Figure 3.11, belonging to the CBAM paper [Woo et al., 2018], there is also a spatial attention module, so it seems pretty logical to use it in the proposed architecture's spatial path and verify if there is any sign of improvement. This module operation is the same as the original paper and is given by the following points.

1. Application of max-pooling and average pooling operations across the channel axis.
2. Concatenation of both feature descriptors.
3. Convolution operation on the concatenated descriptor to obtain a spatial attention map that encodes which parts of the information need to be emphasized and which parts are dispensable.

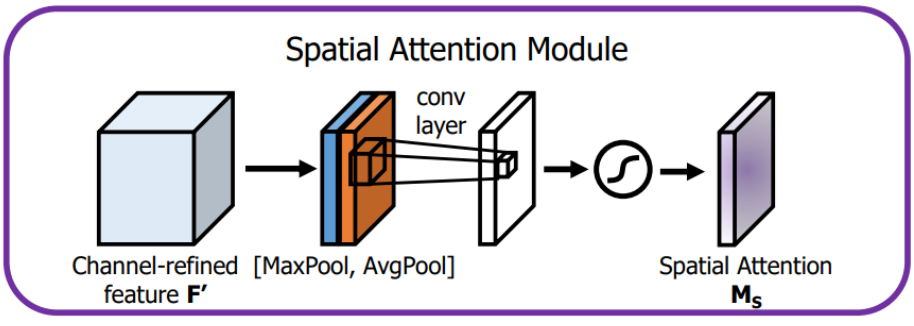


Figure 3.11: Spatial Attention Module.[2]

[2]The image is taken from the paper [Woo et al., 2018]

3.2.3.3.1 Graph-based filter explainability

As seen at the beginning of section 3.2.3.3, there are different approaches to optimizing a neural network depending on how one wants to face the problem.

The network architecture that has been proposed throughout this chapter is focused on semantic segmentation in general and road segmentation in particular. Therefore, in the following section, it is proposed, as a proof of concept, an optimization methodology that tries to enhance the network architecture so that it works better and is more efficient for a specific task, in the particular case at hand, road segmentation.

Initially, in order to try to understand better how the network architecture worked on the data, the NVIDIA Feature Map Explorer tool was used. This tool is capable of displaying the feature maps returned by a convolutional layer as an image along with detailed numerical information about them. Therefore, it is also possible to visually and numerically analyze each channel individually to understand each convolutional layer's performance. Figure 3.12, for example, shows the feature maps belonging to channels 1, 11, 13, and 47 of the first convolutional block of the context path. This tool's initial analysis of the filters provided a relatively quick conclusion. The first convolutional block of the three branches did not use all of its filters. Some of the feature maps appeared visually black, and the statistical values provided by the platform, minimum, maximum, mean, and standard deviation, all appeared with a value of 0.

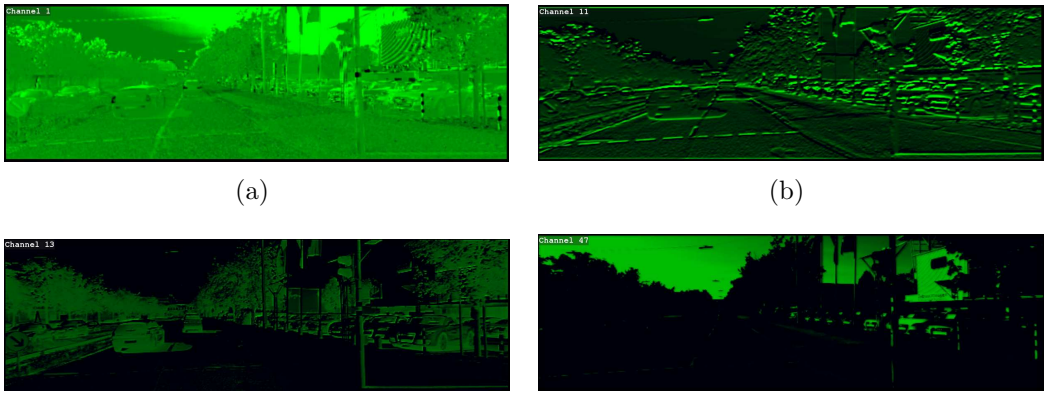


Figure 3.12: Feature maps number 1, 11, 13, and 47 belonging to the first convolutional layer of the same image.

These visual results are pretty helpful, but they did not answer a fundamental question after viewing them: What contribution to the final solution did each active filter make? Some of the visualized filters looked very similar, which made us wonder about the genuine contribution of the two as a whole. To understand the amount of contribution, it is necessary to know how similar the results of the filters are for the same image. With that information, it can be estimated how many of the filters that have an active output contribute to the resolution of the problem. This need is the reason why the following methodology for comparing filters through graphs has been developed.

The initial idea is to obtain the feature maps of a convolutional block, i.e., the 64 channels returned after the first convolutional block of the context path, and compare each one with the rest of them. In order to perform this comparison as efficiently as possible, it has been done through sparse matrix operations, which also can be representations of weighted graphs. The steps to follow are as follows:

A random image is selected from each epoch's batch from which the feature maps to be analyzed are obtained. This selection type is proposed to avoid bias and give us a three-dimensional array for each photo (C, H, W). This three-dimensional array is reshaped to a two-dimensional matrix with dimensions C and $H * W$ and then stored as a sparse matrix.

The cosine similarity is used to compare the vectors of this new matrix. This particular distance is the one selected, but any other vector distance function can be used as desired, such as the Euclidean distance. This function returns the similarities in the form of a sparse square matrix with the shape $(C\ x\ C)$, where v_{ij} is the similarity value between vector i and vector j. This last operation returns a sparse matrix in which the non-zero values represent the higher than the preset threshold similarities between feature maps.

As mentioned before, sparse matrices can also be representations of graphs so that the existence of the value v_{ij} represents the weight of an edge between nodes i and j. This representation offers an advantage, and that is that by calculating the connected components of a graph, it is possible to know the number of filters whose similarity does not exceed the predefined threshold because those that exceed it will be represented as a connected sub-graph.

Therefore, it is possible to calculate the number of feature maps whose similarity does not exceed a certain threshold for each batch to obtain the average number of feature maps used for the whole dataset and the maximum value of the feature maps. These values give us an approximate idea of the utility value of each of the convolutional layers of the network.

3.2.3.3.2 Architecture update

The analysis of the architecture through the methodology described in the previous point, plus the visual and numerical information obtained through the Feature Map Explorer application, led us to certain conclusions that triggered the modification of the initial 3D-Deep architecture.

The first conclusion that the analysis already exposed above could be reached is that the first convolutional layer of both context branches did not use many of its kernels. However, an analysis of the second convolutional layer, in which there were no maps with zero value or with similarities higher than a threshold of 95%, allowed us to deduce that the information of the feature maps of the first convolutional block was sufficient for the second one to work at 100%.

Besides, with the methodology described in the previous point, it is possible to compare filters from different branches, something done with each of the convolutional blocks of the context and the spatial branches. This analysis gave us some exciting conclusions. The deepest convolutional blocks of each branch did not share similarities. Regardless, the first convolutional block had different behavior discussed below.

The first convolutional block of the context branch did not share similarities with the first convolutional block of the three-dimensional context branch. However, the feature maps of each of the context branches' first convolutional block showed many similarities with the first convolutional block of the spatial branch. That analysis allowed us to deduce what was assumed at the beginning of the creation of the architecture, that the information extracted from each data source is different. Moreover, the similarity between spatial and context paths seems logical since the spatial branch works with both data sources, and also, the first convolutional layers usually tend to focus on the most superficial data features such as edges. It can be concluded that this similarity led to think that the first convolutional layer of the spatial branch consisted of a particular "fusion" of the characteristics of the first convolutional blocks of the other branches.

Given the excellent performance that the Feature Fusion Module has in the 3D-Deep architecture, it seems a logical hypothesis to think that replacing the first convolutional layer of the spatial path with a Feature fusion module that has as input the feature maps returned by the first convolutional blocks of both context paths, it will be possible to obtain better results.

After that modification, the architecture was reanalyzed visually and by employing the graph method. The results were enlightening. The initial feature fusion module produces an output of 64 feature maps, the same as the first convolutional block of the original spatial path. However, none of them has similarities with the others, and none of them has 0 output which is considered a success.

After all the modifications made to 3D-Deep, the final architecture network has been rebranded as 3D-Deepest, and its complete structure is shown in 3.13.

3.3 Experimental Analysis

In the following points, the results obtained during the research process will be presented, both with the 3D-Deep architecture published in the paper [Hernández et al., 2020] and 3D-Deepest architecture. In addition, an analysis of the datasets used for training will be performed.

3.3.1 Experimental set-up

Different methodologies have been carried out in the battery of tests described in the following points. The tests performed with the 3D-Deepest architecture were carried out with an NVIDIA TITAN RTX graphics card. The tests performed with the 3D-Deepest architecture have been carried out using the Wandb [Biewald, 2020] tool, which allows an automatic search among the configurable parameters through a *sweep* procedure.

The cross-entropy loss function was selected during all the training performed with the 3D-Deep architecture. Usually, the loss function is calculated from the final results of the architecture with respect to the labels. However, following the methodology exposed in [Yu et al., 2018], the loss function is given from the weighting of several auxiliary error values besides the loss of the whole architecture. The supervision module is the one in charge of providing these auxiliary values. As can be seen in Equation 3.6a, l_p represents

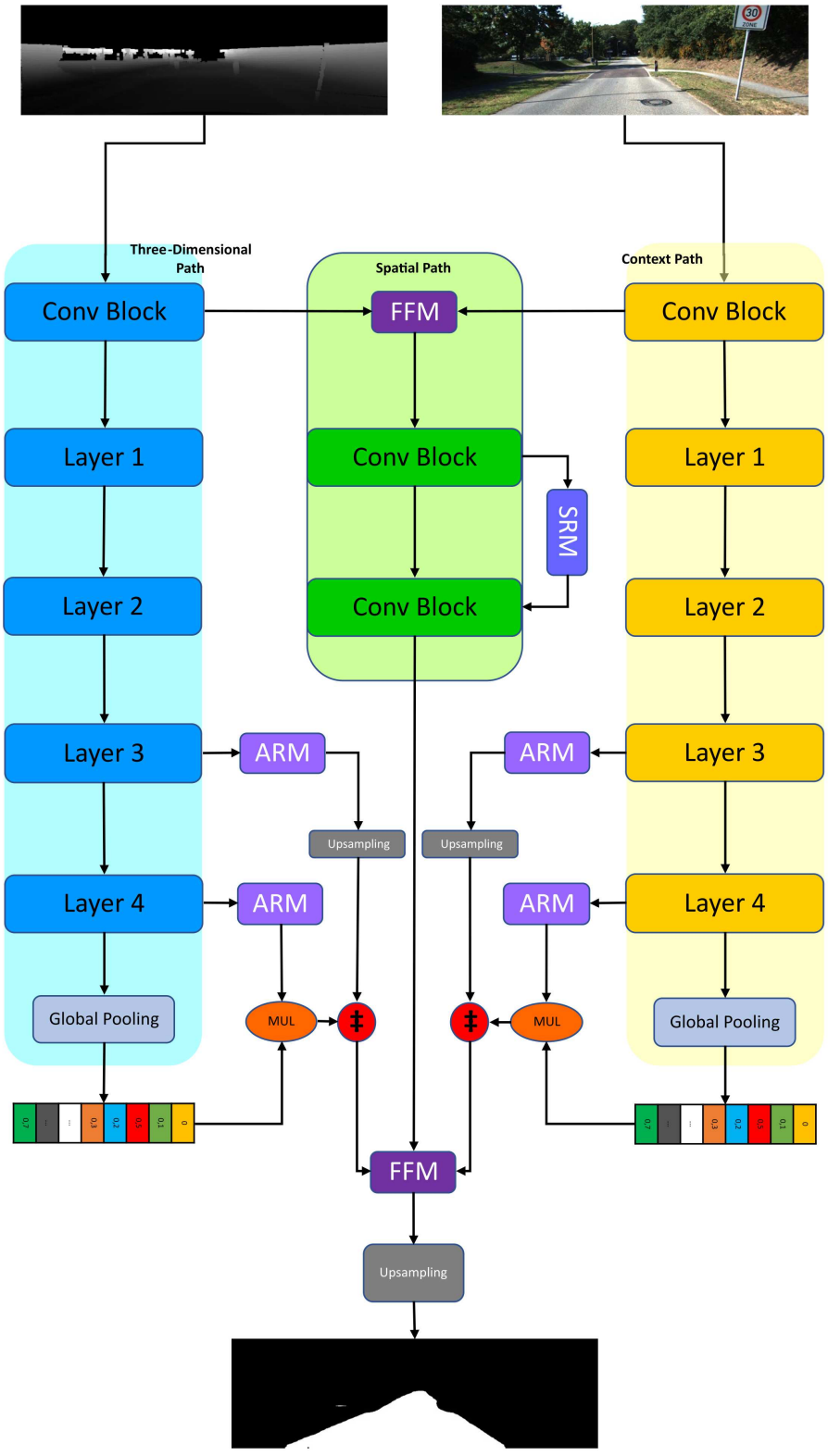


Figure 3.13: 3D-DEEPEST network architecture.

the cross-entropy loss of the whole model, and α and β represent the weighting given to the loss values of the context path and the depth context path, respectively.

$$loss = l_p + \sum_{i=1}^{n} \alpha * L(ARM_i) + \sum_{j=1}^{n} \beta * L(ARM_j) \tag{3.6a}$$

$$L(ARM_i) = \frac{1}{N} \sum_{n} -\log\left(\frac{\exp(p_i)}{\sum_j \exp(p_j)}\right) \tag{3.6b}$$

This module consists of a convolutional layer and an upsampling step that, only during the training process, transforms the outputs of each of the Attention Refinement Module into a feature map equivalent to the network's final output so that it can also be compared with the labels. This loss calculation can be seen in Equation 3.6b, representing the auxiliary cross-entropy loss obtained from the Attention Refinement module i. The role of auxiliary loss functions produced by the supervision layers is to control the output of the context branches independently from the overall architecture, similar to discussed in [Xie and Tu, 2015], and train the attention modules independently.

For the loss functions of the 3D-Deepest architecture, in addition to the Cross-Entropy already used in the 3D-Deep architecture, the *Focal Loss (FL)* function has been added, [Lin et al., 2017b]. This option was selected since it is considered to be able to obtain outstanding segmentation results. The use of one loss function or another has been random since it was configured as one of the parameters used in the training *sweep*.

The calculation of the losses of the model using focal loss does not involve a substantial variation with respect to its implementation. The loss function remains the same as in equation 3.6a but substituting all the terms where the value is calculated using cross-entropy by focal loss, Equation 3.7.

$$loss(p_t) = (1 - p_t)^{\gamma} \log(p_t) \tag{3.7}$$

3.3.1.1 Dataset

The datasets used to carry out the test battery are Cityscapes [Cordts et al., 2016] and KITTI in its road segmentation variant [Fritsch et al., 2013]. These two have been selected precisely because each has a different source of three-dimensional information. In Cityscapes, the three-dimensional data is provided by the disparity maps obtained from stereo vision. However, in the case of KITTI, the three-dimensional input is provided by point clouds obtained from LiDAR.

For training with the Cityscapes dataset under the 3D-Deep architecture, 2975 images from the left camera and their respective disparity maps were used. These images belong to the set of images whose ground truth is finely annotated from the semantic segmentation benchmark. With the idea of improving the disparity maps, the technique proposed in [Min et al., 2014] is applied to complete the information available in the original maps.

For the training with KITTI, the 289 road images provided in their semantic segmentation road dataset and their respective point clouds were used. Since no validation images are provided, Monte Carlo cross-validation [Xu and Liang, 2001] was used during

the training as a method of validating the results. Thirty images with their respective point clouds are randomly separated in each iteration and used as a validation set. The number of iterations considered for each training is four. This number will be increased to 10, with the idea of generating much more stable results, once all the hyper-parameters that previously gave the best results have been decided.

In both datasets, Kaiming initialization is used for those layers that, like spatial path, do not use Imagenet's pre-trained weights.

3.3.1.2 Data Augmentation

Since the size of the KITTI dataset is relatively small, it is necessary to implement techniques that prevent the proposed trainings from overlearning, something that can happen in such cases. That is why data augmentation techniques have been implemented in the system to mitigate this possible disadvantage of the KITTI dataset. In addition, these techniques have also been extended to the Cityscapes dataset based on the hypothesis that will improve the training results.

As is widely known in deep learning research, this type of methodology increases the robustness and generalization capacity of the system in the face of possible variations when working with data outside the training set. It consists of making minor modifications to the training data without altering its nature to generate new data. In the case of images, two types of transformations may be performed:

- Geometric transformations, such as affine transformations, perspective changes, cropping and scaling.
- Variations in pixel values, such as the addition of noise or shifts in color spaces.

The transformations focused on pixel value changes used in the process were of two types. Those based on noise include Poisson noise, salt and pepper noise, speck noise, and Gaussian blur. Those based on color space changes include two different types of modifications: color casting and color jittering, [Wu et al., 2015, Wei, 2015]. On the side of geometric modifications, the ones applied are perspective, distortion, cropping, mirroring, and related random transformations.

All these operations are performed at runtime, which makes the size of the dataset virtually infinite with only a slight trade-off in computation time.

3.3.2 Experimental results

All the training process described below has been performed with the data augmentation techniques explained in Section 3.3.1.2.

3.3.2.1 Cityscapes

The first training set was done with the ResNet18 network as the backbone for the two context paths, using a stochastic gradient descent optimizer and with $5e^{-3}$ as the learning rate. As shown in Figure 3.14a, the results were not as good as expected as from epoch

20, the mIoU does not improve, and its maximum value reached is close to 50%. The change of optimization method to ADAM did not get the desired results either, as seen in the same graph, even slightly varying the learning rate. In both runs, the maximum values remained around 45%.

The optimization algorithm that worked best in this first attempt was ASGD [Polyak and Juditsky, 1992], which showed a more stable trend and reached maximum values for mIoU of 56%. These are still not as good as desired, but they were considered a good starting point, so all the successive training executions in 3D-Deep architecture were done with an ASGD optimizer.

Once the optimization algorithm was decided, the following training sessions were oriented towards choosing the optimal hyperparameters and backbone for the context and three-dimensional paths. Due to the GPU memory limitation caused by the high resolution of the cityscapes images, a heavy learning strategy was implemented [Shelhamer et al., 2017].

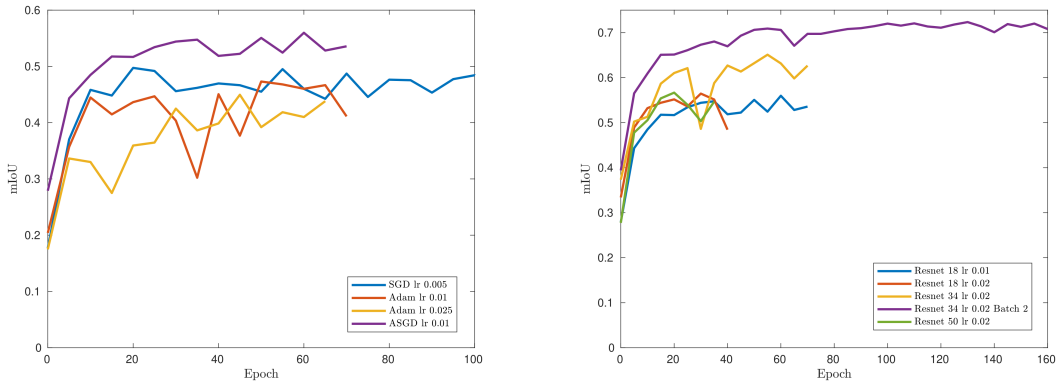
As shown in Figure 3.14b, the learning rate remained close to the values used in the initial trainings. However, in some cases, the learning rate has been increased by 0.01. This increase is done since it is considered that the depth context path requires a more *severe* modification of its layer weights due to the fact that they are not pre-trained, as are those of the context path. The results obtained confirmed that the optimum learning rate was around 0.02 and that the ideal backbone was ResNet34. Increasing the network depth, for example, with the ResNet50 architecture, clearly worsened the results. It is hypothesized that due to size limitations during ResNet50 training, it was necessary to reduce the size of the images by half, which probably also affected the accuracy results.

