Co-funded by the
Erasmus+ Programme
of the European Union

ERASMUS MUNDUS MASTER IN
IMAGE PROCESSING AND COMPUTER VISION

PÁZMÁNY PÉTER
CATHOLIC UNIVERSITY

université de BORDEAUX

UAM
Universidad Autónoma
de Madrid

MSc THESIS

Deep learning-based LiDAR Point-cloud Semantic Segmentation using 2D Range Image

PRESENTED AT

UNIVERSITE DE BORDEAUX

université de BORDEAUX

Author: BRAVO SOLIS, Eduardo Daniel

Academic Supervisors: CLÉMENT Michaël, GIRAUD Rémi

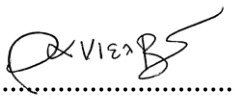
DATE: June, 2020

Declaration of Authenticity

(to be inserted as second page of the thesis)

I, the undersigned Eduardo Daniel Bravo Solis, student of the Image Processing and Computer Vision program, hereby certify that this thesis has been written without any unauthorized help, solely by me, using only the referenced sources. Every part quoted in whole or in part is indicated clearly with a reference.

I declare that I have not submitted this thesis in any other higher education institution, excluding the partner universities of the IPCV Consortium.



.....

Student

Contents

1 Abstract

Semantic segmentation for point clouds is a topic that is thriving strongly in recent years due to the constant release of new algorithms and datasets. In this research we studied the network called LU-Net for lidar segmentation that uses range image representation of the point cloud to perform a CNN-like learning. For the development of this network, the authors used an adaptation of the KITTI dataset for lidar segmentation that has a reduced field of view of the cloud and works with four different classes.

We adapted this network to work on SemanticKITTI, a more recent benchmark that offers a full segmentation of the point cloud using 20 classes for static analysis. Since this dataset is actually an extension of the odometry scans of KITTI, its compatibility and substantiality are making it popular among the research community.

Throughout this research we tested different approaches to improve scores obtained with the original LU-Net by modifying different aspects of the original configuration. The most relevant updates are (i) Data augmentation for lidar and range image. We perform flips on the x-axis on 50 % of the scans to create more data for training. (ii) Redefinition of loss function. Considering the metric that is used in segmentation we included de Dice Loss into the LU-Net original error function to orientate it to the optimization of mean Intersection Over Union. (iii) New 3D feature extraction module architecture. Original LU-Net uses multilayer perceptron (MLP) for this purpose, so we update it to a convolutional neural network (CNN) to exploit the range image representation.

As indirect findings, we provide parameter values that are more proficient for the new dataset such as learning rate, number of layers and kernel sizes in convolutional operations.

With all these updates, we managed to increase the mean intersection over union (mIoU) by around 10%, scoring high on problematic classes such as bicyclists and trunk. This model competes with state-of-the-art scores using a smaller network and reduced Range Image dimensions.

This was planned to be an exploratory research. In the starting point of the internship, there were no preestablished intermediate goals and the intern along with the supervisors would set the activities based on results every iteration, making the exploratory process flexible and dynamic. In addition, it was a didactive study since the intern was able to explore ideas by design, coding and testing architectures.

2 Internship

2.1 LaBRI

The host company is the computer science research lab of Bordeaux (LaBRI by its initials in French: Laboratoire Bordelais de Recherche en Informatique). It is a research unit associated with the University of Bordeaux, and with the National school of electronics, informatics, telecommunication, mathematics and mechanics of Bordeaux (ENSEIRB by its initials in French: École Nationale Supérieure d'Électronique, Informatique, Télécommunications, Mathématique et Mécanique de Bordeaux). It has significantly increased in staff numbers over recent years as well as the PhD students and research interns. The LaBRI's missions are of three types: research (pure and applied), technology application and transfer and training.

Figure 1: Logo of the informatics research lab of Bordeaux (LaBRI).