Finally, two minor improvements were applied to the initial training that improved the results, as seen in the purple graph in Figure 3.14b. First, using the pre-trained weights of Imagenet for the three-dimensional branch. This modification might seem somewhat counterproductive considering that the input images of an Imagenet-trained architecture (RGB) are very different from the input images of the depth context path (three-dimensional data) but surprisingly improved the results. Second, the use of NVIDIA's Apex library [NVIDIA, 2019] enabled a significant reduction in system memory consumption, allowing a minibatch size of two to be used. This minibatch increase ruled out the use of heavy-learning techniques, but as can be seen, it substantially enhanced the results, increasing the mIoU to a maximum of 72.34%.

Although the results were acceptable, especially since the mIoU is a very restrictive metric, they were not considered good enough to be ranked in the Cityscapes benchmark. The results in test data are usually worse than the validation ones, so more work was put on the system until a way to improve the results was found.

3.3.2.2 KITTI

The training with KITTI images with the 3D-Deep architecture was less restrictive in memory consumption thanks to their smaller size than Cityscapes. The absence of the previous limitations made it possible to test all ResNet architectures and increase the size of the minibatch to four. Furthermore, as the trainings were conducted after the ones using the Cityscapes dataset, everything discovered previously was adapted to make the



(a) Evolution of mIoU on training for different optimizers.

(b) Evolution of mIoU on training for different learning rates and context paths.

Figure 3.14: Evolution of training according to the backbone model, the batch size, learning rate, and optimizer.

process much more efficient and quicker in searching for optimal results. Therefore, from now on, all the trainings are considered to be done with the ASGD optimization method. In addition, both context branches will have their weights initialized with the Imagenet pre-trained weights.

The first training with images belonging to KITTI's dataset, taken as a starting point, already obtained excellent results: the minimum F1 value of 96.16% for the ResNet18 architecture and the maximum F1 value of 96.80% for the ResNet152 architecture. Due to the good results reported, it was determined to use a learning rate scheduler to improve the performance, specifically, a cyclical and decreasing cyclical learning rate scheduler with triangular functions, [Smith, 2017]. The upper and lower limits for the scheduler were previously calculated using the "learning rate range test" methodology also proposed in the paper, with the selected values being 0.25 for the upper limit and 0.0001 for the lower limit. However, this scheduler did not lead to any results improvement, in some cases, even worsening them. Nevertheless, the limits of the learning rate obtained in these tests provided us with helpful information for the subsequent development steps.

To continue with the idea that establishing a scheduling policy would help the training convergence, it was decided to use a decreasing polynomial scheduler. Taking advantage of the previously calculated upper limit, this and its successive three divisions by two (0.125, 0.0625, 0.03125) are used as the initial value of the learning rate so that it would come close to the previously obtained lower limit in the last training stages. This idea substantially improved the results enough to consider ranking them in the KITTI benchmark, as can be seen in Table 3.1.

Appendix A shows the complete tables of all the training performed up to the time of ranking the work in the KITTI benchmark, which is not shown here for the sake of simplicity.

Ranking in KITTI posed a problem since the results must be delivered in *Bird's eye view (BEV)* confidence maps, and all the training had been done in panoramic view.

Table 3.1: Training results in the KITTI dataset with polynomial scheduler for the learning rate.

Backbone	Initial lr	F1 value	Max F1
ResNet152	0.25	96.87%	97.3
ResNet50	0.125	**97.37%**	97.66
ResNet34	0.0625	97.33%	**97.77**
ResNet152	0.03125	97.11%	97.35

Initially, the probability map returned by the network was transformed to a BEV perspective, but the results were not good. Therefore it was decided to transform the whole dataset to BEV (using the code provided with the dataset) and train directly in that perspective. The training sesions were equal to the perspective ones and obtained outstanding results, with maximum values of around 98% for the F1 value.

As the last step, a training session was carried out, but with ten iterations in the Monte Carlo Cross-Validation technique. The idea behind it is to obtain an "average" value that could be as close as possible to what would be obtained in the test images. This final step was accomplished using the best configuration achieved at the moment: ResNet101 as the backbone and a learning rate of 0.03125. This training gave an average value for the F1 error of 97.09% and maximum values close to 98% between all the iterations. The results were good enough to rank the model, so the test images were passed through the network to deliver the probability maps. The eighth position was obtained at the time of submission, tied with the seventh, as shown in Table 3.2.

Table 3.2: KITTI dataset Evaluation Results in test images. (percentage)

Benchmark	MaxF	AP	PRE	REC	FPR	FNR
UM	95.35	93.50	95.20	95.51	2.20	4.49
UMM	97.27	95.76	97.01	97.54	3.31	2.46
UU	94.67	93.04	94.23	95.12	1.90	4.88
URBAN	96.02	94.00	95.68	96.35	2.39	3.65

Going into more detail, Table 3.3 shows that our network is 0.12 above the second in AP within the UMM_ROAD category. This category, in which the network proposed in this article has its best results, includes urban roads with multiple marked lanes. Within the general category URBAN_ROAD, which includes the categories UU (Urban Unmarked), UM (Urban Marked), UMM (Urban Multiple Marked), the network is only 0.04 of the first position in the average accuracy. It has been considered an excellent result (see Table 3.4).

3D-DEEP also ranks first in AP and third in MaxF, with more than acceptable results for implementation in a real-time system, considering the fastest methods classified (see Table 3.5).

Table 3.3: Comparison of KITTI evaluation results in UMM. (percentage)

Method	MaxF	AP	PRE	REC
3D-DEEP (Ours)	97.27	**95.76**	97.01	97.54
PILARD [Chen et al., 2019]	97.77	95.64	97.75	97.79
LidCamNet [Caltagirone et al., 2019]	97.08	95.51	97.28	96.88

Table 3.4: Comparison of KITTI evaluation results in Urban (UM+UMM+UU). (percentage)

Method	MaxF	AP	PRE	REC
PILARD [Chen et al., 2019]	97.03	**94.03**	97.19	96.88
3D-DEEP (Ours)	96.02	94.00	95.68	96.35
LidCamNet [Caltagirone et al., 2019]	96.03	93.93	96.23	95.83

Table 3.5: Comparison of evaluation results in KITTI for the fastest architectures. (percentage)

Method	MaxF	AP	PRE	REC	Runtime
NF2CNN	**96.70**	89.93	95.37	98.07	0.006 s
ChipNet [Lyu et al., 2018]	94.05	88.29	93.57	94.53	0.012 s
LoDNN [Caltagirone et al., 2017]	94.07	92.03	92.81	95.37	0.018 s
multi-task CNN [Oeljeklaus et al., 2018]	86.81	82.15	78.26	97.47	0.0251 s
ALO-AVG-MM [Reis et al., 2019]	92.03	85.64	90.65	93.45	0.0296 s
LCFNet	96.42	91.05	96.60	96.24	0.03 s
3D-DEEP (Ours)	96.02	**94.00**	95.68	96.35	0.03 s

Quantitatively the results are also excellent for each category evaluated, as shown in Figure 3.15. The network detects the majority of the labeled road in the test images (pixels labeled green) with hardly any false positives (pixels labeled blue) or false negatives (pixels labeled red). Together with the corresponding quantitative ones, these qualitative results allow 3D-DEEP to consider an optimal architecture for an autonomous driving system implementation.

The results in the KITTI dataset with the 3D-Deepest architecture are not as detailed as those with the 3D-Deep architecture. This shortage of detail is because the architecture update is only a proof of concept for a possible validation of the proposed optimizations. That is why 3D-Deepest also is not ranked in the KITTI road segmentation benchmark.

A *sweep* has been defined in which the different proposed optimizations are introduced as hyper-parameters in order to explore the best configuration extensively. In addition, this methodology allows us to group the training by parameter, which lets us obtain

(a) Urban unmarked road.

(b) Urban multiple marked road.

(c) Urban marked road.

Figure 3.15: Results for KITTI dataset in test images.

averages of the network performance, giving us an approximate understanding of each of the updates separately. As in the 3D-Deep architecture, the training has been executed using the Montecarlo cross-validation methodology with a k-fold equal to ten.

If we only consider the updates involving the extension of the context paths architectures and the initial feature fusion module proposed in Section 3.2.3.3, the results are interesting. As shown in Table 3.6, the results have surpassed the maximum value of the F1-score; yet, the average value has decreased, although it has remained close. However, it is worth mentioning that, as shown in Table 3.1, with the previous architecture, the mean and maximum values did not have to coincide in the same configuration of architectures for the context paths, which has happened with the updates in 3D-Deepest.

Table 3.6: 3D-Deepest best results by optimization.

Optimization Type	**Max F1**	**Average F1**
3D-Deep	97.77	97.33
Graph Optimization	98.02	97.17
NFNET	97.67	97.31
Spatial Attention	**98.12**	**97.36**

The best values are shown among all the trainings performed with the exposed techniques. Note that the values of the 3D-Deep architecture were obtained using a Kfold=4.

Focusing on the Nfnet optimization, if we look at the best training values, it can be seen that the maximum F1-score value is very similar to that obtained with the initial 3D-Deep architecture, although the average value slightly drops (see Table 3.6).

If an average of all trainings grouped by this inclusion is made, the results seem to be consistent with the previous results, as can be seen in Table 3.7. The average values are slightly better if the Nfnet optimizations are included, but the average of the maximum values is slightly higher if they are not included. It should be mentioned that no training has been excluded for the calculation of the mean values shown. Not even those that could be considered outliers, so the values are significantly lower than those of the best training sessions. As in the rest of the averaged results, this proceeding has been done because the main idea was not to seek the best results but rather to perform a proof of concept that would allow us to know if the research was on the right track.

Table 3.7: NFNET optimization impact.

	Max F1	**Average F1**
NFNET=True	94.30	**91.94**
NFNET=False	**95.25**	91.14

The F1 values shown are the mean values calculated from all training runs including and not including NFNET optimization.

The inclusion of the spatial attention module update also provides engaging information. When analyzing the values obtained for the best training, shown in Table 3.6, they are similar to those obtained without this update. However, if the average values of the whole *sweep* in which this update was included are examined (Table 3.8), the values are markedly higher, which indicates that if it does not provide a significant improvement in the maximum accuracy, it does in the stability of the training processes.

Table 3.8: Spatial Attention optimization impact.

	Max F1	**Average F1**
Spatial=True	**95.91**	**92.35**
Spatial=False	94.78	91.54

The F1 values shown are the mean values calculated from all training runs including and not including Spatial Attention optimization.

In order to avoid bias in comparison, the entire testing methodology performed with the panoramic view images has been repeated with the BEV images, like was done with the 3D-Deep Architecture.

Focusing, as with the panoramic images, only on the updates obtained by analyzing the filters, the results are fascinating. Each of the different optimizations offers very similar results in the maximum value of F1 reached by the training run and slightly higher values in the average of the whole set of K-folds. It is also worth noting that this time the comparison is made with values with K equals 10, which is a more accurate comparison.

Table 3.9: 3D-Deepest best results by optimization. (BEV)

	Max F1	**Average F1**
3D-Deep	**97.85**	97.09
Graph Optimization	97.83	**97.22**
NFNET	97.83	97.10
Spatial Attention	97.77	97.12

The best values are shown among all the trainings done with Birds Eye View perspective performed with the exposed techniques.

If we target Table 3.10, in which the average values of all the training sessions are delivered, some quick conclusions can also be drawn. It can be seen that, unlike what happened in the panoramic images, the use of NFNET optimizations, in this case, has not meant an improvement compared to not using them, although taking into account the best of the training sessions, it has. This contradiction concerning Table 3.9 may indicate both that the contribution of the architectures without batch normalization in this problem is not significant and that since this is a proof of concept, a much more detailed exploration of the solution space is necessary.

Table 3.10: NFNET optimization impact. (BEV)

	Max F1	**Average F1**
NFNET=True	95.80	94.35
NFNET=False	**96.53**	**95.02**

The F1 values shown are the mean values calculated from all training runs including and not including NFNET optimization using Bids Eye View perspective.

The spatial attention module results are surprisingly contradictory to the results obtained with perspective images. The comparative results in Table 3.11 show that the network's performance with spatial attention is slightly inferior to that using bird's eye view images. Furthermore, in Table 3.9, which shows the best training results for each enhancement, training with spatial attention is not a significant improvement, remaining on par with the other enhancements. It is possible that this is due to the fact that spatial information is much more critical in panoramic perspective images than in bird's eye view images since the latter has a considerable distortion in the background.

In general, we are pleased with the performance achieved with the different methodologies proposed in this section. The results seem to be encouraging and suggest that

Table 3.11: Spatial Attention optimization impact. (BEV)

	Max F1	**Average F1**
Spatial=True	95.90	93.97
Spatial=False	**96.16**	**94.68**

The F1 values shown are the mean values calculated from all training runs including and not including Spatial Attention optimization using Bids Eye View perspective.

this research can be continued in the future to obtain a complete and functional system for an autonomous vehicle.

Chapter 4

Intersection Classification

As seen throughout Chapter 1, estimating the scene in front of a vehicle is crucial for safe autonomous vehicle maneuvers, and it is also key to *Advanced Driver Assistance Systems (ADAS)*. Once the estimation of the drivable surface has been solved, the next step of this investigation leads to intersections.

Indeed, intersection areas are one of the most critical for both drivers and pedestrians. It is where vehicle's paths might cross at some point, and it is also where pedestrians are most likely to cross the roadway.

Consequently, within the navigation system of an autonomous vehicle, it can be considered of paramount concern to have a reliable system able to identify this specific traffic scene. For example, the vehicle control system might notice the number of road entries to the upcoming crossing and therefore estimate from which trajectories can come either other vehicle or pedestrian who are on a collision course with it. Besides the safety field, the information about the crossing area can allow the vehicle to perform safe maneuvers without the need to have a complete and detailed map of the area, as a human being would do when asking for directions in a city s/he does not know.

From a parallel point of view, it is also believed that the detection and classification of intersections can be exploited for multiple purposes, such as an input to high-level classifiers of other drivers' maneuvers. The achievement of this hypothesis might ease the prediction of position and intentions of *Vulnerable Road Users (VRU)* or plan how to drive centered and safely without road markings.

4.1 Introduction

In computer vision, classification is the task of assigning a label to an image based on what appears in it. If, in addition, the system wants to locate the object within the same image, it would be talking about the field of detection. In the latter case, this does not have to be limited to a single target per image but can detect more than one and of different classes.

As seen in Chapter 2, classification in deep learning is usually performed by convolutional neural networks. As briefly introduced in Chapter 3, this type of network usually has an encoder-classifier structure (Figure 4.1).

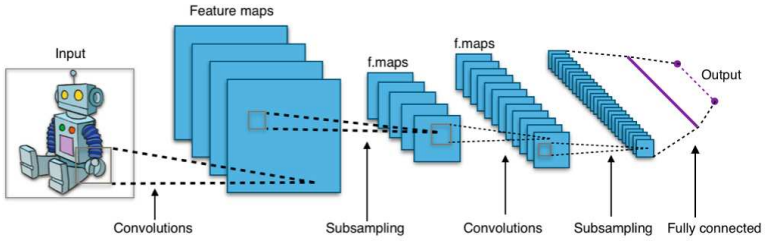


Figure 4.1: Encoder-classifier example.[1]

The encoder performs the same job as explained earlier in Section 3.1 in detail, reducing the image to an n-dimensional feature vector. However, unlike FCNN, features are not expanded again by the decoder but are transformed into class probabilities or confidence values.

This transformation is carried out by the classifier that is the part marked as *Fully connected* in the Figure 4.1.

The classifier can be of multiple types. Usually, it consists of a linear or fully connected layer, as in ResNet architectures, which transform the features into a confidence vector of the number of classes through trainable weights and biases, eq(4.1). Another option, as in VGG architectures, is to use several fully connected layers sequentially for the same purpose. In this case, the first layers usually reduce the size of the feature vector, e.g., in VGG16 from a size of 25088 to 4096, and the last layer is in charge of transforming the last feature vector into a confidence vector. Then, if one wants to work with probabilities, it will only be necessary to apply the softmax function to the output confidence vector.

$$h_i^{in} = h_{i-1}^{out} * W_i + B_i \tag{4.1}$$

There are also options outside the fully connected layer, but in general, they cannot be trained together with the encoder, as is the case with a linear layer classifier. An excellent example of it is SVMs or the *k-nearest neighbors (K-NN)* algorithm. Both algorithms are used to classify the neural network feature vectors after being trained independently of the neural network.

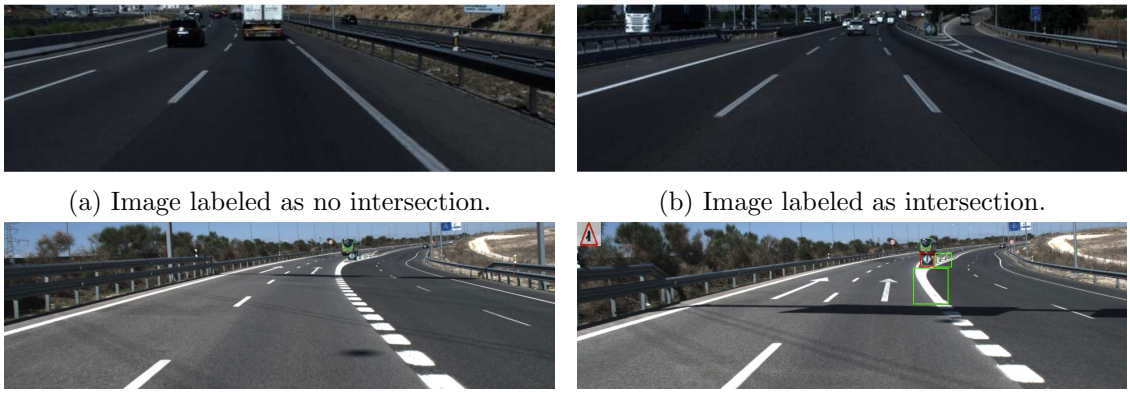
4.2 Method

Considering how classification usually works, the following section and its corresponding subsections will explain the methodology followed during the research for the classification of intersections and the different techniques tested to compare the results and find the one that best suits the problem.

4.2.1 Early works

Early work on intersection classification was conducted in conjunction with the road intersection segmentation work discussed in Section 3.2.2.

[1] Image obtained by creative commons license. [Commons, 2015]



(a) Image labeled as no intersection.

(b) Image labeled as intersection.

(c) Image detected as no intersection.

(d) Image detected as intersection.

Figure 4.2: Example of images belonging to the dataset recorded on the M-40 highway with which the classification network has been trained.

One possible approach to improving the intersection segmentation system consisted of a divide-and-conquer strategy. If one can train a network that is able to detect whether an intersection exists in an image of a highway, then the task can be divided into the next steps. For images where there is no intersection, a network is already in place that gives good segmentation results and which at the time was intended to be improved (see ection 3.2.1). On the other hand, for images where there is an intersection, it seems a simple assumption that a network that segments the main road and the detour or incorporation will do the work much better if only trained with intersection images.

The training of this classification model, and since it is also considered a proof of concept, do not pose great difficulty. The dataset used is the same as for segmentation in Section 3.2.2, labeled in sequences of frames in which there is or is no intersection (See Figures 4.2a and 4.2b). The networks to be trained are also the same; however, they are not FCNN networks but the original architectures proposed in paper [He et al., 2016] substituting the last 1000-class fully connected layer by a bi-class one.