The laboratory is located in the community of Talance, in the South-West region of Bordeaux (figure 2), France. Address is: Domaine universitaire, 351, cours de la Libération, 33405 Talence (figure 3) inside the campus of the University of Bordeaux. In less than one-kilometer perimeter there are three university restaurants/cafeterias that are open for students and University staff. In this area several buildings of the university reside that come handy for interns, such as administrative offices and languages faculty. In a three-kilometer ratio there are streets with restaurants and stores of different kind, making the location of LaBRI remarkable.

Figure 2: LaBRI location in the Bordeaux area

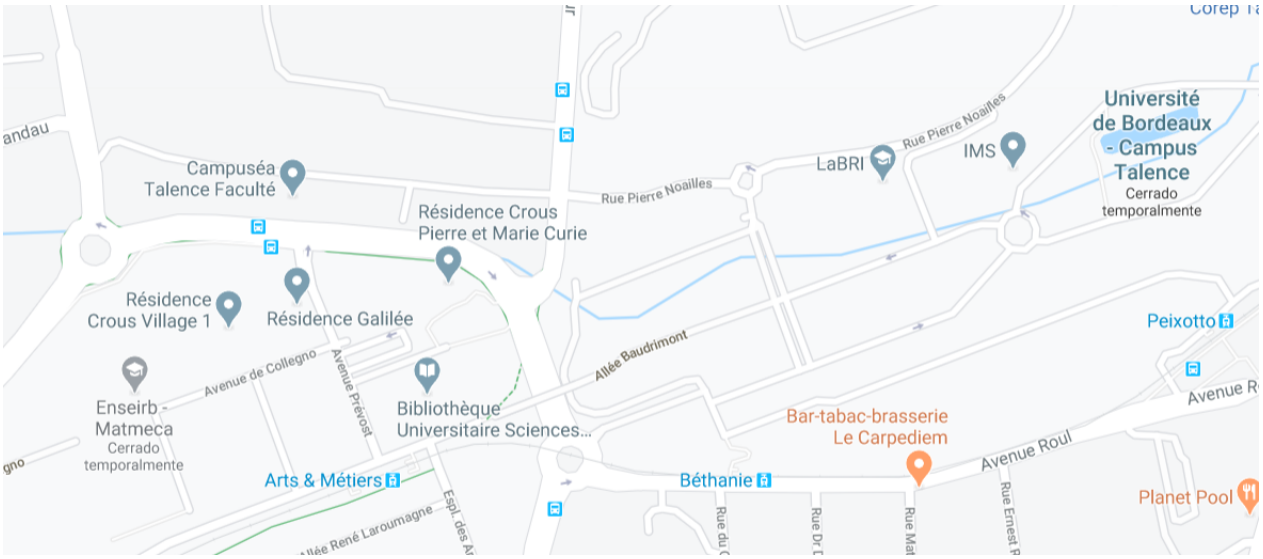


Figure 3: LaBRI in the the university of Bordeaux campus Talance

The lab receives international interns and researches for long and short periods having a mix of French people and foreigners, creating a multicultural but yet respectful atmosphere. There were conferences, gatherings, presentations and social activities frequently to favor interaction between the workers. Sometimes all this was orientated to everyone in the building and in other times, they were organized specially for closed groups. The installation also holds outsiders for courses and activities of different kind commonly when the topic is related to computer science or the person hosting the activity works at LaBRI.

Figure 4: Facade of LaBRI building

The length of the internship contract is of six months and during the first two months I shared

an office with another researcher with on extra member joining two months after and another one by the end of the semester. The office was located in the ground floor of the building, but I also had access to all the common areas, kitchen and coffee machine.