The results obtained at first sight were quite promising, reaching in some cases an accuracy of 94%. However, a more detailed analysis of the results revealed some essential flaws in the system. As shown in Figures 4.2c and 4.2d, the intersection's detection in most cases was performed a few meters from the end of the road convergence, which does not seem to be a safe situation for the vehicle if it wanted to take the intersection. In addition, a more detailed analysis of the reason for this led to the conclusion that the network was not learning the intersection geometry and its intrinsic features but was detecting both the previous zebra marks and the presence of the green plastic divider (see bounding boxes in Figure 4.2d. This misdetection was a risk since the system could detect as intersections places where there had been one, and the road markings were still painted, but now there was a concrete wall preventing access to the intersection.

This first approach has made us realize that intersection classification is not a simple research topic and requires more effort to achieve good results. As seen in Chapter 2, there is more research for a consult or draw upon if the focus is on urban intersections.

Therefore, studying the possible procedures that can improve the urban intersections classification in this section seems a sensible aim.

As with highway intersections, the initial approach is through a preliminary study, [Ballardini et al., 2021]. This research initiated evaluations by studying the classification capabilities of two well-established networks for image classification, namely the ResNet [He et al., 2016] and VGG [Simonyan and Zisserman, 2014] networks. For this purpose, intersection images were directly classified as a starting point, and the results obtained were compared with the same architectures trained using a Teacher/student methodology. The results, although preliminary, are promising since, in most of the categories, better accuracy is obtained using the Teacher/student methodology than the direct classification.

Overall, the classification of intersections by deep learning can yield good results, and the benefits that this would bring to autonomous vehicle development are clear. Therefore, the subsequent research step is a more detailed examination of the subject. So, this investigation's purpose consists of assessing different methodologies addressed to detect the geometry of an upcoming intersection inside metropolitan areas. To this end, the classification capabilities of state-of-the-art CNN systems have been evaluated, including networks designed to analyze single-frame images as well as video sequences.

Furthermore, concerning single-frame image analysis, the aim is to confirm the performance gain of the metric learning and teacher/student learning paradigm for a standard classification baseline. Different loss functions have also been investigated, including triple loss techniques. Each of these network configurations and training methods is carefully described in the following sections, exploring the entire research activity. For the sake of simplicity, Table 4.1 and Figure 4.3 summarize all the approaches presented.

Table 4.1: Overall Scheme of Training Approaches.

Learning Scheme	**Key Features**
Baseline	End to End Classification Standard RGB image classification method
Metric Learning	Learning to classify by using distances between embedding vectors
Teacher/Student	Two networks are sequentially trained metric learning can also be applied
LSTM	These networks are well-suited to process time series data
PyTorchVideo	Deep Learning library for video understanding research

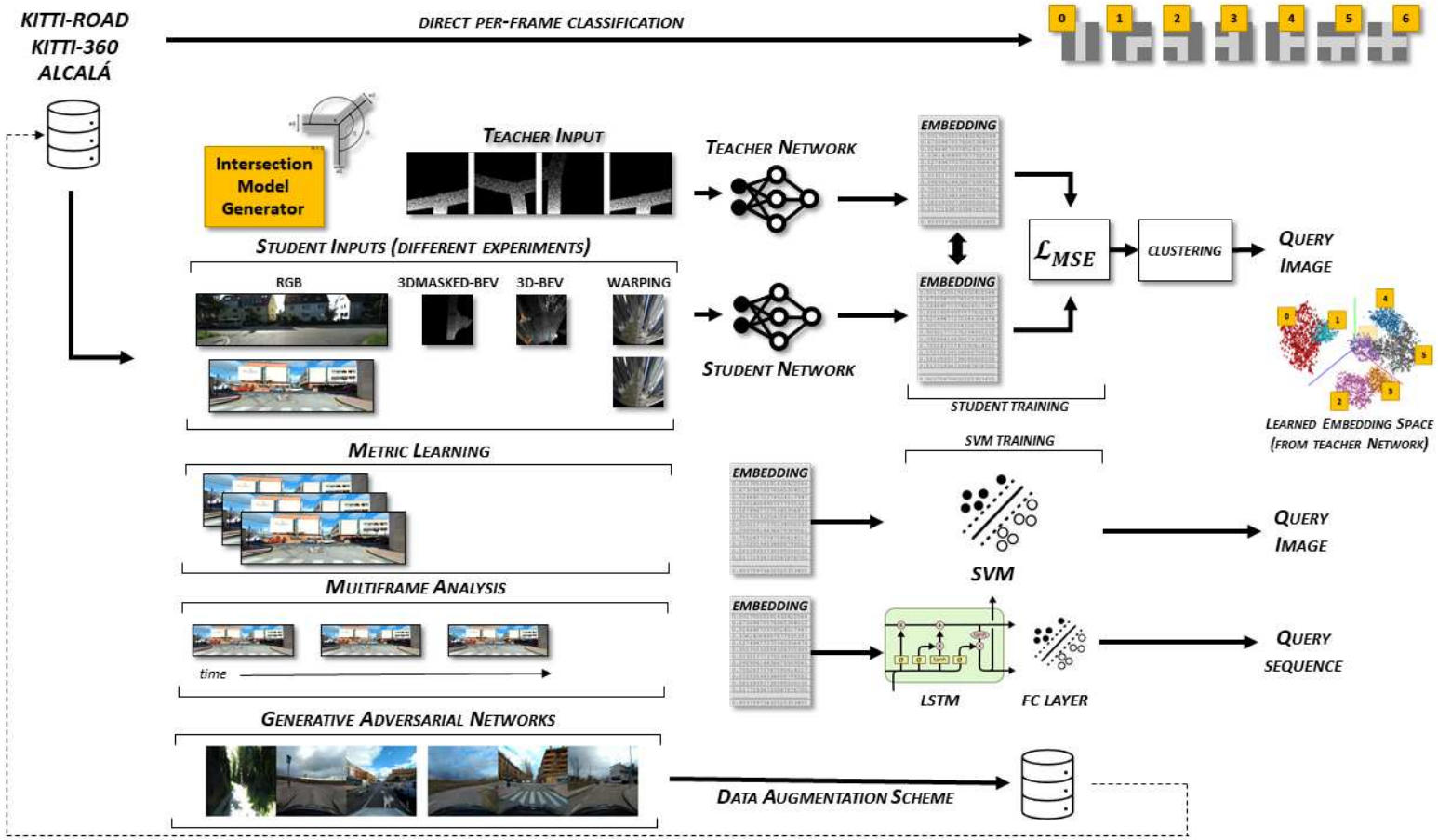


Figure 4.3: The picture includes all the approaches researched in this chapter.

4.2.2 Baseline

As in the preliminary study, it is necessary to establish a baseline against which to compare the proposed training techniques. This baseline will be established by the results obtained by direct classification. For the sake of an unbiased evaluation of the proposed methodologies, more network architectures were incorporated to explore the possibilities of other backbone capabilities. In addition to all common ResNet and VGG implementations,MobileNet v3 [Howard et al., 2019] and Inception v3 [Szegedy et al., 2016] architectures have also been included. Besides representing the more trivial learning paradigm, these end-to-end configurations allowed us to create an architecture baseline to compare systems with each other.

Following the first preliminary work concerning the KITTI and KITTI-360 sequences, the RGB images from the left camera of the stereo rig are the ones used for the direct classification, as will be explained in detail furthermore. For all KITTI image sequences, similar to that proposed by other authors, a second image set was prepared in which a 2D-homography is applied for transformation to the BEV perspective. In addition, this same transformation has been performed with the new images of the Alcalá dataset introduced later in Section 4.2.2.1. The intention in creating BEV images for each of the correspondent RGBs is trying to help the classification network by eliminating cluttering elements of the scene, e.g., above horizon elements, while at the same time emphasizing the intersection geometry.

Together with the RGB images directly acquired from the camera, these sets of images allowed us to perform a first classification assessment of the networks mentioned above and evaluate the improvements described in the following sections.

4.2.2.1 Datasets

It is intended to address the intersection classification challenge by exploiting the most popular datasets available in the autonomous driving field, even though not designed explicitly for intersection detection purposes.

This dataset selection would help compare the previous classification works' results effortlessly. However, the first faced issue was related to the locations where these datasets have been recorded. As can be seen in Figure 4.4, these datasets were collected inside massive metropolitan areas like San Francisco (Pandaset/Lyft5), Beijing, and other cities in China (ApolloScape/Baidu), Boston, and Singapore (nuScenes). The intersection geometry configuration on these Manhattan-like environments is simple and repetitive, and the dominant difference consists only on the traffic conditions. Moreover, it is of extreme difficulty to generalize these kinds of settings to mid-sized cities. The intersection visibility in these conditions is hampered due to several reasons, including the width of the avenue and the related lens' field-of-view, the separation of travel directions that make two-way traffic streets look like one-way ones, and not least the vast amount of vehicles. One possible solution for the identification and classification in these extreme conditions would be, for example, exploiting the detection of traffic lights and the analysis of other car trajectories, such as in work in [Geiger, 2013]. On the other hand, less urbanized areas like the ones in Figure 4.5 can benefit from the capabilities of classification networks.

Figure 4.4: Manhattan-like environments images.

The figure depicts typical intersections in the well-known ApolloScape (a-b), Lyft-5 (c-d), NuScenes (e-f) and Pandaset (g-h) datasets.

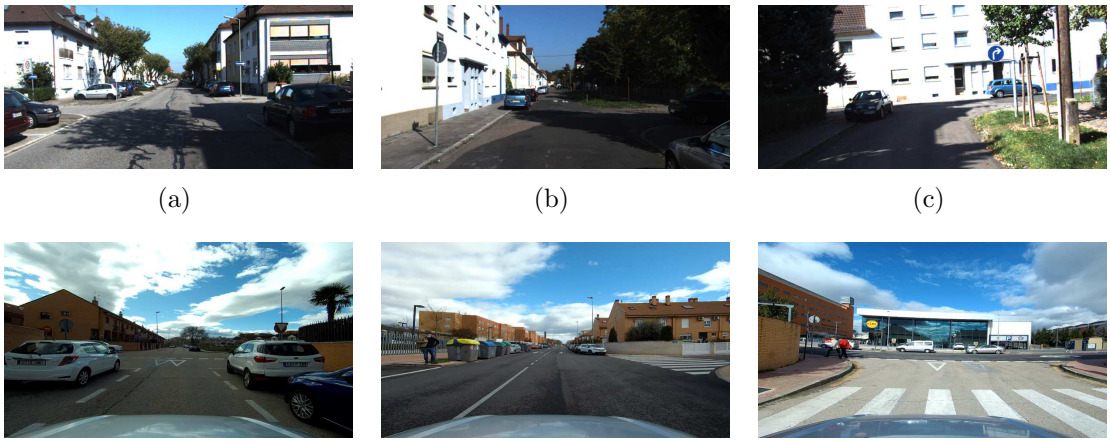


Figure 4.5: KITTI and Alcala dataset image comparision.

Some examples from the mid-size cities of Karlsruhe for the KITTI and KITTI-360 datasets and Alcalá de Henares (our recorded dataset). The reader can appreciate the differences in the spatial distribution of street elements with respect to Figure 4.4, as well as the change in the field-of view between the KITTI cameras (a-c) and our dataset (d-f).

To the best of our knowledge, the only public dataset previously evaluated for the urban intersection classification problem is the KITTI dataset and its recent extension, KITTI-360. There are two issues that, after careful analysis, prevent state-of-the-art works based on image-based detectors from obtaining excellent results. First, as can be seen in Table 4.2, the number of intersection frames is relatively low. These issues make the training phase of any network extremely challenging. Please note that the number of frames is not related to the number of different intersections, as multiple consecutive frames can be associated with the same area.

Table 4.2: Intersections frames per-class, all datasets.

Sequence	0	1	2	3	4	5	6
Alcala-1 (jan26)	4693	1321	1467	2144	4429	2291	5374
Alcala-2 (feb12)	2686	745	446	1893	1973	1287	2948
KITTI-ROAD	✗	308	21	452	52	523	1035
KITTI-360	1190	711	999	1040	706	1048	1129
Total	8569	3085	2933	5529	7160	5149	10486

The number of frames associated to each intersection class, for each of the used datasets.

Besides the frame availability, another interesting number to analyze is the actual number of intersections. As can be noticed in Figure 4.6 and considering only the numbers associated with KITTI-ROAD sequences (a previously selected subset of the original KITTI dataset), the number of per-type intersections is critical, far below the ordinary number of elements typically used for CNNs training's, especially for those networks that use sequences instead of the mere number of frames. It should be noticed that, for a given intersection, the visual appearance of the scene on consecutive frames has minimal changes. Together with the low frame availability, this forced us to pay special attention to the dataset split-phase: randomly choosing frames from the whole dataset was not an option due to the multiple frames associated with every intersection.

By randomly selecting frames, it would have been possible to include almost equal frames of the same intersection into both training and validation or testing, frustrating the separation efforts and leading to biased results. Since there is no easy solution for the availability issue, these considerations led us to create a much more extended dataset of intersections recorded in the surrounding area of Alcalá de Henares, Madrid, Spain, during the first half of 2021. A current subset of the dataset, which contains all intersection geometries in different weather conditions and seasons, is used in this work, and a few images can be appreciated in Figure 4.7. Together with a set of scripts and additional descriptions, all the images will be released to the community[2].

Among the datasets used in this work, one substantial difference lies in the availability of *single* vs. *stereo* camera head as well as *camera* (single-sensor) vs. *camera and LiDAR* (multi-sensor) configurations. This difference allowed or denied some of the configurations explained in Section 4.2.3.1.3.

[2] https://invett.aut.uah.es/intersectiondataset

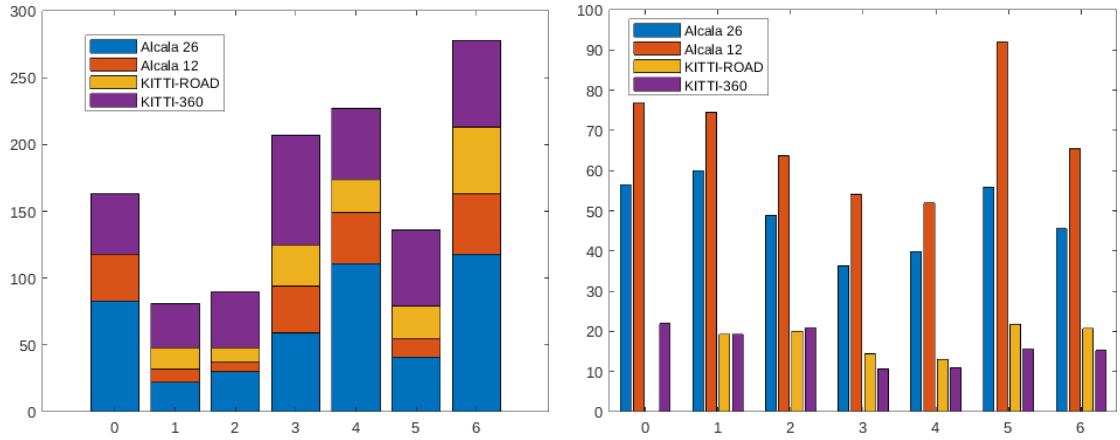


Figure 4.6: Dataset samples distribution diagram.

Distribution of intersection types (a) and sequence lenght (b) for KITTI, KITTI-360, and the two sequences of the Alcalá de Henares dataset.

Figure 4.7: Examples of images from the Alcala dataset.

Each column represents an example of the seven configuration geometries that were used in this work. The images were recorded at different times of the day and using a different camera setup.

4.2.3 Metric Learning

The so-called *metric learning* represents an exciting and emerging technology in the machine learning community. This notation, used as a generic term to indicate a distance, a correspondence, or difference between elements, allows for a slightly different classification approach from the classical classification approach. As it is well described in work in [Bellet et al., 2014], where they use a metric-learning technique, what can be done is to shift the classification problem towards an optimization problem, so that pair-wise or triplet-based distance constraints between items are both enforced and minimized through a specific loss function.

From a technical perspective, the distance is evaluated starting from the *embedding vectors* associated with a couple of input images, where the *embeddings* are generated by a function $f(\cdot)$ in the form of CNN, see Figure 4.8. For example, when using the triplet loss

approach, the idea is that given three samples and the associated vectors called anchor, positive and negative, a generic distance d satisfies Equation 4.2.

$$d(f(a), f(p)) < d(f(a), f(n)) \tag{4.2}$$

In this Ph.D., two different implementations of the metric-learning technique are used depending on the specific problem analysis. These include an ad-hoc implementation through the teacher/student paradigm and the much more exhaustive work presented by Kevin Musgrave in [Musgrave et al., 2020b] (please note that the remainder of the manuscript is referred to as the metric-library).

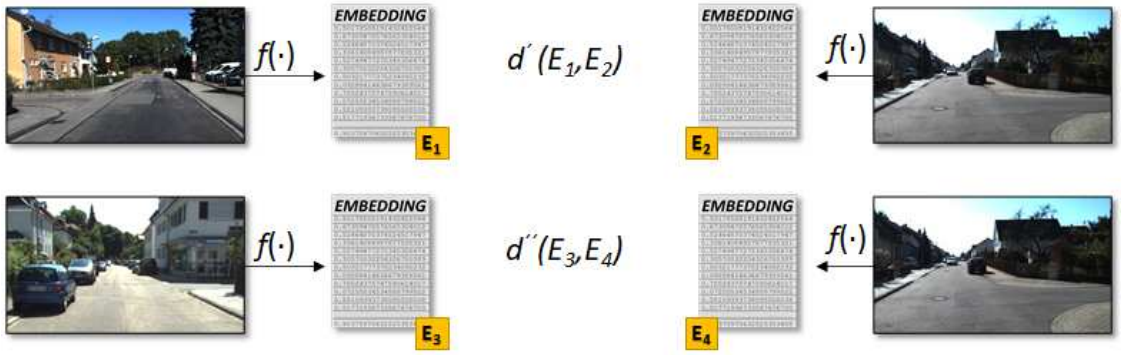


Figure 4.8: Image distance diagram.

The distance between the two images is calculated using the numerical N-dimensional embeddings resulting from a generic CNN network.

4.2.3.1 Teacher/student

The first learning paradigm is proposed to evaluate the standard end-to-end scheme concerning the so-called teacher/student. Among all the possible applications, the idea behind this paradigm includes transferring knowledge from a simple domain to a much more complex one.

4.2.3.1.1 The intersection Model

From a technical perspective, the proposed starting domain to transfer the knowledge consists of a synthetic set of simple BEV images generated with the intersection generator proposed in [Ballardini et al., 2017, Ballardini et al., 2019]. This simple intersection generator, shown in Figure 4.9, can render all of the seven configuration classes commonly used in the literature as seen in the state-of-the-art, in the form of binary masks, such as those shown in Figure 4.10. From now on, these images will be referred to as *Model-Based Bird Eye View (MBEV)*. These masks contain the shape of the most common intersections entities found in the mid-size cities. The model parameterization allows us to generate different intersection geometries, i.e., changing the number and position of intersecting

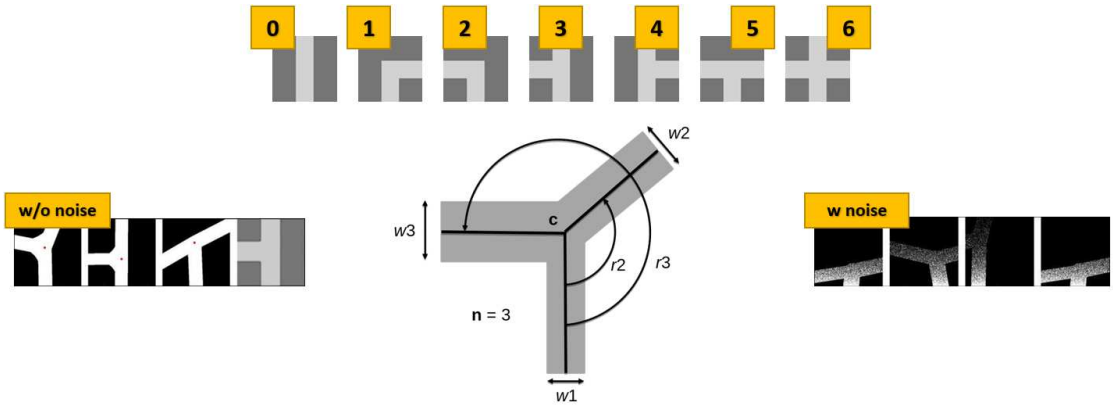


Figure 4.9: Intersection model diagram.

On the top of the image, the seven so-called *canonical* intersection classes along with the model used to generate the training dataset. In the leftmost part of the figure, identified with the label *"w/o noise"*, a triplet consisting of two samples of the canonical *type-three* (for simplicity also shown in the last box of the row) and a different one, e.g., *type-five*. Before passing these images to the teacher network during the training phase, random noise is added to each of the images by an increasing extent along the vertical direction. The effect is shown in the rightmost part of the figure identified with the label *"w-noise"*.

Figure 4.10: More examples of MBEV images generated with the model in Figure 4.9.

roads, the center position for the generated image, and finally, the roadway of each road segment involved. Without pretending to be the definite intersection generator, as many configurations still cannot be represented with this proposal, e.g., complex intersections that can be typically seen in extensive metropolitan areas, this model allowed us to assess the intersection classification capabilities of the evaluated network configurations using the datasets proposed in Section 4.2.2.1. This model also acted as a trivial data-augmentation scheme for the CNN during the training phases of the teacher networks. To mimic how some of the training images are created, an increasing amount of random noise is added optionally starting from the bottom part of the mask in a line-by-line fashion, as visible in Figure 4.9.

4.2.3.1.2 Triplet Loss

Within the plethora of metric-learning different patterns, the triplet approach described in [Schroff et al., 2015] has been used. In this technique, a set of three images (M_1^a, M_2^s, M_3^d) belonging to the same domain, i.e., M, composed of one *anchor* class image M_i^A, a sample of the same class (positive-anchor) M_j^S and a different class sample (negative-anchor) M_k^D, are passed through the triplet margin loss function. Again, the idea is to have a

CNN generating a high-dimensional embedding vector associated with input images. The distance between vectors belonging to different intersection classes should be higher than the distance derived from the vectors of same-class intersections. This loss function is defined similarly to each of the two parts of Equation 4.2, as follows:

$$\mathcal{L} = \sum_{i}[d(f(M_i^A), f(M_j^S)) - d(f(M_i^A), f(M_k^D)) + m]_+ \tag{4.3}$$

where $[\cdot]_+$ means $max(0, [\cdot])$ and $d(x_i, y_i) = \|x_i - y_i\|_p$ with p as the norm degree for pairwise distance and m is an extra scalar value used to extend the *margin* between the embeddings.

4.2.3.1.3 RGB pre-processing

In order to exploit the MBEV images generated with the intersection model, a set of three different pipelines is created to transform the RGB images into a similar viewpoint. This set includes:

- *3D-Generated Bird Eye Views (3D-BEVs)*: this first transformation, applicable only to datasets supporting stereo-camera configurations, creates a bird's eye view representation of the scene using a 3D-reconstruction process. This transformation is the most accurate 2D plan view that could be generated from images, as no distortions are introduced in this procedure. The effect of the virtual camera can be seen in the center box of Figure 4.11. The work in [Xu and Zhang, 2020] is used to generate the depth image that in turn allowed us to create a 3D representation and then the desired 2D image. Please notice that having a 3D representation allows us to change the virtual camera position, retaining the scene's consistency and simultaneously acting as a data augmentation methodology.

- *Masked 3D-Generated Bird Eye Views (3DMASKED-BEVs)*: for this second transformation, the previous pipeline is extended by including the segmentation results exposed in Chapter 3 and published in [Hernández et al., 2020] to remove the 3D information that does not belong to the road surface, thus generating an image containing only road pixels. The central insight here is to evaluate whether the classification may benefit from less cluttered yet pre-segmented images. For this purpose, the previous depth image is combined with the generated road mask before creating the 2D view.

- *Warping with Homographys (WARPINGs)*: this last transform tries to overcome the limitation of stereo and LiDAR availability at the cost of introducing distortions in the generated images. Standard computer vision techniques are applied to create a homography between the RGB image and the desired 2D image. As homographies are only defined for flat surfaces, but roads usually have at least slight roughness and undulations, this image transformation introduces distortions to the final image, as depicted in the *warping* images in Figure 4.11. Moreover, since fixed homographies are used, the actual attitude of the vehicle introduces similar distortion effects as the vehicle moves along its route. Nevertheless, since the videos have more than one

frame-per-intersections, this effect can be conceptualized as a data augmentation scheme, as the same intersection visually appears different across successive frames. Please notice that this scheme is the same one proposed for the RGB-Baseline described in Section 4.2.2.

All of these three configurations, where feasible, i.e., if LiDAR data is not available 3DMASKED-BEVs images are not created, were evaluated concerning the learning schemes presented in this research in addition to the primary RGB data.

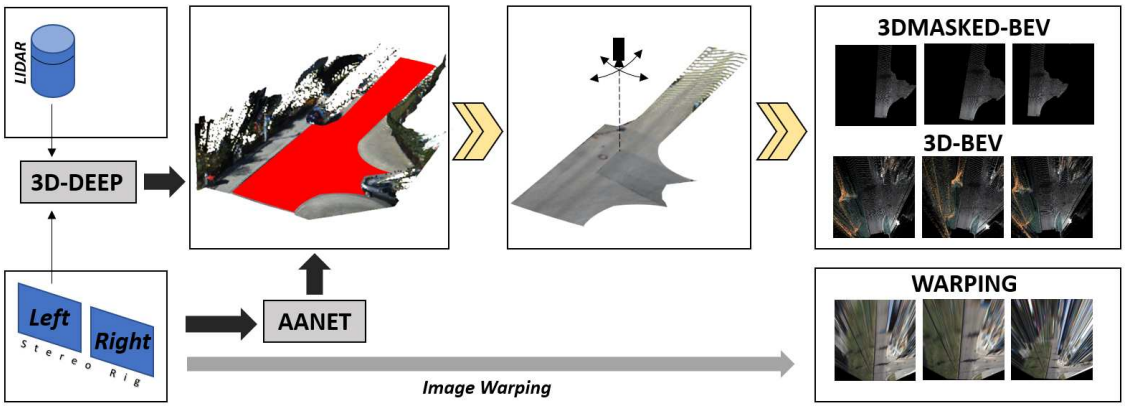


Figure 4.11: Image transformation workflow.

The figure depicts the overall pipeline used to generate the 2D planar images used in this work. Among them, only the 3DMASKED-BEV uses the LiDAR data. In the central part of the image, the picture shows the two methods of the *Data augmentation scheme*, i.e., the 3D-based realistic virtual camera that exploits the 3D point-cloud based reconstruction of the image and the 2D-based homography.

4.2.3.1.4 Applying the teacher-student paradigm

Besides the teacher/student paradigm, the central insight is learning a shared embedding space between the 2D images created with the intersection model and those generated by the pipelines mentioned above. This approach is inspired by the work of Cattaneo [Cattaneo et al., 2020], which performs visual localization using 2D and 3D inputs in a bi-directional mode, teaching two networks to create a shared embedding space and thus enabling a two-way localization, starting either from 2D or 3D inputs.

Recalling the metric technique described in the first part of Section 4.2.3, the teacher-student paradigm introduced some minor changes, particularly to Equation 4.2. In detail, given two instances of the same intersection class belonging to different domains, e.g., *Class 0* in domains *D1* and *D2* ($D^1_{C=0}$ and $D^2_{C=0}$), such as $D1$ is the space of the images with the intersection model, and $D2$ is the RGB-transformed space, and two different non-linear functions $f(\cdot)$ and $g(\cdot)$ represented in the form of CNN, the distance between the embeddings is lower than any other negative intersection instance, e.g., $D^2_{c=2}$. Formally, given the *Intersection-Model* domain M and the *Camera* domain C such that $M = C = \{0, 1, 2, 3, 4, 5, 6\}$, each of which contains the seven intersection typologies

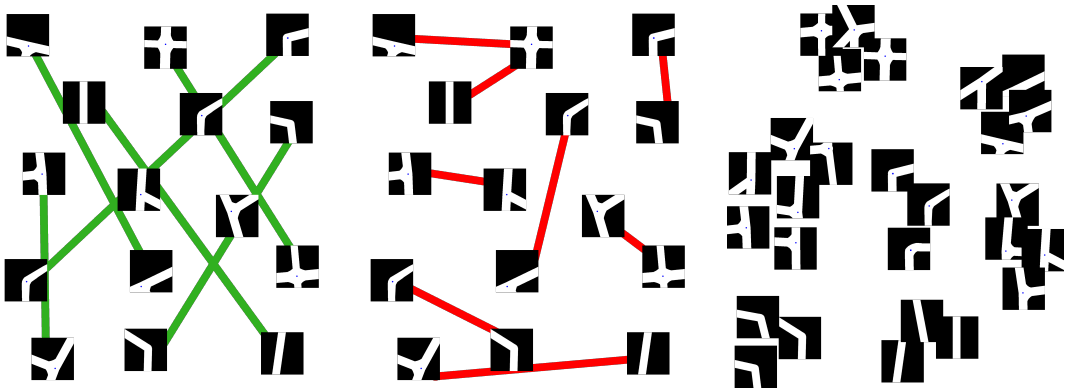
considered in our intersection model, and given one element $m_i \in M$, then Equation 4.4 is satisfied $\forall i, j \in C | i \neq j$, where $d(\cdot)$ is a distance function.

$$d(f(m_i), g(c_i)) < d(f(m_i), g(c_j)) \tag{4.4}$$

Regarding the teacher network, which is the first part to work with, it is trained with the triplet loss scheme mentioned before, but this time the images generated from the intersection model are used, see Figure 4.12. Once the teacher model has been trained, the student network is trained using the pre-processed RGB images presented in Section 4.2.3.1.3 as input data in a way to obtain a similar embedding vector. Towards this goal, the loss-function of the student network is composed as follows:

$$\mathcal{L} = \sum_i [d(f(M_i^A), g(C_i^S))] \tag{4.5}$$

where M and C are the model-domain and camera-domain respectively and *Mean Squared Error (MSE)* was used as distance function $d(\cdot)$ between the embeddings. Differently from Equation 4.3, here, the triplet scheme is replaced by a pair-wise distance between the embedding vectors. It is worth mentioning that to maintain a consistent distance within same-class classifications, M_i^A elements were chosen not from the list of embedding vectors used in the training phase of the teacher network but rather from the average of 1000 new random samples generated after the teacher network was trained, i.e., never seen before from the CNN. These per-class averages, i.e., cluster centroids, are shown in Figure 4.13 with black crosses and represent therefore the M_i^A set.



(a) (b) (c)

Figure 4.12: Metric learning hypothesis.

The picture depicts the intuition behind the metric learning approach. Good (green) and bad (red) constraints in (a) and (b) respectively are used to find a metric in a way that similar images are grouped as in (c) in clusters. Besides this first intuition of constraints in terms of *pairs*, we can imagine *triplets* to increase the complexity of the optimization problem further, helping the system to obtain better and broader distances between clusters.

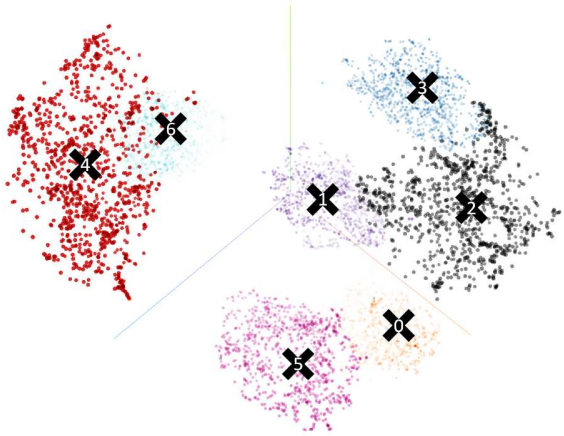


Figure 4.13: Embedding clusterization visual space.

The embedding space is visually represented using 7K samples from the intersection model, using the *T-Distributed Stochastic Neighbor Embedding (T-SNE)* function. In black, we conceptually represent the centroid of each of the clusters.

4.2.3.1.5 Training details

A data augmentation process was introduced in both networks to avoid overfitting during its training phase. A set of 1000 per-class intersection configurations is generated by sampling the generative model for what concerns the teacher network. A normal random noise is applied to the seven *canonical* intersection configurations on each parameter involved in the generation of the intersection, e.g., width, angle, and intersection center, in a measure of [2.0m, 0.4rad, 9.0m]. As far as noise is concerned, an increasing number of random noise is added to each line, always starting from the bottom of the image, to mimic the 3D density effect of the 3DMASKED-BEVs. At this time, despite its triviality, it should be noted that the point-density of 3DMASKED-BEVs is not constant over the distance with respect to the vehicle. Therefore, to simulate comparable MBEVs random noise is added proportional to the distance, see Figure 4.9. Regarding the student network, since the low number of intersections present in the two KITTI datasets in comparison with the overall number of frames, data augmentation is performed by adding a 6-DoF displacement to a looking-down virtual camera initially set at $[10m, 22.5m]$ above the road surface and $[17, 22m]$ in front of the vehicle for the KITTI and KITTI-360 respectively. Due to the nature of type-1 and type-2 intersection classes, which contain any curve without a specific curvature threshold, the rotation along the vertical axis is zeroed to limit the chance of assimilating these samples to the type-0 class. The code leverages the PyTorch 1.6 learning framework [Paszke et al., 2017], and both teacher and student images were scaled to images with a size of 224x224 pixels.

4.2.3.2 Metric library

The investigation with the metric library began by directly verifying the benefits introduced by the metric learning approach. First, according to Equation 4.3, different

distance functions were evaluated. Starting from the basic $\|\cdot\|_1$ and $\|\cdot\|_2$, the distance function set has been expanded to assess the impact of Signal-to-Noise Ratio and Cosine Similarity functions provided within the metric library. As the reader might guess, not all the samples will have similar scores. Unexpected geometries might also generate confusion and lead to unsatisfactory results. For this reason, the use of the *Triplet Margin Miner* feature is proposed to specify a minimum distance threshold for the generation of the triplets.

Specifically, this miner model has been used in this research, with two of its possible configurations, *"all"* and *"hard"*. The first of these configurations lets estimate the loss value by using all the triplets that violate the preset margin. The second one allows for creating a subset of the previous triplets, where the positive example is further away from the anchor than the negative example and using them to calculate the loss. These two miner configurations allowed focusing the training efforts on the most challenging examples of the dataset for the neural network. However, for completeness, training without any miner has been performed, letting the system use any possible triplet regardless of its difficulty.

Since this training methodology works directly with distances, it is necessary to identify a metric to evaluate the system's performance. The metric library provides different functions that can be used in this situation, such as *Mean Average Precision@R (MAPR)*, *Mean Average Precision (MAP)*, *Precision-at-1 (PA1)*, *R-Precision (RP)*. All these values have been recorded during the training process to assess how the training was performing. However, the value taken into account to run the validation patience and select the best training is MAPR, as this metric operates directly on the embedding space and provides better information [Musgrave et al., 2020a].

Once the model has been trained, it is necessary to find a way to use it as a classifier, as an embedding does not directly refer to a specific intersection type. An easy and trivial way could be to use the metrics above to calculate the minimum distance for a specific class representative vector. For example, using a clustering procedure, the representative vector might be the centroid of each cluster. However, apart from relying on such elements as centroids, the following methods were also evaluated. Following the distance concept, a first approach consists in using the Mahalanobis distance rather than the Euclidean distance to calculate the classification of each new example in such a way to capture better the distance concerning the learned concepts. The second method consists of a transfer learning adaptation: once the network has been trained using the previous proposals, its weights are frozen, and a fully connected layer is trained to classify the embeddings returned by the network. Finally, the third proposal consists of training a SVM classifier using the embeddings of the training and validation sets. Once the classifier is trained, the implementation of the system in a real environment would consist of two steps, obtaining the embedding from the trained architecture and its classification employing the SVM.

All training activities performed with this methodology have been carried out with different data transformations, explained in detail in Section 4.2.3.1.3, for each of the three datasets already explained above.

4.2.4 Multi-frame approach

The research continued by assessing the results of multi-frame learning schemes. Since the system is intended for autonomous vehicles, it seems logical to think that the vehicle will get more information about the surrounding environment as it approaches the intersection, and the interpretation accuracies will increase, as confirmed in [Ballardini et al., 2017]. Therefore, implementing a system that can exploit the spatio-temporal information from a sequence of images until the vehicle is inside the intersection should give even better results.

4.2.4.1 Recurrent neural networks

With the mentioned idea of further refining the work done so far with the CNN networks, possible architectures were investigated that could be combined to perform temporal integration of the data. The two predominant models for this in state-of-the-art are the *Gated recurrent unit (GRU)* and LSTM architectures. These networks usually have data packets of vectors grouped in time sequences as input. Since the CNN architectures that have been used so far return embeddings, the only process necessary to group the two architectures lies in the temporal aggregation of the embeddings. For this, the following process has been tracked. First, the images have been grouped into sets belonging to the same intersection forming a single temporal sequence. These samples are then passed through the already trained CNN, which returns a sequence of embeddings. It is worth mentioning that each recurrent network was trained with the best-performing CNN previously trained for each dataset and data type configuration. Once the CNN has processed all the batch sequences, the shortest ones are zero-padded so that all the sequences that compose the data block supplied to the recurrent network have the same length. Finally, since both GRU and LSTM return feature vectors, they must be classified to obtain a label to tell us to which class it belongs.

Similarly, as has been done for the metric approaches, two different methodologies were used to classify the embeddings: the inclusion of a fully connected layer trained simultaneously with the recurrent architecture and the use of an SVM classifier in a similar way as described in Section 4.2.3.2. Several architecture variations were made to perform an adequate ablation study with both architectures (GRU and LSTM). These variations range from the number of layers in the network or the hidden layer's size to the internal dropout of the architecture. The training has been performed using all the available datasets and respective transformations presented in Section 4.2.3.1.3. More detailed information on training will be provided later in Section 4.3.2.4.

4.2.4.2 Video Classification Networks

Apart from the recurrent architectures, a second exciting network design for temporal sequence analysis is represented by the expansion of standard 2D architectures. The basic idea is to feed a standard classification network with a set of consecutive images, allowing for the creation of so-called spatiotemporal filters. The main issue with this architecture, as mentioned before in Chapter 3, is the considerable increase in terms of

network parameters with respect to the corresponding 2D network version, which makes the training phase harder and computationally heavy. Moreover, the 3D extension also seems to neutralize the benefits of pre-trained models [Carreira and Zisserman, 2017]. Extensions to these basic video recognition architectures were introduced in the very last months by the works presented in [Feichtenhofer, 2020], where a family of efficient video networks is presented. A preliminary remark that must be made is that all of these architectures are usually employed to detect specific activities in video sequences. Typical datasets include KINETICS [Kay et al., 2017], Charades [Sigurdsson et al., 2016], EPIC-KITCHENS [Damen et al., 2020], or something-something [Goyal et al., 2017], where a huge set of activities is provided in terms of short videos. However, it is reasonable that similar actions or high-level *concepts* could also appear in a sequence representing a vehicle approaching a type of intersection. With the idea of having a similarity between the action *"picking something up"* and *"approaching a type-x intersection"* and the promising results of these approaches, a decision was made to perform a set of preliminary experiments using the works proposed in [Fan et al., 2020] and generally contained within the PyTorchVideo framework. This kind of approach is the first time it is evaluated in the intersection classification context to the best of our knowledge.

4.2.5 Artificial data-augmentation: GAN

As previously stated when talking of datasets, one of the major issues affecting all the aforementioned techniques is the limited availability of intersections, primarily in terms of intersection instances rather than the number of frames. In an attempt to overcome this limitation, several GANs frameworks were implemented and compared, from more simplistic approaches like DCGAN, WGAN, and CGAN to the last state-of-the-art networks, such as StyleGAN2, SWAGAN, or StyleGAN-ADA, which includes an adaptive augmentation procedure to deal with limited data regimes [Karras et al., 2020a]. Given the limited amount of data, the best quality and regularization metrics results were founded for SWAGAN and StyleGAN2 with the adaptive discriminator augmentation. SWAGAN [Gal et al., 2021] noticeably increases computational performance; however, some checkboard artifacts appeared in the images, and it achieved worse perceptual path length regularization. Hence, StyleGAN2-ADA was finally chosen. Figure 4.14 depicts some generated examples of the latter after 800K epochs.