2.2 Home Office

During the internship period, after two months of working in the office in LaBRI building, the laboratory needed to close due to a health situation that forced companies in France to temporally close and ask all employees to work from home. Considering the type of work that I was performing, adapting to the new way of working was not very challenging, and I only needed to get used to use the remote tools of LaBRI from home. Hardware limitations were an obstacle, considering that the computer I needed to use with this change was my personal laptop and is not suited for professional software development. Fortunately, the GPU is part of the tools that I could access remotely and there was also a large amount of data storage available in the laboratory servers.

After mid-March, my new working area was my room's desk in a students dorm nearby and from there I worked for the rest of the internship period.

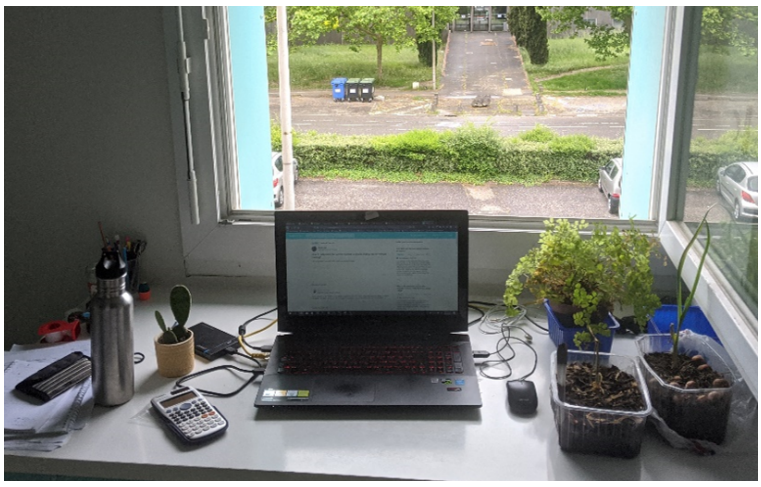


Figure 5: Home office

2.3 Internship Requirements

The idea of the internship is to do an exploratory research about point cloud segmentation, taking as starting point some of the state-of-the-art algorithms. Particularly the network LU-Net was used as baseline, a model developed by a previous PhD student of LaBRI, presented the year before to this internship. This algorithm was selected for further research since it offers promising results,

code re-usability and areas of improvement.

There is no predefined final goal or path to follow for this research. This way, the intern can plan and execute weekly goals, present results and based on them, take decisions about the next steps. All this process is always guided by the PhD supervisors with background on the field.

At the beginning of the intership only few initial goals where listed:

1. Review the state of the art on Deep learning-based point cloud segmentation algorithms.
2. Adapt LU-Net architecture to new dataset with more data and classes.
3. Explore parameters and configurations to increment the accuracy of the method.
4. Improve the network accuracy using deep learning techniques (focal loss, data augmentation, multy-modality)

Up to the publication of this document, only the first two were reviewed and presented below. This corresponds to 2/3 of the research program. The review of multi-modality is planned for the last part of the internship and will be considered for the final version of this report.

For being part of this research, the intern needed to have several characteristics to go along with the research:

- Programming background in
  - Python
  - C
  - C ++
  - Matlab
- Knowledge of Linux and command prompt
- Fluidity in Deep learning
  - Concepts
  - Architecture
  - Applications
- Preferable background in deep learning libraries such as TensorFlow or PyTorch.
- Knowledge in Image processing and computer vision
  - Basics of Image processing
  - Handcrafted methods
  - CNNs
- Understanding of sensors and data in the automotive field such as lidar, camera, GPS, etc.

Soft skills were also preferable that were not listed in the job description but can weight the performance and achievements of the intern during the research period:

- Communication skills to share ideas and suggestions during stand-up meetings.
- To be able to work in a multicultural environment
- Quick adaptability
- Fast understanding and learning
- To be able to work without supervision and results-orientated.
- Integrity, creativity and flexible with goals and workflow.

The supervision for the research internship is flexible and dynamic, having stand-up meetings every one or two weeks with both supervisors. Meetings generally include concepts discussions, code review and brain storming sessions for defining path to follow.

2.3.1 Software & Hardware Requirements

The software requirements are defined from the algorithms that were the inspiration for this research and are used as reference to start the internship. Names and details will be listed in the state-or-the-art section, while here, more specific information about libraries and version will be presented:

- Ubuntu 16 or higher
- TensorFlow 2 or Tensorflow-gpu 1.6 or higher
- CUDA 9 or higher, cuDNN 7 or higher
- CloudCompare V2 or higher
- Python 2 and libraries (OpenCv,Yaml, Numpy, Numpy, Matplotlib, Sklearn, etc.)

The user is responsible of checking the versions matching of each of the libraries and drivers, considering that there are specifications for the CUDA-Python-TensorFlow compatibility. CUDA version is usually embedded in the lower level of the system and it can be harder to modify without augmented privileges, so in the case of working in hardware provided by the laboratory, is common to have restricted access. Since this was the case for my internship, I needed to match versions of python and TensorFlow, that will be compatible with the CUDA version available and with the code of the Network as well. Matching versions and configurations can be found in the TensorFlow webpage:

`https://www.tensorflow.org/install/source#tested_build_configurations`

The use of virtual environment is recommended to make the proper configurations without requesting security privileges and to be able to work with different source codes in the same computer. Since this is an exploratory research that includes the analysis of different architectures, only the common software requirements are listed, for reproducing the state-of-the-art methods on point cloud segmentation. This are sufficient as well for the last model proposed at the end of this report. However, months after this document is submitted the team will continue working on the architecture of the network and more requirements could be needed for the final model.

The requirements in terms of hardware and workplace for this research are typical base tools to start any software development process, that enable the user to install more specifics, depending on the research:

- CPU, monitor, keyboard, mouse
- 1TB of free ROM memory (for datasets, results and code)
- Linux OS
- Internet access
- GPU with around 12GB of RAM. LaBRI uses Titan X and RTX 2080 Ti on the remote servers.

2.4 Supervisors

There are two academic supervisors for this internship, CLÉMENT, Michaël and GIRAUD, Rémi that helped out with the administrative activities as well with advising for the research. Since the internship is in an university laboratory, they both have the roles as academic and company consultant.

2.4.1 Michaël Clément

Associate professor of computer science (maître de conférences) at Bordeaux INP, in the ENSEIRB-MATMECA engineering school, and researcher at the LaBRI laboratory, in the Image and Sound team. He has a PhD in computer science by Université Paris Descartes, in France working in modeling and learning spatial relations between objects for image understanding, with applications in document analysis, remote sensing and medical imaging. After his PhD, he was a postdoctoral researcher at the Centre for Vision Research of York University, where he worked on shape data analysis for 2D and 3D reconstruction problems. He has made over six publications in national and

international conferences and actively work with several teams and interns to keep pushing forward the research in Image Processing. More info can be found in:

`https://www.labri.fr/perso/mclement/`

2.4.2 Rémi Giraud

Since 2018, he is an associate professor at ENSEIRB-MATMECA, and an engineer school member of Bordeaux INP in the Electronic department. He is also a researcher at the IMS laboratory in the Signal and Image group. Rémi has a M.Sc. degree in telecommunications by ENSEIRB-MATMECA, and the M.Sc. in signal and image processing from the University of Bordeaux. In 2017, he obtained his Ph.D. in computer science at the LaBRI and IMB labs of the University of Bordeaux. His main research areas are:

- Applications: Unsupervised segmentation, classification, image synthesis, color and style transfer
- Data: Natural and 3D medical images
- Methods: Patch-based matching algorithms, Superpixel representation and evaluation metrics, Artificial intelligence and deep learning methods

More info available in his ENSEIRB-MATMECA webpage:

`http://rgiraud.vvv.enseirb-matmeca.fr/home/`

3 Introduction

To carry out a research in applied computer vision we rely on data to understand, design and execute algorithms. This necessity has motivated institutions to produce datasets to encourage and ease reserach. Such is the case of the KITTI, a public and open-source dataset that offers information from different sensors (lidar, camera, GPS). The acquisition process is done using a car going around diverse scenarios like real-world traffic situations, freeways, rural areas and inner-city, with both dynamic and static objects [1]. Since the release of KITTI in 2013, several others have made public different resources related to autonomous driving. Some of them as independent projects [23], and others as an extension of original KITTI benchmark like SemanticKITTI [6]. There is a wide variety of analysis that can be performed in all this data that concerns the computer vision community, such as video tracking, image segmentation or object recognition. Commonly there are signals of a different nature also available in these datasets that coexist with optical flow in the sense that they represent the same environment as well, but in a different domain [1]. The one that is relevant for our research is the Light Detection and Ranging LiDAR, a light-based sensor that generates point clouds of its surroundings. This sensor is of particular interest for the research community due to its capability of capture a scene based on 4-dimension points. These dimensions are (x,y,z,r), where (x,y,z) are the 3D coordinates of the point in the scene and r is the reflectance, that can be interpreted as the intensity of the signal when acquired back to the sensor [14].

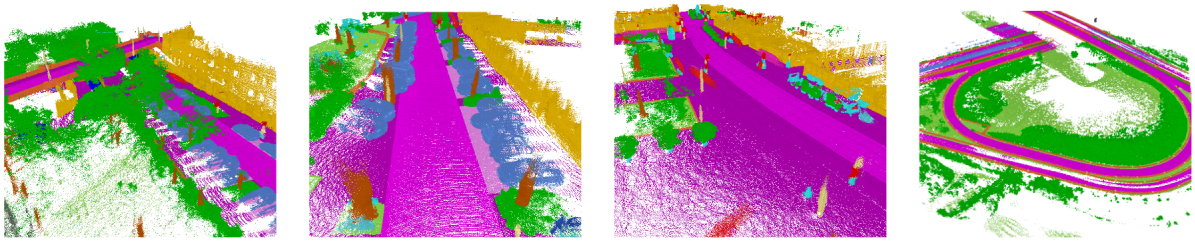


Figure 6: Full Point Cloud Segmentation by SemanticKITTI

There is a strong relation between images and point clouds, encouraging researches to close the gap between these two scene representations and be able to work in a multidomain environment. Images are related to the way humans acquire information through sight, perceiving colors and shapes of objects. Even distance can be acquired up to some extend when using binocular acquisition (as humans with two eyes) but it only works as a comparative method. LiDAR, on the other hand, provides detailed information about the distance, and more specifically, 3D coordinates to each of the points captured. This sensor also makes available the reflectance, also called remission, that

represents the intensity with what the signal was acquired back in the sensor. This can be an indicator of the objects materials but is not reliable for an accurate visual analysis. Considering that the analysis of pictures is older and it has been more studied than LiDAR signals, there are more approaches and theory about image processing compared to the later one. However, some research lines are trying to make connections between image analysis methods and other types of input signals. The one that is deeply studied in this research is the Convolutional Neural Networks.

Unlike Multilayer Perceptron, Convolutional Neural Networks (CNN) offer an analysis based on convolution, therefore it considers the spatial distribution of the input, and offers a local and global study. In addition, the number of parameters to learn is smaller due to the weights sharing in convolutional operations in addition to pooling layers, so more complex architectures can be achieved with less computational resources. The current accuracy and speed achieved by CNNs in image processing tasks, encourage research to consider them as a possible option for point cloud analysis, specially when software and hardware for deep learning has grown exponentially in the recent years.

The integration of point clouds into CNN analysis requires a preprocessing to arrange the data in image-like fashion, to match the nature of the convolution operation. The approach that is thriving is the Range-Image representation, where the point cloud is projected into a cylindrical plane based on the rotation angles of each point [19]. By doing this, the list of unordered points are arranged in a matrix with shape (hight, width, depth), where typically the height is limited by the vertical range of the sensor, width can variate based on the desired size of the image, and depth or number of channels are the features of the points projected. With this transformation some points are lost due to point occlusions and discretization, but still outputs a fair image-like representation of the point cloud that can be further analyzed with CNNs.