(a) (b)

Figure 4.14: StyleGAN-2 image comparison.

On the left, an example of the performances of StyleGAN-2. On the right, an example of the output of the StyleGAN-2-ADA network after 800K epochs. However, to avoid presenting too similar images, we decimated the input images to have a constant frame rate equal to 1 fps.

Following the intuition that feeding the network with warped images could improve the generative process since the network would be able to focus only on intersections, different tests were performed training the GAN with RGB images and with warped images.

In order to balance less represented classes, the GAN was trained following a conditional setting. However, for RGB images, the best results were found for the non-conditional framework. Hence, a new augmented dataset was created by generating 10K random images and labeling them frame by frame, discarding those which do not depict an unambiguous intersection. Figure 4.15 shows some examples of good and bad generated images. One issue found with the generated images is that most of them were similar to the original ones. Probably, this matter is because the number of training images obtained with all the KITTI and Alcalá dataset sequences is more than six times smaller than the 30k training images used in the StyleGAN-ADA original contribution, [Karras et al., 2020b]. Notwithstanding, results show better classification performances with the augmented dataset

(a) (b) (c) (d) (e) (f)

Figure 4.15: Generated RGB intersections.

The image shows some examples of correctly generated RGB intersections, included in the new augmented dataset (a-c), and discarded ones (d-f).

Although most generated images were realistic enough, many of them were discarded due to inconsistencies on the road or an unclear intersection. This issue is currently being tackled by introducing new elements to the standard StyleGAN proposal, and it is planned to present the outcomes in future work.

On the other side, generated warped images showed similar performance when training in a conditional setting, probably due to the amount of information contained in these types of images. The latent space is better regularized, and generated images are closer to the real ones in the embedded space. Therefore, 10 thousand warped images were generated and added to the warped training set. Figure 4.16 shows some examples of the generated warped images under a conditional setting. It can be seen that they are realistic when compared to the real ones.

4.3 Experimental Analysis

The following section presents the extensive experimental results of all the tests performed with the different methodologies. These tests have been performed using an Nvidia DGX-A100 server.

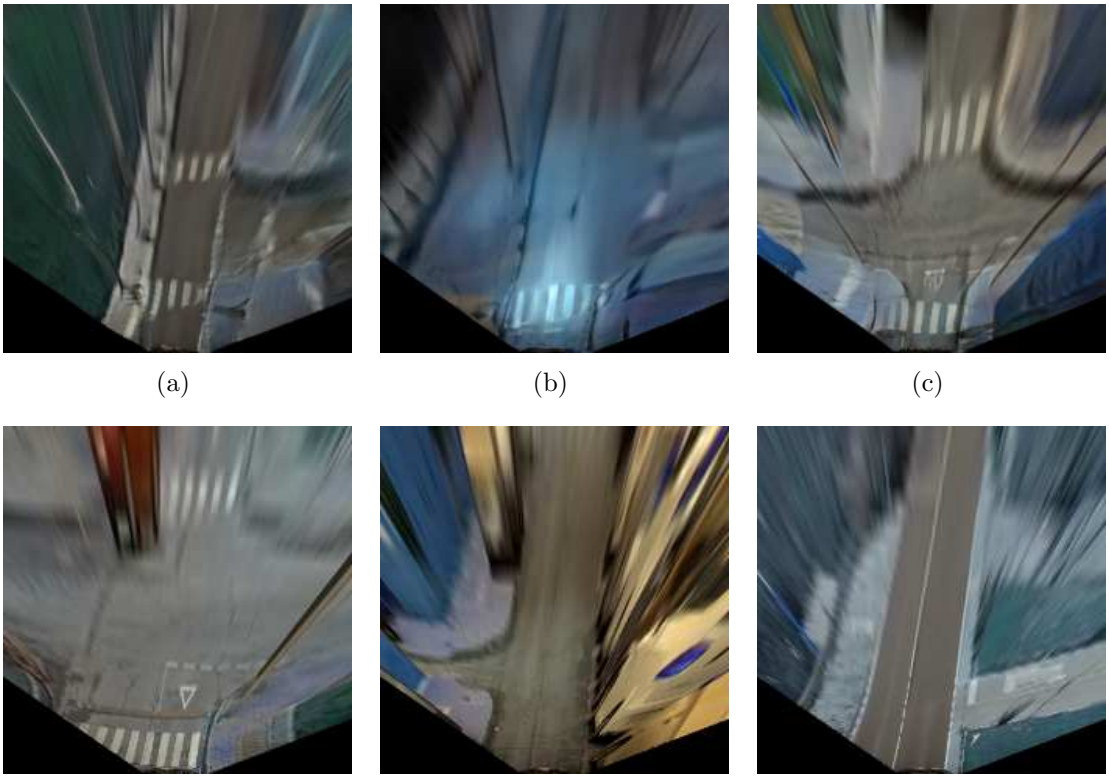


Figure 4.16: Real and synthetic warped images.

In the first row: some examples of generated images when training the GAN with warped real images. In the second row: examples of real warped images.

4.3.1 Experimental set-up

The test battery, as described below, has been configured using Wandb [Biewald, 2020], which has allowed performing an extensive ablation study using the automatic search procedure called *sweep*. The Wandb service automatically calculates the best combination of hyperparameter values by performing a *sweep*. In order to work with embeddings of similar size, and given that different network architectures have been used, a space reduction should be made in those architectures that return larger feature vectors. To adapt the returned vectors among different architectures is necessary to use fully connected layers to reduce the size of the embeddings to 512 values. This value comes from the embeddings' size used in the initial work and has been considered a standard for this research. The selected architectures are ResNet and VGG in all its versions, the two versions of MobileNet-v3 and Inception-v3. The used datasets have been divided, regardless of the number of images contained in each dataset, into the following percentages: 70% for the training set, 20% for the validation set, and 10% for the test set.

4.3.2 Experimental results

The first training sessions carried out are those of direct classification, on which comparisons will be made with the rest of the methodologies. This way of action has been done because, in the initial proof of concept, it was the chosen way of proceeding, and for simplicity, it has been decided to continue with it.

As stated in Section 4.2.2, a sweep has been initially configured to find the best training solution for the proposed architectures using the direct classification paradigm with the RGB and WARPED images. The baseline results are shown in Table 4.3.

Table 4.3: Results in direct classification.

			ResNet					**VGG**			**MobileNet v3**		**Inception v3**
			18	*34*	*50*	*101*	*152*	*11*	*13*	*16*	*large*	*small*	
KITTI ROAD	RGB	Validation	0.69	0.71	0.69	0.72	0.70	0.58	**0.74**	0.73	0.66	0.69	0.65
		Test	-	-	-	-	-	-	**0.53**	-	-	-	-
	Warping	Validation	0.60	0.67	**0.70**	0.68	0.60	0.60	0.63	0.61	0.65	0.60	0.67
		Test	-	-	**0.62**	-	-	-	-	-	-	-	-
KITTI 360	RGB	Validation	0.83	0.82	0.77	0.81	0.81	0.80	0.84	0.81	0.82	0.73	**0.87**
		Test	-	-	-	-	-	-	-	-	-	-	**0.78**
	Warping	Validation	0.81	0.78	0.83	0.84	0.80	0.81	0.80	0.82	0.83	0.80	**0.87**
		Test	-	-	-	-	-	-	-	-	-	-	**0.70**
ALCALÁ	RGB	Validation	0.82	0.87	0.85	0.86	0.80	0.74	0.86	0.85	0.89	0.80	**0.89**
		Test	-	-	-	-	-	-	-	-	-	-	**0.94**
	Warping	Validation	0.88	0.91	0.82	0.90	0.88	0.85	0.89	0.85	**0.91**	0.87	0.91
		Test	-	-	-	-	-	-	-	-	**0.92**	-	-

Results using direct classification by the proposed architectures using RGB images and bird's eye view images from the three selected datasets. Marked in bold are the best accuracy values for each type of data in both training and validation.

4.3.2.1 Teacher training

The *teacher* is trained with artificial images created at run-time by the mentioned intersection model. These images try to simulate intersections as they were obtained from BEVs to create an easy yet effective classification problem for the *teacher* network. Since the primary use of the *teacher* network is to obtain a set of embeddings that can represent road junctions useful for the training of the *student* network, the loss function used is Triplet loss. As in the preliminary work [Ballardini et al., 2021], this decision was taken to separate as much as possible the space between the embeddings of different types of intersections so that two similar types are as close as possible and two different types as far apart as possible. Because the images are generated at run-time, once the anchor and the positive samples are chosen, the negative one is selected randomly from the remaining labels.

The teacher performance was evaluated using all the proposed architectures as the backbone and a set of 2000 artificial images for training and 1000 for validation. All training runs, regardless of the backbone model, quickly became overfitted. The accuracy values in the training and validation sets reached 1, so the model stopped learning and was no longer generalizing. In case the architecture was too deep, the training was repeated with a customized network, which contained only one or two convolutional blocks (Conv + BN + ReLU) depending on the chosen configuration and a classifier composed of a fully connected layer. The results obtained were similar, so the proposed architecture was discarded. In our opinion, this overfitting may be due to the synthetic images created by the intersection model since those images are effortless to classify. Therefore, the path is as follows: since it is still believed that the ease with which the embeddings of the synthetic images were grouped can be of great help in the training of the student network, we select as weights for the trained network those belonging to the checkpoint immediately prior to the time when the accuracy value in the validation set reached 1.

As seen in Figure 4.13, performing a clustering analysis over the resulting set of embeddings shows clearly how the CNN can distinguish the intersection classes. Since it is intended to use this information to train the student network, the centroids of each cluster are calculated to be used as reference marks in the subsequent training.

4.3.2.2 Student training

In order to explore the widest solution space possible, a Wandb [Biewald, 2020] sweep has been deployed with the following parameters, randomly chosen at each iteration:

- Learning Rate: max: 0.01 min: 2.5e-06
- Optimizer: adamW, adam, rmsprop, sgd, ASGD, Adamax
- Loss Function: SmoothL1, L1, MSE
- Batch Size: 8, 16, 34, 64, 128

Table 4.4 presents the results obtained using different input data types (RGB, WARPING, 3D-BEV, and 3DMASKED-BEV) for the three used datasets (KITTI-ROAD, KITTI-360, ALCALÁ). Since the training activities have been performed with *sweeps* the table reports the maximum validation accuracy value among all the training performed in the *sweep* for each architecture. The value corresponding to the accuracy with testing data is obtained using the architecture and the weights corresponding to the maximum validation accuracy value among all reported, marked in bold. The operation has been performed for each dataset's available data source.

The experimental activities were addressed to verify:

1. First, how the different data transformations exposed in 4.2.3.1.3 affect the results in the Teacher/Student paradigm and the possible benefits.
2. Second, whether the use of the Teacher/Student paradigm substantially improves results over the direct classification of the data.

3. Third, if the data recording methodology and the camera angle are critical points in obtaining good results.

4. Fourth, whether the inclusion of new architectures in training produces a significant variation in the results obtained.

Table 4.4: Results in Teacher/Student paradigm.

			ResNet					**VGG**			**MobileNet v3**		**Inception v3**
			18	*34*	*50*	*101*	*152*	*11*	*13*	*16*	*large*	*small*	
KITTI ROAD	RGB	Validation	0.51	0.50	0.53	0.50	0.57	0.66	0.70	0.71	**0.75**	0.71	0.57
		Test	-	-	-	-	-	-	-	-	**0.64**	-	-
	Warping	Validation	0.47	0.58	**0.70**	0.59	0.52	0.61	0.63	0.66	0.66	0.64	0.57
		Test	-	-	**0.61**	-	-	-	-	-	-	-	-
	3D-BEV	Validation	0.45	0.55	0.65	**0.67**	0.58	0.56	0.62	0.60	0.60	0.62	0.46
		Test	-	-	-	**0.64**	-	-	-	-	-	-	-
	3D-Mask	Validation	0.64	0.71	0.72	0.71	0.71	0.70	0.72	0.72	**0.73**	0.72	0.69
		Test	-	-	-	-	-	-	-	-	**0.71**	-	-
KITTI 360	RGB	Validation	0.60	0.65	0.79	0.79	0.83	0.73	0.82	**0.86**	0.8	0.79	0.60
		Test	-	-	-	-	-	-	-	**0.75**	-	-	-
	Warping	Validation	0.63	0.61	0.69	0.8	0.63	**0.85**	0.79	0.83	0.82	0.73	0.70
		Test	-	-	-	-	-	**0.73**	-	-	-	-	-
	3D-BEV	Validation	0.51	0.65	0.71	0.75	0.65	0.61	0.68	0.59	**0.79**	0.73	0.57
		Test	-	-	-	-	-	-	-	-	**0.73**	-	-
	3D-Mask	Validation	0.65	0.61	0.76	0.78	**0.79**	0.74	0.77	0.78	0.78	0.76	0.78
		Test	-	-	-	-	**0.67**	-	-	-	-	-	
ALCALÁ	RGB	Validation	0.76	0.49	0.88	0.88	0.86	0.82	**0.90**	0.87	0.88	0.77	0.81
		Test	-	-	-	-	-	-	**0.96**	-	-	-	-
	Warping	Validation	0.88	0.87	0.88	0.87	**0.91**	0.88	0.90	0.80	0.90	0.88	0.90
		Test	-	-	-	-	**0.90**	-	-	-	-	-	-

Results by dataset and type of data obtained using the Teacher/Student paradigm. Marked in bold are the best values for each type of data in both training and validation. The values shown are for accuracy.

Looking at the results in Table 4.4, no substantial difference can be seen in the precision values according to the input data type. Taking into account, for example, the results in the KITTI-360 dataset, the lowest validation value is 0.79 and the highest 0.86, making a disparity of 0.07. The differences in test values are equally minimal, 0.08 between the value achieved with the 3D-Masked data and the value achieved with the RGB images. This pattern seems to be repeated across datasets, which leads us to think that the pre-processing of the data does not seem to be such a relevant issue.

Focusing on the validation results obtained by the different architectures, there does seem to be a notable difference. For example, in the RGB images of the KITTI-ROAD dataset, there is a 0.25 difference between best and worst architecture. However, although similar variations are repeated between the different datasets and data types, the best

and worst-performing architectures do not seem to be the same. This variation leads us to think that other factors cause the inequality, not the selected architecture.

The results do not seem to be very enlightening regarding the second working hypothesis. Comparing the values obtained respectively with the baseline and the Teacher/Student approaches (see tables 4.3 and 4.4), the differences between the values achieved on validation are minimal, of the order of 0.01. The accuracy values achieved in testing are also quite similar, the highest difference being 0.11, in the training of the KITTI-ROAD dataset with RGB images.

The results obtained in the preliminary work for this research showed a slight improvement in almost all the Teacher/Student paradigm results concerning the direct classification. However, this improvement does not seem to have appeared during the current research process. It is believed that this is because using the Teacher/Student paradigm is a risky option, and the results vary considerably depending on the initial configuration of the problem, especially if the data available are not extensive.

Finally, to validate the third working hypothesis, a comparison is made with the results obtained using the KITTI dataset and both the baseline and the Teacher/Student approaches (see tables 4.3 and 4.4). As can be seen, the results in the Alcalá dataset, both in direct classification and with the Teacher/Student methodology, are substantially higher than those achieved in KITTI in the same category. A clear example of this is the difference in accuracy between the KITTI-ROAD dataset and the Alcalá dataset when classifying RGB images.

In direct classification, the validation value for the Alcalá dataset is 0.15 better than in KITTI-ROAD, and the test value is almost 0.4 higher. These results are repeated within the Teacher/Student paradigm, with the Alcalá dataset being 0.15 better on validation and 0.30 on testing.

These results, in our opinion, are very enlightening. The two KITTI datasets suffer from a lack of images and field of view. These two points have been addressed in the Alcalá dataset. It is believed that the evident improvement of the results is mainly because the Alcalá dataset has a much wider field of view than the KITTI datasets. After all, although it also has more images, the difference is not so significant, especially when looking at the KITTI-360 dataset (see Section 4.2.2.1).

Since the Alcalá dataset is a proof of concept, a significant improvement in the image number and field of view will be addressed with a complete dataset in the medium term[3].

In addition, the lack of images in the KITTI dataset also seems to lead to some instability and overfitting in the training that could be observed during the sweeps, see Figure 4.17. This fluctuation makes us think that perhaps the conclusions obtained on the first two hypotheses are not definitive since working with a more extensive dataset would be necessary.

4.3.2.3 Metric-library training

Due to the results obtained with the Teacher/Student paradigm and the conclusions drawn from them, it is believed that it is necessary to use the data available at that

[3] https://invett.aut.uah.es/intersectiondataset

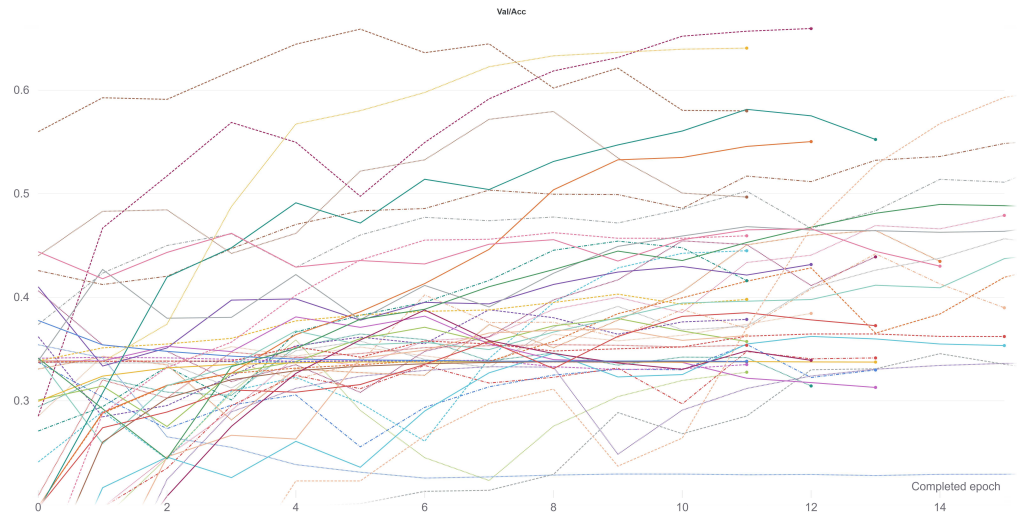


Figure 4.17: Results of the first 100 runs of a sweep.

The plot shows on the ordinates axis the accuracy metric used to evaluate the performances of the system. As the reader will notice, slightly different parameters lead to considerable differences in the accuracy metric values.

moment more extensively to find better results. That idea led to the metric learning library, [Musgrave et al., 2020b].

If we look at the previous methodology, the following process was used to calculate the losses when training the student network. First, the distance between each student's network embedding and the centroid of the corresponding label within each batch is calculated. Then, the losses are calculated based on how close the embedding is to the corresponding centroid according to that measure. Finally, the losses are balanced according to the weights of each class and averaged. Thus, if the batch has eight data samples, the losses will be calculated as a function of eight different distances. Unlike the Teacher/Student methodology, the metric-library does not use a teacher to set the reference centroids to each label but uses distance functions to separate/join as much as possible the embeddings returned by the network according to their labeling. This methodology change allows the number of distances computed per batch to be substantially increased since each batch element's distance with each other can be calculated. The actual number of distances to be estimated per batch is given by N/K, where k is the number of batch elements needed to determine the distance and n is the batch size. According to the researcher's criteria, this exhaustive way of calculating the losses during the training process could somehow circumvent the lack of data in the training sets. Then a set of experimental activities were addressed to verify:

1. First, whether the new comprehensive way of calculating batch losses helps to alleviate the problems detected in the previous paradigm with the lack of images.