Just like in some image challenges, the segmentation of point cloud is of particular interest to interpret it and digitalize the elements in the scene. In other words, going from single and unrelated points to a conceptual view, where each point is interpreted as part of a class. For example, in the automotive field is common to segment point clouds with the classes: car, bikes and pedestrians[9], but also some datasets consider more challenging classes such as traffic sign, grass or fence [6]. Some benchmarks are trying to incorporate the information of lines on the street that are of particular relevance for autonomous driving but represent a higher challenge for LiDAR analysis due to the sensor nature [23].

Throughout this research the team tried to integrate algorithms developed in previous years to

the recent challenges and dataset, that are of a higher level of complexity. This demanded for a deep study in each of the elements of CNN architectures and of the task itself, elements that will be presented in the following sections. Particularly the team worked in LU-Net [9], a point cloud segmentation algorithm that works on range image representation. This algorithm was developed by a former Image Processing PhD student of LaBRI, and it was published October 2019 concluding that the network is suitable for further research and expansion. By having a small architecture and reduced number of parameters this network scored high in comparison with other reference algorithms.

This research acknowledges the new datasets that were published recently and try to integrate LU-Net into this new data. A newer benchmark offers labels for full segmentation under more than 30 classes, making it more robust and flexible for different applications.

After an exploratory process the team found approaches that gave significant improvement to the scores of the predictions, studying possible changes in architecture, parameters and input data. These are listed in the section 7.

For presenting this work, first there will be a short background with basic definitions and history of the tools and concept used. Later some of the state-of-the-art algorithms and datasets will be listed. These are the references of this work considering them as starting points. After that comes the methodology section that states how pre and post processing steps are performed, to adapt the model demands to the training and data. Then we list the proposals of the research with details and justification. Later on, we present the results and discussion showing the comparison to reference methods. And finally, future work and conclusions are offered.

4 Background

4.1 Velodyne LiDAR

Figure 7: Velodyne HDL-64E

The Light Detection and Ranging (LiDAR or lidar) is a method that measures distance to a target by illuminating that target with a pulsed laser light and measuring the reflected pulses with a sensor. Differences in laser return times and wavelengths can then be used to measure distances [14]. There is an specific lidar device that is widely used in the automotive field, and became more popular for autonomous driving applications: The Velodyne LiDAR. This model has the ability to rotate over its axis and capture the surroundings, using a vertical arrange of beams to increase the vertical field of view. The model that is used in this research is the Velodyne HDL-64E (figure 7) which is a rotating 3D laser scanner that works at 10 Hz. It uses 64 vertical beams causing a 0.09-degree angular resolution. The measured distance accuracy is up to 2 cm. With this configuration the device is able to collect around 1.3 million points/second, a number that variates due to the possible missing samples. The field of view is: 360 degrees horizontally and 26.8 degrees vertical, with a range: 120 m [1]. These characteristics are enough to generate a representation of the scene through a point cloud. Each point contains information of four channels. The first three are the coordinates of the point in the 3D world and the fourth is the measure of the intensity, also called remission. This last value only represents the intensity which the signal was captured back in the sensor but it has created an ongoing research on its own, where the impact and relevance of this parameter is studied. Some studies suggest that this value offers key information to perform more precise classifications or segmentations but this variate depending on the scene and proximity of the points [15]. Therefore is common to find different opinions on how to incorporate this parameter into algorithms without introducing noise into the system.

Velodyne data can be indirectly used to complement and test algorithms of different nature. One example is the work on Collision Risk Estimation, a method for measure rough distance using images [25]. The accuracy of this method was only proven with the use of Velodyne data from KITTI.