2. Second, once the lack of images is solved, the usefulness of the data transformations and the improvement over the direct classification can be verified again.

Before discussing the results, specific points that differentiate them from those shown in the previous paradigm should be commented on.

Since the triplet loss function is implemented within the library and worked quite well in the teacher network training, it is the one that has been chosen to perform all subsequent training sweeps. In addition, the implementation of this loss function allows selecting between different functions to calculate the distance between embeddings, which allows a broader field of exploration.

In order to explore the widest solution space possible, as in the previous methodology, a Wandb [Biewald, 2020] sweep has been deployed with the following parameters, randomly chosen at each iteration:

- Learning Rate: max: 0.01 min: 2.5e-06
- Optimizer: adamW, adam, rmsprop, sgd, ASGD, Adamax
- Distance Function: SNR, Cosine Distance, Pairwise
- Batch Size: 8, 16, 34, 64, 128
- Margin: max: 5.0 min: 0.5 q: 0.5
- Miner: All, Hard

As a comment on the above parameters, the margin is the Triplet Margin Loss parameter that can be stated as the desired difference between the anchor-positive distance and the anchor-negative distance.

As said before, the Metric-library offers different options to calculate the model's accuracy from the embeddings returned by the network. During the training, several of them have been registered to have a global view of the training, but among them, MAPR [Musgrave et al., 2020a] has been selected to establish which is the best performing model, believing that it can be the one that best represents a good separation between classes. As stated before in Section 4.2.3.2, since MAPR is not reasonably comparable to accuracy and it is necessary to establish a direct classification methodology, the three proposed methodologies have been tested to obtain the results. Because these methodologies have only been used for testing, the validation values of the training cannot be fairly compared with other methodologies such as Teacher/Student or direct classification.

Initially, training a fully connected layer to classify the embeddings returned by the architecture was used as a testing methodology. This methodology was discarded since the validation values of the training were relatively low, especially compared to the other methodologies, and therefore the testing values were expected to be even lower.

Once this first methodology was discarded, the following two procedures were tested, SVM and Mahalanobis distance. The results of both were similar, however in most cases in the SVM was slightly better, so, and given the substantially longer computation time required to calculate the covariance matrix to use the Mahalanobis distance, it was decided to use SVM to perform the tests. Please notice that the values for the Metric Learning paradigm reported in Table 4.5 always belong to the value returned by this last methodology.

Table 4.5: Results in Metric learning paradigm.

			ResNet					**VGG**			**MobileNet v3**		**Inception v3**
			18	*34*	*50*	*101*	*152*	*11*	*13*	*16*	*large*	*small*	
KITTI ROAD	RGB	Validation	0.50	0.37	0.52	0.44	**0.54**	0.41	0.42	0.45	0.48	0.51	0.34
		Test	-	-	-	-	**0.56**	-	-	-	-	-	-
	Warping	Validation	**0.62**	0.47	0.40	0.35	0.46	0.37	0.41	0.35	0.47	0.56	0.38
		Test	**0.60**	-	-	-	-	-	-	-	-	-	-
	3D-BEV	Validation	**0.49**	0.48	0.41	0.43	0.47	0.29	0.32	0.30	0.44	0.40	0.36
		Test	**0.47**	-	-	-	-	-	-	-	-	-	-
	3D-Mask	Validation	**0.65**	0.51	0.51	0.50	0.45	0.46	0.52	0.41	0.49	0.54	0.59
		Test	**0.72**	-	-	-	-	-	-	-	-	-	-
KITTI 360	RGB	Validation	0.32	0.64	0.64	0.59	**0.74**	0.59	0.23	0.25	0.55	0.53	0.72
		Test	-	-	-	-	**0.69**	-	-	-	-	-	-
	Warping	Validation	0.63	**0.65**	0.31	0.42	0.38	0.65	0.31	0.60	0.64	0.41	0.44
		Test	-	**0.78**	-	-	-	-	-	-	-	-	-
	3D-BEV	Validation	0.57	0.53	0.29	0.40	0.62	0.36	0.42	0.33	**0.67**	0.56	0.60
		Test	-	-	-	-	-	-	-	-	**0.73**	-	-
	3D-Mask	Validation	0.65	0.61	0.76	0.78	**0.79**	0.74	0.77	0.78	0.78	0.76	0.78
		Test	-	-	-	-	**0.81**	-	-	-	-	-	
ALCALÁ	RGB	Validation	0.72	0.78	0.83	0.75	0.50	0.80	0.45	**0.84**	0.70	0.60	0.40
		Test	-	-	-	-	-	-	-	**0.94**	-	-	-
	Warping	Validation	0.74	0.81	0.72	0.52	0.81	0.71	0.77	0.59	**0.84**	0.83	0.82
		Test	-	-	-	-	-	-	-	-	**0.91**	-	-

Results by dataset and type of data obtained using the metric learning paradigm. Marked in bold are the best values for each type of data in both training and validation.

Looking at the testing results between Teacher/Student and Metric learning paradigms (see Tables 4.4 and 4.5), the higher number of comparison examples per batch has not substantially improved accuracy. Many values are similar or differ by only a few decimal places, such as the warped images in the KITTI-ROAD dataset or the RGB images in the KITTI-360 dataset. In some cases, as with the 3D images in the KITTI-ROAD dataset, it has even worsened slightly. Therefore, it is believed that the same problem of missing images can be seen in both methodologies and that further comparison between the available data has not led to a substantial improvement.

In turn, the comparison between the Baseline and the metric learning approach (Tables 4.3 and 4.5) yields similar conclusions to those in Section 4.3.2.2. As mentioned above, the values are still very similar and have not led to a change in the trend.

Looking backward to the previous comparisons with the Teacher/Student paradigm, the training using this new methodology seems to validate the assumptions established earlier for the first methodology. Comparing the results obtained in the Alcalá dataset with those obtained in the other two KITTI datasets, a substantial improvement in accuracy values can be observed. The accuracy reaches its highest level, with a 0.3 decimals increment, comparing the Alcalá dataset and the KITTI-ROAD dataset using

RGB images. These results virtually confirm that the critical point, as in the previous methodology, is above all the camera's field of view with which the images are recorded.

In order to explore the results further, they are broken down into other types of figures. Figure 4.18 shows the confusion matrices of the best tests by training methodology and dataset. As can be seen, the results are pretty good, more than it might seem if only the combined accuracy of the seven classes is evaluated. These disaggregated results show that the system's overall performance is good and that with better starting conditions, such as a better dataset, it can be excellent.

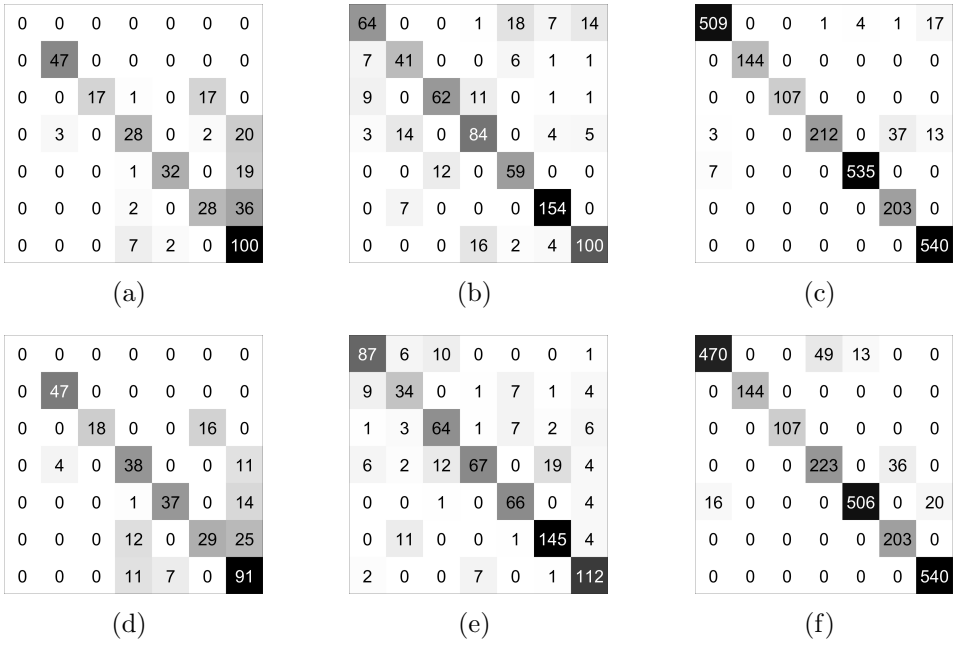


Figure 4.18: Confusion matrix in testing for the single-frame methodologies.

Different confusion matrix results in (a) student KITTI-ROAD, (b) student KITTI-360, (c) student Alcalá, (d) metric KITTI-ROAD, (e) metric KITTI-360, (f) metric Alcalá. Please notice that in (a) and (d) we did not introduce straight roads (type-zero intersections).

4.3.2.4 Recurrent Network Scheme Results

As stated before, it seems a reasonably straightforward deduction that vehicle traffic is an event that runs overtime. The approach to an intersection is not an event that only exists at point t. As the vehicle progresses to the junction, it looks logical to think it will receive more information. Therefore, it appears that if it is possible to group all the information from instant t to instant $t + n$ when the car is already at the intersection, better accuracy should be achieved when classifying each intersection. With this idea in mind, the next step considered is to group, in order of approach, all the embeddings that belong to the same intersection to be classified as a *single sequence*. The experimental activities were addressed to verify:

1. First, checking if temporal integration helps in achieving the goal of classifying intersections.

2. Second, if possible, try to see if this new approach offers more light on the previous results.

As explained in Section 4.2.4, the inclusion of an *Recurrent Neural Network (RNN)* was the most straightforward methodology to implement since they can directly work with embeddings as input. Training has been performed with two selected RNN architectures, GRU and LSTM. Preliminary results for both structures were very similar, with slightly superior ones belonging to the LSTM network. Therefore, the subsequent training was performed with an LSTM architecture. Tests have also been performed in such a way that it is possible to obtain a final classification of the embeddings returned by the RNN, being the options to choose, as specified in Section 4.2.4.1 SVM, Mahalanobis, and Fully connected layers. In this case, unlike in the single-frame approaches, the best results have been obtained by training a fully connected layer while training the RNN network architecture. In order to explore the most comprehensive solution space possible, as in the single-frame approaches, a Wandb [Biewald, 2020] sweep has been deployed with the following parameters, randomly chosen at each iteration:

- Learning Rate: max: 0.01 min: 2.5e-06
- Optimizer: adamW, adam, rmsprop, sgd, ASGD, Adamax
- Loss Function: Cross Entropy, Focal
- Batch Size: 16, 34, 64
- FC dropout: max: 0.5 min: 0.1 q: 0.1
- LSTM dropout: max = 0.5 min = 0.1 q = 0.1
- LSTM hidden layer: 256, 128, 64, 32, 16, 8
- LSTM layers: 1, 2

As a comment to the above parameters, the dropout of the fully connected layer is located just between the embedding and the classifier. The Focal Loss (FL) function has been implemented from the [Lin et al., 2017b].

Regarding the first assumption, Table 4.6 does not reflect that temporal integration has significantly changed the results. The validation values have increased in some cases, as in the KITTI-ROAD 3D images, but the test value is significantly worse, indicating overfitting. However, looking at the results in the Alcalá dataset, the values for the three paradigms are relatively similar. This connection leads us to think that the problems from previous approaches may have been transferred to this one. Some comparisons were made between the results obtained by the LSTM and the results obtained by the single-frame approach. A simple temporal integration was made by voting between the results of all the frames of the same sequence to make a fair analysis. The results were very similar,

Table 4.6: Results in multi-frame paradigm.

	KITTI ROAD		KITTI-360		Alcalá	
	validation	Test	validation	Test	validation	Test
RGB	0,69	0,55	**0,9**	**0,85**	0,85	0,84
Warping	0,82	0,45	0,82	0,73	**0,91**	**0,92**
3D-BEV	0,88	0,65	0,84	0,79	-	-
3D-Mask	**0,9**	**0,7**	0,89	0,81	-	-

Results by dataset and type of data obtained using temporal integration schemes. The accuracies in testing greatly decreases in all configurations on KITTI datasets but not with the Alcalá one.

and analyzing them in-depth leads us to confirm the assumptions in previous points. The problem lies on two fronts, the number of images and the viewing angle. The lack of images produces that when grouping them in sequences, the examples used to train the LSTM are smaller than in a single-frame approach, so it seems logical that the overfitting has increased. The lack of viewing angle in the KITTI datasets means that the network does not have enough information to extract good features from the crossing, something that cannot be solved with temporal information. If the feature extractor cannot divide the embeddings correctly, the LSTM will not be able to classify accurately.

4.3.2.5 GAN related results

Switching to GAN architectures' results, ten thousand RGB images and ten thousand warped images have been generated. As stated in Section 4.2.5, a conditional setting was used for the warped images but a normal one for the RGB images since it was noticed that a significant detriment in performance, obtaining images of poorer quality and with strange artifacts. Figures 4.15 and 4.16 show some of the examples of the generated RGB and warped images, respectively. Before including the artificially generated images in the original train dataset, it was tested if they were close to the real ones by feeding them to a model trained with images of KITTI-ROAD, KITTI-360, and the recorded datasets and comparing the codes in the embedding space. In order to do so, distances of these codes to the nearest clusters' centroids previously computed with the authentic images were measured. The same was done with a random dataset and checked the statistics of the three setups, showing consistency among the generated and authentic images, while random images' embeddings were far from all centroids.

Before including the artificially generated images in the original train dataset, it was tested if they were close to the real ones by feeding them to a model trained with images of KITTI-ROAD, KITTI-360, and Alcala datasets and comparing the codes in the embedding space. In order to do so, distances of these codes to the nearest clusters' centroids previously computed with the authentic images were measured. The same was done with a random dataset and checked the statistics of the three setups, showing consistency among the generated and authentic images, while random images' embeddings were far from all centroids.

Table 4.7 shows accuracy results for the Teacher/Student paradigm as well as for the metric learning scheme for both RGB and warped images. Please note that the MAPR metric is used for validation results in the metric learning scheme. Dataset *Real* refers to KITTI-ROAD, KITTI-360, and Alcala, while *Real + Fake* refers to the same datasets together with the generated images. An improvement in test performance can be seen for the four setups when augmenting the dataset with GAN images. In the RGB setting, validation accuracy is lower in the case of the augmented dataset, which reaffirms the hypothesis that it improves generalizability.

Table 4.7: Results with GAN-Augmented Dataset.

	Warped				RGB			
	Teacher/Student		Metric Learning		Teacher/Student		Metric Learning	
	Validation	Test	Validation	Test	Validation	Test	Validation	Test
Real	0.890	0.894	0.901	0.872	**0.885**	0.894	**0.893**	0.817
Real + Fake	**0.932**	**0.935**	**0.920**	**0.928**	0.876	**0.935**	0.882	**0.935**

4.3.2.6 Pytorch video results

The pilot experiments have been performed using ResNet3D and X3D networks [Feichtenhofer, 2020] together with the KITTI-360 and Alcalá datasets. Few changes to the original code were applied. Among them, all the mirroring data augmentation schemes were removed in the Charades data loader implementation and modified the number of classes from 157 to our seven basic geometries. Another distinctive feature of the experiments is related to the length of the input data, which strongly differs from the original video feed. As the intersections' frames were manually selected, the part of the code where a clip is randomly sampled from a whole video sequence was modified. Research experimental activities were addressed to verify:

1. First, how the video-analysis networks handle both RGB and MBEVs images, comparing their performances with respect to the learning schemes proposed in this research. This test was executed using the KITTI360 and KITTI360-masked imagery;

2. Second, how different frame rates affect the classification capabilities. For this second test, the Alcala-1 sequence was used, subsampling the original 30 fps to 6 and 15 fps. These experiments could not have been performed with the KITTI sequences due to the dataset low frame rate. Please notice that all the *actions* datasets originally used with these networks share the same length, which is not realistic with videos containing intersection approaches due to different speeds of the vehicle while approaching a generic intersection.

These tests were repeated using both the previously mentioned ResNet3D and X3D approaches. The system was trained on an Nvidia DGX-A100 machine, following the

video classification tutorial provided in the PytorchVideo GitHub repository, a variable epoch number between 200 and 1000 iterations with the CosineAnnealingLR Pytorch scheduler and the AdamW optimizer with different learning rates between 1e-1 and 1e-5 and weight decay set to $1e^{-4}$. As for the image size, a standard 224x244 size was used, similar to all the other approaches. Regarding the first experiment, tests showed that different clip frame-rate does not significantly impact the prediction performances, see Figure 4.19. This knowledge is of particular interest and suggests that high frame rates are not strictly required for this analysis, suggesting that the network can infer the geometry with just a few frames. A first interesting comparison can be made with respect to the different frame length experiments. As shown in Figure 4.21c, the performances of the LSTM system are better than all the different experiments performed with both the ResNet3D and the X3D models and the different time lengths used. It is worth saying that the Pytorch implementation of the LSTM considers different frame lengths, something that, by its nature PytorchVideo framework does not. This framework restriction is a clear advantage in favor of LSTMs approaches, as the approach to an intersection trivially depends on the vehicle's speed. This issue might be solved with approaches that involve multiple pathways for activity recognition, such as the SlowFast approach. However, this model could not be investigated further due to the PytorchVideo code's technical issues and the fact that it is at an early stage of development at the time of the research. Lastly, given the results, it can be concluded that the investigated networks do not achieve better results concerning the RGB inputs. The second experiment further corroborates the latter opinion, and here the advantages of MBEV images are undisputed. As shown in Figures 4.20a-4.20b and 4.20c-4.20d, both the transformed images have better results with respect to the corresponding RGB images.

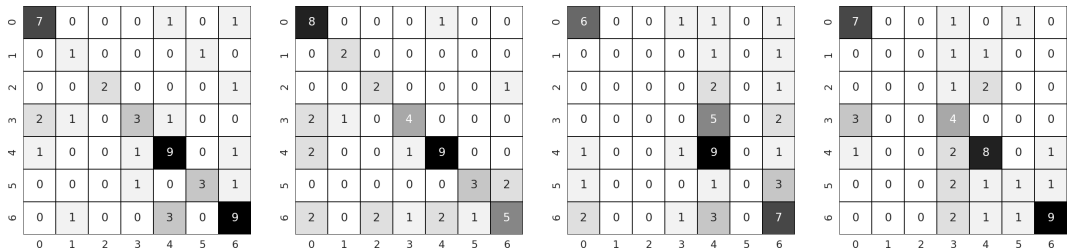


Figure 4.19: Confusion matrix test results with different frame length.

(a/b): ResNet3D model with input at 15 or 6 fps. (c/d): X3D model with input at 15 or 6 fps. Dataset: Alcalá-1, test-split, 51 total intersections.

Regarding the comparison between ResNet3D and X3D architectures, no clear advantage within the two architectures has been observed. The comparison with respect to the most similar temporal-sequence-oriented architectures is shown in Figure 4.21. As the reader can see from Figure 4.21a, the LSTM approach outperforms both ResNet3D and X3D approaches of Figures 4.20a and 4.20c, while the corresponding MBEV experiments obtain much more similar results, see Figures 4.21b *vs.* 4.20b and 4.20d respectively for ResNet3D and X3D.

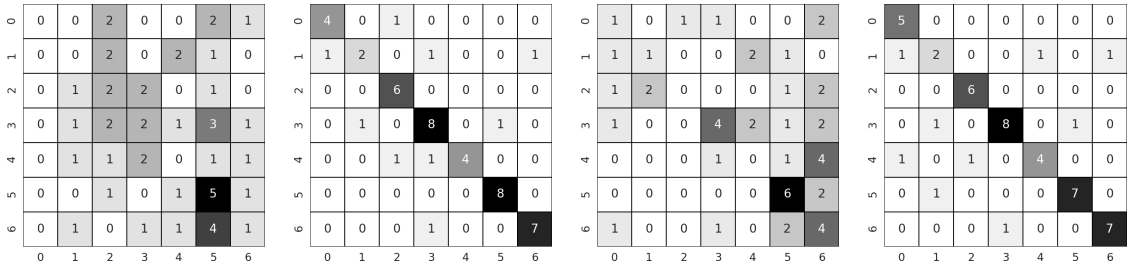


Figure 4.20: Confusion matrix test results with different view.

(a/b): ResNet3D model, using RGB and MBEVs respectively. (c/d): X3D model, using RGB and MBEVs respectively. Dataset: KITTI360, test-split, 48 total intersections.

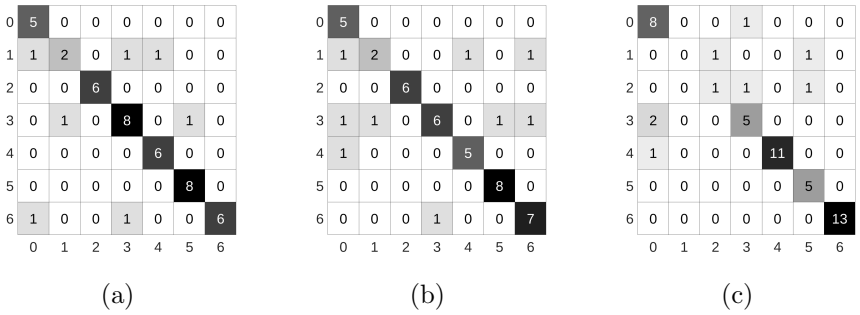


Figure 4.21: Confusion matrix test LSTM results.

Results using KITTI360 dataset and RGB data (a), MBEVs (b) and Alcalá-1 RGB dataset (c).

4.3.3 Experimental conclusions

The reader might appreciate that this doctoral thesis's research on intersection classification has suffered from a somewhat notorious drawback. This drawback has led to a noticeable lack in the research itself: comparative data. As mentioned in Chapter 2, it can be observed that work on intersection detection is not very profuse among the research community, as might be others, such as road segmentation. In our opinion, this is due to two main factors. The first, which is quite evident throughout this research work, lies in the lack of data. To the best of our knowledge, there is no specific dataset for urban intersection classification. This lack of data is not a minor problem, as it limits the research on two fronts. The first, and perhaps the most time-consuming, is that any research in the field must go through creating its own dataset from an already published dataset. Although it may seem a trivial statement, the latter must meet specific conditions for its use. Not just any dataset will do for autonomous driving since it must have enough occurrences for each type of intersection and be sufficiently differentiable so that the training is not biased. These conditions are not easy to achieve since many of the currently published datasets consist of fairly predefined and sometimes even repetitive

routes. The second is given by the previous need, which results in the fact that each investigation will work with a different dataset as a general rule. Even if the initial data set chosen is the same for two different investigations, the final result with which one works will certainly have notable differences due to differences in labeling. This restriction implies that if each research paper uses its own dataset. Strictly speaking, it cannot be said that comparisons are not possible, but in our opinion, they would not be fair since they carry over the biases of the dataset's creation.

On the other hand, and perhaps the more subjective of the two factors, lies in the severe difficulty of the problem. A human being can visualize an intersection and understand that he is at one when driving. However, if we ask about the type of intersection, s/he may not even be able to classify it in some cases. The construction of cities, especially older ones such as those in Europe, does not always follow a clear geometrical pattern, and of course, until recently, no thought was given to vehicles when urbanizing them. The result is that intersections have practically infinite possibilities of classification, not so much by the number of streets that converge but by the angle of convergence between them. In addition, the environment in cities is highly changeable, generally due to the vehicles themselves and parking places, which in many cases can make it very difficult not only to classify but also to stop due to the lack of "vision" that this causes. Combining these two factors makes it very difficult to compare research papers without incorporating any bias into the research, so it becomes essential to create a proprietary data set that allows all research to start from the same place. In addition, this could attract research in this particular field, which, as presented in Chapter 1, is of critical importance.

Against this background, as a final qualitative conclusion to the results, we can say that the results obtained throughout the in-depth study of intersection classification are considered to be quite enlightening. It can be considered that the dataset is much more critical than the methodology, obviously without forgetting about it. It is necessary that the dataset has enough information, number of images, and that it is as complete as possible, enough angle of view of the camera so that the results when classifying intersections are as relevant as possible to be included in a system.

A major conclusion of this research is that this is a field in which there is still much research to be done before it can be used in an autonomous vehicle, but it is still of crucial importance.

Chapter 5

Conclusions and Future Work

The objective of this thesis was to implement, using deep learning, optimization, and data preprocessing techniques, environment identification systems that allow the safe navigation of an autonomous car in an urban environment. Together with a suitable navigation system, these techniques could radically reduce the percentage of accidents and fatalities that occur due to collisions between vehicles, run-overs, and off-roads. In order to achieve the proposed objectives, the following techniques have been used:

1. Semantic segmentation
   (a) Point cloud data transformation.
   (b) Data integration through the creation of a new network architecture.
2. Network optimization
   (a) Feature map visual inspection.
   (b) Feature map analysis through connected graphs.
3. Classification
   (a) Metric Learning.
   (b) Teacher/Student training paradigm.
   (c) Temporal integration.
   (d) Data transformation through three-dimensional and RGB data fusion.

5.1 Main Contributions

The main contributions of this thesis are as follows:

1. Creation of a specific network architecture for semantic segmentation that allows the integration of data from different sources.
2. Creation of a methodology for the projection of three-dimensional points on 2d images for road segmentation.

3. Creation of a feature map analysis method for use in network optimization.
4. Analysis of different methodologies for improving classification by deep learning in the face of high difficulty problems.
5. Analysis of the usefulness of input data pre-processing when classifying intersections using deep learning.
6. Preliminary study of the usefulness of synthesized images created using a GAN network for solving the problem of classifying intersections in the field of intersection classification.

5.2 Future work

Despite the good results obtained for each of the contributions during the research process required for this doctoral thesis, there are still many possibilities for future research, some of which are listed below.

1. A deeper and more accurate investigation of the optimization techniques proposed in the proof of concept in section 4 to validate or refute the results in a consistent manner.
2. Study of the possibility of using new architectures, such as the BiSeNet upgrade, BiSeNet v2 [Yu et al., 2020], for the upgrade of the 3D-Deep semantic segmentation network architecture proposed in this thesis.
3. Creation and release of a dataset of intersections in which the images have a sufficient viewing angle to have as much information as possible of what is in front of the ego-vehicle.
4. Implementation of a GAN-type system for generating synthesized images similar to those used for intersection classification to allow a much more robust training of any classification system. These images will also be included in the previously proposed dataset in [1].

[1] https://invett.aut.uah.es/intersectiondataset

Appendices

Appendix A

Tables of results of the different training processes

Table A.1: 3D-Deep trainin runs. (1/2)

Resnet Model	Batch Size	Momentum	Optimizer	Learning Rate	Scheduler	Kfold	F1 Average	F1 Maximum
18	1	0,99	ASGD	0,02	None	3	96,16	96,51
34							96,58	97,25
50							95,68	96,84
101							96,70	97,13
152							96,80	97,06
18	4	0,9	ASGD	0,02	None	3	96,22	96,81
34							96,66	96,78
50							97,33	97,63
101							96,88	97,05
152							96,91	97,24
18	4	0,9	ASGD	N/A	Triangular (0.25-0.0001)	4	95,78	96,98
34							96,89	97,16
50							96,94	97,59
101							96,58	97,32
152							96,82	97,42
18	4	0,9	ASGD	N/A	Decreasing Triangular (0.25-0.0001)	4	95,69	✗
34							95,24	✗
50							✗	✗
101							✗	✗
152							✗	✗

Set of runs for hyperparameter optimization in 3D-Deep architecture with panoramic view imaging

Table A.2: 3D-Deep trainin runs. (2/2)

Resnet Model	Batch Size	Momentum	Optimizer	Learning Rate	Scheduler	Kfold	F1 Average	F1 Maximum
18							96,65	97,18
34							96,70	97,01
50	4	0,9	ASGD	0,25	Polynomial	4	96,81	97,24
101							96,65	97,37
152							96,87	97,30
18							97,05	97,18
34							97,19	97,33
50	4	0,9	ASGD	0,125	Polynomial	4	97,37	97,66
101							97,14	97,65
152							97,18	97,45
18							96,45	97,16
34							97,33	97,77
50	4	0,9	ASGD	0.0625	Polynomial	4	96,95	97,40
101							96,84	97,68
152							97,29	97,68
18							96,80	97,12
34							96,93	97.00
50	4	0,9	ASGD	0.03125	Polynomial	4	96.76	97.21
101							96.60	97.04
152							97.11	97.35

Set of runs for hyperparameter optimization in 3D-Deep architecture with panoramic view imaging

Table A.3: 3D-Deep trainin runs BEV. (1/2)

Resnet Model	Batch Size	Momentum	Optimizer	Learning Rate	Scheduler	Kfold	F1 Average	F1 Maximum
18							96,64	97,00
34							96,21	97,13
50	4	0,9	ASGD	0,25	Polynomial	4	96,39	97,02
101							97,03	97,49
152							96,14	96,66
18							96,47	97,23
34							96,88	97,52
50	4	0,9	ASGD	0,125	Polynomial	4	96,12	96,87
101							96,23	97,12
152							96,72	97,46
18							96,02	97,33
34							96,53	96,95
50	4	0,9	ASGD	0.0625	Polynomial	4	96,28	96,57
101							96,71	97,23
152							96,76	97,85
18							95,88	97.06
34							96,66	97.21
50	4	0,9	ASGD	0.03125	Polynomial	4	96.41	96.80
101							96.81	97.08
152							96.34	97.20

Set of runs for hyperparameter optimization in 3D-Deep architecture with Birds eye view imaging

Table A.4: 3D-Deep trainin runs BEV. (2/2)

Resnet Model	Batch Size	Momentum	Optimizer	Learning Rate	Scheduler	Kfold	F1 Average	F1 Maximum
18							95,84	97,48
34							96,44	96,85
50	4	0,9	ASGD	0,25	Polynomial	4	95,45	96,95
101							95,88	97,03
152							97,18	97,98
18							97,02	97,45
34							96,21	96,51
50	4	0,9	ASGD	0,125	Polynomial	4	97,97	97,55
101							97,29	97,77
152							97,33	97,50
18							96,95	97,24
34							97,13	97,63
50	4	0,9	ASGD	0.0625	Polynomial	4	96,71	97,11
101							97,07	97,33
152							97,26	97,66
18							96,85	97.48
34							96.76	97.15
50	4	0,9	ASGD	0.03125	Polynomial	4	97.24	97.92
101							97.34	97.66
152							97.16	97.69
101	4	0.9	ASGD	0.03125	Polynomial	10	97.09	97.85

Set of runs for hyperparameter optimization in 3D-Deep architecture with Birds Eye view imaging

Appendix B

Publications Derived from this PhD Dissertation

B.1 Journal Publications

2021 **CAPformer: Pedestrian Crossing Action Prediction Using Transformer**, *Javier Lorenzo, Ignacio Parra Alonso, Ruben Izquierdo, Augusto Luis Ballardini, Álvaro Hernández Saz, David Fernández Llorca, Miguel Ángel Sotelo*, Sensors (ISSN: 1424-8220), DOI: 10.3390/s21175694.

2021 ***Urban Intersection Classification: A Comparative Analysis**, *Augusto Luis Ballardini, Álvaro Hernández Saz, Sandra Carrasco Limeros, Javier Lorenzo Díaz, Ignacio Parra Alonso,Noelia Hernández Parra, Iván Garcia Daza, Miguel Ángel Sotelo*, Sensors (ISSN: 1424-8220), DOI: 10.3390/s21186269.

B.2 Conference Publications

2020 **License Plate Corners Localization Using CNN-Based Regression**, *David Fernández Llorca; Ignacio Parra Alonso; Héctor Corrales Sánchez; Mónica Rentero Alonso de Linaje; Rubén Izquierdo Gonzalo; Álvaro Hernández Saz; Iván García Daza.*, Lecture Notes in Computer Science (ISSN: 0302-9743), DOI: 10.1007/978-3-030-45096-0_14.

2020 ***3D-DEEP: 3-Dimensional Deep-learning based on elevation patterns for road scene interpretation**, *Álvaro Hernández Saz; S. Woo; Héctor Corrales Sánchez; Ignacio Parra Alonso; E. Kim; David Fernández Llorca; Miguel Ángel Sotelo Vázquez.*, IEEE Intelligent Vehicles Symposium (IV-IEEE 2020), Las Vegas (EEUU).

2021 ***Model Guided Road Intersection Classification**, *Augusto Luis Ballardini, Álvaro Hernández Saz, Miguel Ángel Sotelo*, IEEE Intelligent Vehicles Symposium (IV-IEEE 2021), Nagoya (Japan).

B.3 Future Publications

2022 ***Multi-Season Multi-weather Road Intersection Dataset**, *Augusto Luis Ballardini, Álvaro Hernández Saz, Sandra Carrasco Limeros, Miguel Ángel Sotelo*, writing in progress.

*Publications directly related to this thesis.

Bibliography

[Administration, 2019] N. H. T. S. Administration, "Fatality and injury reporting system tool of u.s. national highway traffic safety administration". https://cdan.dot.gov/query (2019), accessed: 2021-05-21.

[An et al., 2016] J. An, B. Choi, K.-B. Sim and E. Kim, "Novel intersection type recognition for autonomous vehicles using a multi-layer laser scanner". *Sensors*, volume 16(7), page 1123 (2016).

[Badrinarayanan et al., 2017] V. Badrinarayanan, A. Kendall and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation". *IEEE transactions on pattern analysis and machine intelligence*, volume 39(12), pages 2481–2495 (2017).

[Ballardini et al., 2017] A. L. Ballardini, D. Cattaneo, S. Fontana and D. G. Sorrenti, "An online probabilistic road intersection detector". In *IEEE International Conference on Robotics and Automation (ICRA)* (2017).

[Ballardini et al., 2018] A. L. Ballardini, D. Cattaneo and D. G. Sorrenti, "A dataset for benchmarking vision-based localization at intersections". *arXiv preprint arXiv:1811.01306* (2018).

[Ballardini et al., 2019] A. L. Ballardini, D. Cattaneo and D. G. Sorrenti, "Visual localization at intersections with digital maps". In *IEEE International Conference on Robotics and Automation (ICRA)* (2019).

[Ballardini et al., 2021] A. L. Ballardini, Álvaro Hernández and M. Ángel Sotelo, "Model guided road intersection classification" (2021).

[Barrios et al., 2001] D. Barrios, D. Manrique, M. R. Plaza and J. Ríos, "An algebraic model for generating and adapting neural networks by means of optimization methods". *Annals of Mathematics and Artificial Intelligence*, volume 33(1), pages 93–111 (2001).

[Baumann et al., 2018] U. Baumann, Y.-Y. Huang, C. Gläser, M. Herman, H. Banzhaf and J. M. Zöllner, "Classifying road intersections using transfer-learning on a deep neural network". In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 683–690, IEEE (2018).

[Bellet et al., 2014] A. Bellet, A. Habrard and M. Sebban, "A survey on metric learning for feature vectors and structured data" (2014).

[Bernard Goldbach, 2008] Bernard Goldbach, "Roy batty, blade runner" (2008), [Online; accessed 03-February-2021].
https://commons.wikimedia.org/wiki/File:Roy_Batty.jpg

[Bhatt et al., 2017] D. Bhatt, D. Sodhi, A. Pal, V. Balasubramanian and M. Krishna, "Have i reached the intersection: A deep learning-based approach for intersection detection from monocular cameras". In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4495–4500, IEEE (2017).

[Biewald, 2020] L. Biewald, "Experiment tracking with weights and biases" (2020), software available from wandb.com.
https://www.wandb.com/

[Bond, 2019] S. Bond, "Amazon introduces computer vision into warehouses" (2019).
https://www.ft.com/content/ce0a7828-97bd-11e9-8cfb-30c211dcd229

[Brock et al., 2021] A. Brock, S. De, S. L. Smith and K. Simonyan, "High-performance large-scale image recognition without normalization". *arXiv preprint arXiv:2102.06171* (2021).

[Caltagirone et al., 2019] L. Caltagirone, M. Bellone, L. Svensson and M. Wahde, "Lidar-camera fusion for road detection using fully cnns". *Robotics and Autonomous Systems*, volume 111, pages 125–131 (2019).

[Caltagirone et al., 2017] L. Caltagirone, S. Scheidegger, L. Svensson and M. Wahde, "Fast lidar-based road detection using fully convolutional neural networks". In *2017 IV symposium*, pages 1019–1024, IEEE (2017).

[Carreira and Zisserman, 2017] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset". In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308 (2017).

[Cattaneo et al., 2020] D. Cattaneo, M. Vaghi, S. Fontana, A. L. Ballardini and D. G. Sorrenti, "Global visual localization in lidar-maps through shared 2d-3d embedding space". In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4365–4371 (2020).

[Chen et al., 2014a] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs". *arXiv preprint arXiv:1412.7062* (2014a).

[Chen et al., 2017] L.-C. Chen, G. Papandreou, F. Schroff and H. Adam, "Rethinking atrous convolution for semantic image segmentation". *arXiv preprint arXiv:1706.05587* (2017).

[Chen et al., 2014b] Y. Chen, Z. Lin, X. Zhao, G. Wang and Y. Gu, "Deep learning-based classification of hyperspectral data". *IEEE Journal of Selected topics in applied earth observations and remote sensing*, volume 7(6), pages 2094–2107 (2014b).

[Chen et al., 2019] Z. Chen, J. Zhang and D. Tao, "Progressive lidar adaptation for road detection". *IEEE/CAA Journal of Automatica Sinica*, volume 6(3), pages 693–702 (2019).

[Chong, 2018] Z. Chong, "Ai helps grow 6 billion roaches at china's largest breeding site" (2018).
https://www.cnet.com/news/ai-helps-grow-6b-roaches-at-chinas-largest-breeding-facility/?utm_source=fark&utm_medium=website&utm_content=link&ICID=ref_fark

[Clevert et al., 2015] D.-A. Clevert, T. Unterthiner and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)". *arXiv preprint arXiv:1511.07289* (2015).

[Commision, 2020] E. Commision, "Road safety key figures 2020" (2020).
https://ec.europa.eu/transport/road_safety/sites/roadsafety/files/pdf/scoreboard_2020.pdf

[Commission, 2018] E. Commission, "European union annual accident report 2018". https://ec.europa.eu/transport/road_safety/specialist/observatory/statistics/annual_accident_report_archive_en (2018), accessed: 2021-05-21.

[Commons, 2015] W. Commons, "Aphex34, cc by-sa 4.0" (2015).
https://commons.wikimedia.org/wiki/File:Typical_cnn.png

[Cordts et al., 2016] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth and B. Schiele, "The cityscapes dataset for semantic urban scene understanding". In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[Couprie et al., 2013] C. Couprie, C. Farabet, L. Najman and Y. LeCun, "Indoor semantic segmentation using depth information". *arXiv preprint arXiv:1301.3572* (2013).

[Dai et al., 2017] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu and Y. Wei, "Deformable convolutional networks". In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2017).

[Damen et al., 2020] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price and M. Wray, "The epic-kitchens dataset: Collection, challenges and baselines". *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2020).

[de la Hoz Galiana, 2020] D. de la Hoz Galiana, "Construcción de arquitecturas neuronales profundas utilizando programación genética guiada por gramáticas y algoritmos de estimación de distribución" (2020).
http://oa.upm.es/63690/

[Deng et al., 2019] L. Deng, M. Yang, H. Li, T. Li, B. Hu and C. Wang, "Restricted deformable convolution-based road scene semantic segmentation using surround view cameras". *IEEE Transactions on Intelligent Transportation Systems*, volume 21(10), pages 4350–4362 (2019).