Generally this device is mounted in the top of the car, above the roof to avoid any possible obstructions for the 360 degree scan (figure 8).

Figure 8: Renault Zoe car equipped with VelodyneHDL64 used for Ground Removal and Segmentation research [27]

4.2 Segmentation

Segmentation is the process of labeling each pixel (or any of the smallest addressable unit in the input) to a class. This concept can be extended to other domains such as lidar point cloud where each point is classified.

Segmentation is of particular relevance for autonomous driving since is vital for the system to identify the objects present in the scene, their location and size. Unlike image classification, segmentation offers a more local analysis and that can be used later on more complex process such as localization, decision making, collision estimation [24], among many others.

The diversity and complexity of the classes depend on the dataset. In the automotive field, some research lines find enough to have four classes like in [4] but more recent dataset offer segmentation up to thirty-four [6]. Since the complexity of the model is proportional to the number of classes, current algorithms struggle with the accuracy in challenging classes. However, a richer segmentation could offer a better understanding of the scene and a higher reliability of the system.

There are several characteristics that are desirable for a segmentation algorithm that potentially could be used in a vehicle such as high accuracy, robustness and high processing time. The system

should be able to process the data in real time and be as fast as possible to forward its output to other modules for analysis and every scan can be processed before the new input arrives. If this is not achieved, the security of the system could be at risk considering the accumulative problem of late processing and the high demands of autonomous driving situations. Figure 3 shows an example of Image Segmentation in the automotive scene extracted from the KITTI benchmark website [24]. Another example is testing the work on Correlation Coefficient to discard irrelevant information in the automotive field [25], applying concepts that originally were tested on images to work on lidar information.

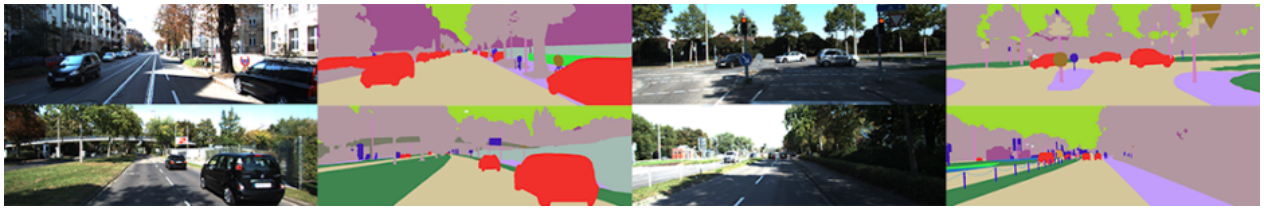


Figure 9: Semantic Segmentation in the Automotive Field for images

Figure 10: Instance Segmentation for Cars

Typically segmentation can be addressed by two different approaches:

1. Semantic Segmentation. This is when the system labels the input based purely in classes as shown in figure 9.
2. Instance Segmentation. This method segments the image also into different groups that correspond to the independent objects in the 3D scene. Fig 10 exemplifies this in an automotive scene with the class *Car*. The system labels pixels that correspond to a car in addition to the car that belong to.

4.3 Deep Learning

4.3.1 History

Deep learning resides inside the concept of artificial intelligence, a topic that has evolved and redefined itself through the years, but the idea of simulating cognitive thinking digitally is the one that has preserved and thrive until now [17]. Basic forms of artificial neurons came with the perceptron, a model that represents a single neuron that combines inputs to make predictions [18]. With this model, predictions can be made on small problems such as logic gates, but they are not able to model more complex data due to the size of the architecture. Limitations of this model encouraged researches to create more elaborated models that can handle complex data and be more robust as well. Just like in human brain, artificial neural networks are made of several neurons connected, each of them tuning their parameters according to the problem. In addition, the order, layers and type of connections between the neurons also determines the accuracy of the system and the user should configure the architecture depending on the application.