[DGT, 2017] DGT, "Las principales cifras de siniestralidad vial en zona urbana, 2017" (2017).
https://www.dgt.es/Galerias/seguridad-vial/estadisticas-e-indicadores/publicaciones/principales-cifras-siniestralidad/Principales_Cifras_2017_Urbana.pdf

[DGT, 2019] DGT, "Las principales cifras de siniestralidad vial, 2019" (2019).
https://www.dgt.es/Galerias/seguridad-vial/estadisticas-e-indicadores/publicaciones/principales-cifras-siniestralidad/Las_principales_cifras_de_la_siniestralidad_vial_Espana_2019.pdf

[Everingham et al., 2010] M. Everingham, L. Van Gool, C. K. Williams, J. Winn and A. Zisserman, "The pascal visual object classes (voc) challenge". *International journal of computer vision*, volume 88(2), pages 303–338 (2010).

[Fan et al., 2020] H. Fan, Y. Li, B. Xiong, W.-Y. Lo and C. Feichtenhofer, "Pyslowfast". https://github.com/facebookresearch/slowfast (2020).

[Fan et al., 2018] R. Fan, M. J. Bocus and N. Dahnoun, "A novel disparity transformation algorithm for road segmentation". *Information Processing Letters*, volume 140, pages 18–24 (2018).

[Feichtenhofer, 2020] C. Feichtenhofer, "X3d: Expanding architectures for efficient video recognition". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 203–213 (2020).

[Felzenszwalb and Huttenlocher, 2004] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation". *International journal of computer vision*, volume 59(2), pages 167–181 (2004).

[Forbes, 2019] Forbes, "How artificial intelligence is transforming digital marketing" (2019).
https://www.forbes.com/sites/forbesagencycouncil/2019/08/21/how-artificial-intelligence-is-transforming-digital-marketing/?sh=78876bb21e1b

[Fritsch et al., 2013] J. Fritsch, T. Kuehnl and A. Geiger, "A new performance measure and evaluation benchmark for road detection algorithms". In *International Conference on Intelligent Transportation Systems (ITSC)* (2013).

[Gal et al., 2021] R. Gal, D. Cohen, A. Bermano and D. Cohen-Or, "Swagan: A style-based wavelet-driven generative model" (2021).

[Gamboa, 2017] J. C. B. Gamboa, "Deep learning for time-series analysis". *arXiv preprint arXiv:1701.01887* (2017).

[Geiger, 2013] A. Geiger, *Probabilistic models for 3D urban scene understanding from movable platforms*, volume 25. KIT Scientific Publishing (2013).

[Geiger et al., 2012] A. Geiger, P. Lenz and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite". In *Conference on Computer Vision and Pattern Recognition (CVPR)* (2012).

[Glorot et al., 2011] X. Glorot, A. Bordes and Y. Bengio, "Deep sparse rectifier neural networks". In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323 (2011).

[Goodfellow et al., 2014] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative adversarial networks". *arXiv preprint arXiv:1406.2661* (2014).

[Goyal et al., 2017] R. Goyal, S. E. Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag, F. Hoppe, C. Thurau, I. Bax and R. Memisevic, "The "something something" video database for learning and evaluating visual common sense". In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5843–5851 (2017).

[Guillaume Chevalier, 2018] Guillaume Chevalier, "The lstm cell" (2018), [Online; accessed 22-March-2021].
hhttps://commons.wikimedia.org/wiki/File:The_LSTM_Cell.svg

[Gupta et al., 2014] S. Gupta, R. Girshick, P. Arbeláez and J. Malik, "Learning rich features from rgb-d images for object detection and segmentation". In *European conference on computer vision*, pages 345–360, Springer (2014).

[Habermann et al., 2016] D. Habermann, C. E. Vido, F. S. Osório and F. Ramos, "Road junction detection from 3d point clouds". In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 4934–4940, IEEE (2016).

[He et al., 2019] H. He, D. Yang, S. Wang, S. Wang and X. Liu, "Road segmentation of cross-modal remote sensing images using deep segmentation network and transfer learning". *Industrial Robot: the international journal of robotics research and application* (2019).

[He et al., 2017] K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask r-cnn". In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969 (2017).

[He et al., 2016] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition". In *Proceedings of the IEEE CVPR*, pages 770–778 (2016).

[Henry et al., 2018] C. Henry, S. M. Azimi and N. Merkle, "Road segmentation in sar satellite images with deep fully convolutional neural networks". *IEEE Geoscience and Remote Sensing Letters*, volume 15(12), pages 1867–1871 (2018).

[Hernández et al., 2020] A. Hernández, S. Woo, H. Corrales, I. Parra, E. Kim, D. F. Llorca and M. A. Sotelo, "3d-deep: 3-dimensional deep-learning based on elevation patterns for road scene interpretation". In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 892–898 (2020).

[Hochreiter and Schmidhuber, 1997] S. Hochreiter and J. Schmidhuber, "Long short-term memory". *Neural computation*, volume 9(8), pages 1735–1780 (1997).

[Hornik et al., 1989] K. Hornik, M. Stinchcombe and H. White, "Multilayer feedforward networks are universal approximators". *Neural networks*, volume 2(5), pages 359–366 (1989).

[Howard et al., 2019] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan et al., "Searching for mobilenetv3". In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324 (2019).

[Hu et al., 2018] J. Hu, L. Shen and G. Sun, "Squeeze-and-excitation networks". In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141 (2018).

[Iglovikov and Shvets, 2018] V. Iglovikov and A. Shvets, "Ternausnet: U-net with vgg11 encoder pre-trained on imagenet for image segmentation" (2018).

[International, 2014] S. International, "Automated driving levels of driving automation are defined in new sae international standard j3016". https://www.sae.org/standards/content/j3016_202104/ (2014).

[Ioffe and Szegedy, 2015] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift". *arXiv preprint arXiv:1502.03167* (2015).

[Kagaya and Aizawa, 2015] H. Kagaya and K. Aizawa, "Highly accurate food/non-food image classification based on a deep convolutional neural network". In *International conference on image analysis and processing*, pages 350–357, Springer (2015).

[Karras et al., 2020a] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen and T. Aila, "Training generative adversarial networks with limited data". In *Proc. NeurIPS* (2020a).

[Karras et al., 2020b] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen and T. Aila, "Training generative adversarial networks with limited data" (2020b).

[Kay et al., 2017] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev et al., "The kinetics human action video dataset". *arXiv preprint arXiv:1705.06950* (2017).

[Kelion, 2019] L. Kelion, "Deepmind ai achieves grandmaster status at starcraft 2" (2019). https://www.bbc.com/news/technology-50212841

[Kim, 2014] Y. Kim, "Convolutional neural networks for sentence classification" (2014).

[Koji and Kanji, 2019] T. Koji and T. Kanji, "Deep intersection classification using first and third person views". In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 454–459, IEEE (2019).

[Krizhevsky et al., 2012] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks". In *Advances in Neural Information Processing Systems*, volume 2, pages 1097–1105 (2012), cited By :43723.

[Kuo and Tsai, 2021] C.-L. Kuo and M.-H. Tsai, "Road characteristics detection based on joint convolutional neural networks with adaptive squares". *ISPRS International Journal of Geo-Information*, volume 10(6), page 377 (2021).

[Kushner and Puri, 1987] T. R. Kushner and S. Puri, "Progress in road intersection detection for autonomous vehicle navigation". In *Mobile Robots II*, volume 852, pages 19–24, International Society for Optics and Photonics (1987).

[Kussul et al., 2017] N. Kussul, M. Lavreniuk, S. Skakun and A. Shelestov, "Deep learning classification of land cover and crop types using remote sensing data". *IEEE Geoscience and Remote Sensing Letters*, volume 14(5), pages 778–782 (2017).

[Leviathan and Matias, 2018] Y. Leviathan and Y. Matias, "Google duplex: an ai system for accomplishing real-world tasks over the phone" (2018).

[Lin et al., 2017a] G. Lin, A. Milan, C. Shen and I. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation". In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1925–1934 (2017a).

[Lin et al., 2017b] T.-Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, "Focal loss for dense object detection". In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988 (2017b).

[Liu et al., 2020a] H. Liu, Y. Yao, Z. Sun, X. Li, K. Jia and Z. Tang, "Road segmentation with image-lidar data fusion in deep neural network". *Multimedia Tools and Applications*, volume 79(47), pages 35503–35518 (2020a).

[Liu et al., 2020b] S. Liu, H. Zhang, L. Shao and J. Yang, "Built-in depth-semantic coupled encoding for scene parsing, vehicle detection and road segmentation". *IEEE Transactions on Intelligent Transportation Systems* (2020b).

[Liu et al., 2015] W. Liu, A. Rabinovich and A. C. Berg, "Parsenet: Looking wider to see better". *arXiv preprint arXiv:1506.04579* (2015).

[Long et al., 2015] J. Long, E. Shelhamer and T. Darrell, "Fully convolutional networks for semantic segmentation". In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440 (2015).

[Lyu et al., 2018] Y. Lyu, L. Bai and X. Huang, "Chipnet: Real-time lidar processing for drivable region segmentation on an fpga". *IEEE Transactions on Circuits and Systems I: Regular Papers*, pages 1769–1779 (2018).

[Min et al., 2014] D. Min, S. Choi, J. Lu, B. Ham, K. Sohn and M. N. Do, "Fast global image smoothing based on weighted least squares". *IEEE Transactions on Image Processing*, volume 23(12), pages 5638–5653 (2014).

[Mnih, 2013] V. Mnih, *Machine Learning for Aerial Image Labeling*. Ph.D. thesis, University of Toronto (2013).

[Mohsen et al., 2018] H. Mohsen, E.-S. A. El-Dahshan, E.-S. M. El-Horbaty and A.-B. M. Salem, "Classification using deep learning neural networks for brain tumors". *Future Computing and Informatics Journal*, volume 3(1), pages 68–71 (2018).

[Musgrave et al., 2020a] K. Musgrave, S. Belongie and S.-N. Lim, "A metric learning reality check". In *European Conference on Computer Vision*, pages 681–699, Springer (2020a).

[Musgrave et al., 2020b] K. Musgrave, S. Belongie and S.-N. Lim, "Pytorch metric learning" (2020b).

[Muñoz-Bulnes et al., 2017] J. Muñoz-Bulnes, C. Fernandez, I. Parra, D. Fernández-Llorca and M. A. Sotelo, "Deep fully convolutional networks with random data augmentation for enhanced generalization in road detection". In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 366–371 (2017).

[Nakazawa and Kulkarni, 2018] T. Nakazawa and D. V. Kulkarni, "Wafer map defect pattern classification and image retrieval using convolutional neural network". *IEEE Transactions on Semiconductor Manufacturing*, volume 31(2), pages 309–314 (2018).

[Nordin, 2018] M. Nordin, "Teaching ai-agents to play battlefield" (2018). https://www.ea.com/en-au/news/teaching-ai-agents-battlefield-1

[NVIDIA, 2019] NVIDIA, "Apex". github.com/NVIDIA/apex (2019).

[Oeljeklaus et al., 2018] M. Oeljeklaus, F. Hoffmann and T. Bertram, "A fast multi-task cnn for spatial understanding of traffic scenes". In *2018 21st International Conference on ITS*, pages 2825–2830, IEEE (2018).

[Oniga et al., 2008] F. Oniga, S. Nedevschi and M. M. Meinecke, "Curb detection based on a multi-frame persistence map for urban driving scenarios". In *2008 11th International IEEE Conference on Intelligent Transportation Systems*, pages 67–72, IEEE (2008).

[Otsu, 1979] N. Otsu, "A threshold selection method from gray-level histograms". *IEEE transactions on systems, man, and cybernetics*, volume 9(1), pages 62–66 (1979).

[Panboonyuen et al., 2017] T. Panboonyuen, K. Jitkajornwanich, S. Lawawirojwong, P. Srestasathiern and P. Vateekul, "Road segmentation of remotely-sensed images using deep convolutional neural networks with landscape metrics and conditional random fields". *Remote Sensing*, volume 9(7), page 680 (2017).

[Park et al., 2020] J. Park, D. Yi and S. Ji, "A novel learning rate schedule in optimization for neural networks and it's convergence". *Symmetry*, volume 12(4), page 660 (2020).

[Paszke et al., 2017] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga and A. Lerer, "Automatic differentiation in PyTorch". In *Advances in Neural Information Processing Systems, Autodiff Workshop* (2017).

[Polyak and Juditsky, 1992] B. T. Polyak and A. B. Juditsky, "Acceleration of stochastic approximation by averaging". *SIAM Journal on Control and Optimization*, volume 30(4), pages 838–855 (1992).

[Qi et al., 2017] C. R. Qi, H. Su, K. Mo and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation". In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660 (2017).

[Reis et al., 2019] F. A. Reis, R. Almeida, E. Kijak, S. Malinowski, S. J. F. Guimarães and Z. K. do Patrocínio, "Combining convolutional side-outputs for road image segmentation". In *2019 IJCNN*, pages 1–8, IEEE (2019).

[Ronneberger et al., 2015] O. Ronneberger, P. Fischer and T. Brox, "U-net: Convolutional networks for biomedical image segmentation". In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241, Springer (2015).

[Russakovsky et al., 2015] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein et al., "Imagenet large scale visual recognition challenge". *International journal of computer vision*, volume 115(3), pages 211–252 (2015).

[Saleem et al., 2019] M. H. Saleem, J. Potgieter and K. M. Arif, "Plant disease detection and classification by deep learning". *Plants*, volume 8(11), page 468 (2019).

[Scharwächter and Franke, 2015] T. Scharwächter and U. Franke, "Low-level fusion of color, texture and depth for robust road scene understanding". In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 599–604, IEEE (2015).

[Schroff et al., 2015] F. Schroff, D. Kalenichenko and J. Philbin, "Facenet: A unified embedding for face recognition and clustering". In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823 (2015).

[Shamsolmoali et al., 2020] P. Shamsolmoali, M. Zareapoor, H. Zhou, R. Wang and J. Yang, "Road segmentation for remote sensing images using adversarial spatial pyramid networks". *IEEE Transactions on Geoscience and Remote Sensing* (2020).

[Shelhamer et al., 2017] E. Shelhamer, J. Long and T. Darrell, "Fully convolutional networks for semantic segmentation". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 39(4), pages 640–651 (2017).

[Siegemund et al., 2010] J. Siegemund, D. Pfeiffer, U. Franke and W. Förstner, "Curb reconstruction using conditional random fields". In *2010 IEEE Intelligent Vehicles Symposium*, pages 203–210, IEEE (2010).

[Sigurdsson et al., 2016] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev and A. Gupta, "Hollywood in homes: Crowdsourcing data collection for activity understanding" (2016).

[Simonyan and Zisserman, 2014] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition". *arXiv preprint arXiv:1409.1556* (2014).

[Smith, 2017] L. N. Smith, "Cyclical learning rates for training neural networks". In *2017 IEEE WACV*, pages 464–472, IEEE (2017).

[Sobel, 2014] I. Sobel, "An isotropic 3x3 image gradient operator". *Presentation at Stanford A.I. Project 1968* (2014).

[Stephen et al., 2019] O. Stephen, M. Sain, U. J. Maduh and D.-U. Jeong, "An efficient deep learning approach to pneumonia classification in healthcare". *Journal of healthcare engineering*, volume 2019 (2019).

[Stier et al., 2018] J. Stier, G. Gianini, M. Granitzer and K. Ziegler, "Analysing neural network topologies: a game theoretic approach". *Procedia Computer Science*, volume 126, pages 234–243 (2018).

[Sturgess et al., 2009] P. Sturgess, K. Alahari, L. Ladicky and P. H. Torr, "Combining appearance and structure from motion features for road scene understanding" (2009).

[Szegedy et al., 2015] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going deeper with convolutions". In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9 (2015).

[Szegedy et al., 2016] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the inception architecture for computer vision". In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826 (2016).

[Tabian et al., 2019] I. Tabian, H. Fu and Z. Sharif Khodaei, "A convolutional neural network for impact detection and characterization of complex composite structures". *Sensors*, volume 19(22), page 4933 (2019).

[Tan and Le, 2019a] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks". In *International Conference on Machine Learning*, pages 6105–6114, PMLR (2019a).

[Tan and Le, 2019b] M. Tan and Q. V. Le, "Mixconv: Mixed depthwise convolutional kernels". *arXiv preprint arXiv:1907.09595* (2019b).

[Tümen and Ergen, 2020] V. Tümen and B. Ergen, "Intersections and crosswalk detection using deep learning and image processing techniques". *Physica A: Statistical Mechanics and its Applications*, volume 543, page 123510 (2020).

[Wang et al., 2019a] K. Wang, F. Yan, B. Zou, L. Tang, Q. Yuan and C. Lv, "Occlusion-free road segmentation leveraging semantics for autonomous vehicles". *Sensors*, volume 19(21), page 4711 (2019a).

[Wang et al., 2019b] P. Wang, X. Huang, X. Cheng, D. Zhou, Q. Geng and R. Yang, "The apolloscape open dataset for autonomous driving and its application". *IEEE transactions on pattern analysis and machine intelligence* (2019b).

[Wei, 2015] X.-S. Wei, "Must know tips/tricks in deep neural networks". *URL http://lamda. nju. edu. cn/weixs/project/CNNTricks/CNNTricks. html* (2015).

[Woo et al., 2018] S. Woo, J. Park, J.-Y. Lee and I. S. Kweon, "Cbam: Convolutional block attention module". In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19 (2018).

[Wu et al., 2015] R. Wu, S. Yan, Y. Shan, Q. Dang and G. Sun, "Deep image: Scaling up image recognition". *arXiv preprint arXiv:1501.02876*, volume 7(8) (2015).

[xenonstack, 2019] xenonstack, "Generative adversarial networks" (2019), [Online; accessed March 22, 2021].
https://www.xenonstack.com/images/insights/2019/12/generative-adversarial-networks-applications-xenonstack.png

[Xie et al., 2016] J. Xie, M. Kiefel, M.-T. Sun and A. Geiger, "Semantic instance annotation of street scenes by 3d to 2d label transfer". In *Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[Xie and Tu, 2015] S. Xie and Z. Tu, "Holistically-nested edge detection". In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403 (2015).

[Xu and Zhang, 2020] H. Xu and J. Zhang, "Aanet: Adaptive aggregation network for efficient stereo matching". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1959–1968 (2020).

[Xu et al., 2016] P. Xu, F. Davoine, J.-B. Bordes, H. Zhao and T. Denœux, "Multimodal information fusion for urban scene understanding". *Machine Vision and Applications*, volume 27(3), pages 331–349 (2016).

[Xu and Liang, 2001] Q.-S. Xu and Y.-Z. Liang, "Monte carlo cross validation". *Chemometrics and Intelligent Laboratory Systems*, volume 56, pages 1–11 (2001).

[Yan et al., 2020] F. Yan, K. Wang, B. Zou, L. Tang, W. Li and C. Lv, "Lidar-based multi-task road perception network for autonomous vehicles". *IEEE Access*, volume 8, pages 86753–86764 (2020).

[Yu et al., 2020] C. Yu, C. Gao, J. Wang, G. Yu, C. Shen and N. Sang, "Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation". *arXiv preprint arXiv:2004.02147* (2020).

[Yu et al., 2018] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu and N. Sang, "Bisenet: Bilateral segmentation network for real-time semantic segmentation". In *Proceedings of the European conference on computer vision (ECCV)*, pages 325–341 (2018).

[Yu and Koltun, 2015] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions". *arXiv preprint arXiv:1511.07122* (2015).

[Zeiler et al., 2010] M. D. Zeiler, D. Krishnan, G. W. Taylor and R. Fergus, "Deconvolutional networks". In *2010 IEEE Computer Society Conference on computer vision and pattern recognition*, pages 2528–2535, IEEE (2010).

[Zhang et al., 2018] Y. Zhang, H. Chen, Y. He, M. Ye, X. Cai and D. Zhang, "Road segmentation for all-day outdoor robot navigation". *Neurocomputing*, volume 314, pages 316–325 (2018).

[Zhao et al., 2017] H. Zhao, J. Shi, X. Qi, X. Wang and J. Jia, "Pyramid scene parsing network". In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890 (2017).