4.3.2 CNN

With the thrive of Deep Learning, researches tried to solve more type of problems with this logic, including image processing tasks. At first, for simple problems with small dimension images multilayer perceptron was good enough, but as the data size and complexity of task grows, this architecture is insufficient and the number of parameters increase as well as the computational expenses.

Convolutional neural networks or CNN are biologically inspired networks that are used in computer vision that are well known for image classification and object detection [18], but currently are also expanding to further image processing applications such as segmentation and object tracking. As explained in [18], for the convolution layers, a convolution operation is defined, in which a filter is used to map the activations from one layer to the next. A convolution operation uses a 3-dimensional filter of weights with the same depth as the current layer but with a smaller spatial extent. In traditional convolution, it uses the dot product to relate the elements each time and producing single number out of any convolution. The operation between the filter and the spatial regions in a layer is performed at every possible position in order to define the next layer (in which the activations retain their spatial relationships from the previous layer).

Methods before the expansion of deep learning for image segmentation included histogram-based methods and clustering in the color space [15] where the analysis was made from a more handcrafted

point of view.

The use of CNN allows the system to interpret the data and how to model it so the user's priority is the configuration of the network based on the application.

U-Net.

There are several CNN architectures that are reference model depending on the application. When it comes to image segmentation the U-Net is considered a reference model, being the most prominent and popular architecture in the medical imaging community [28], but being flexible enough for other fields of study [9].

This Network was published in 2015 and was originally developed for biomedical image processing, particularly image segmentation. As shown in figure 11 the network has an U shape having sections of early steps concatenated later on.

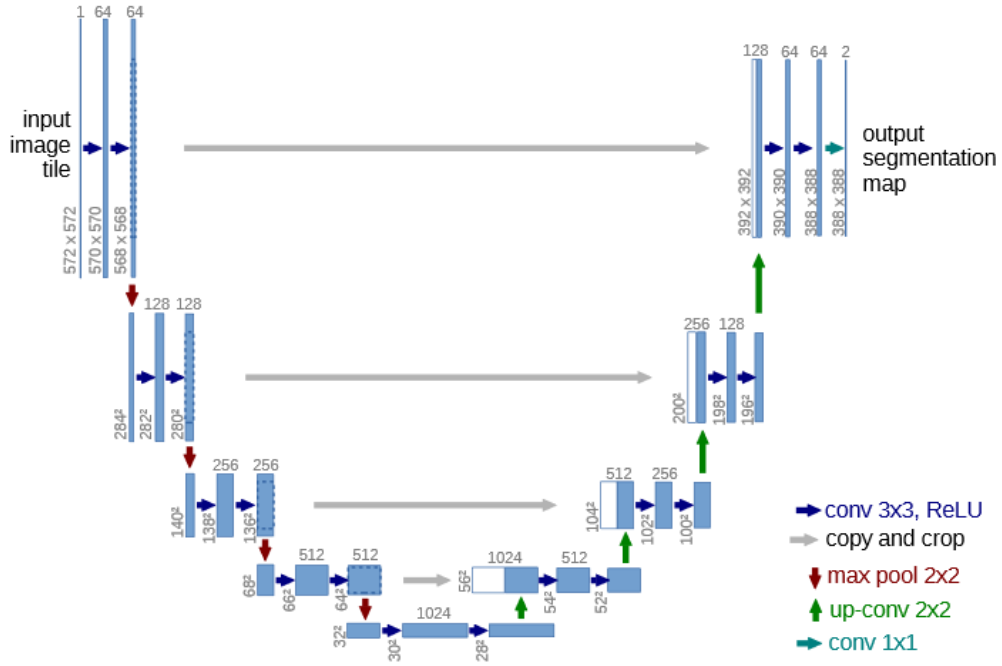


Figure 11: U-Net architecture as shown in original paper

The U-Net algorithm also propose a loss function that weights the edges of objects in the scene to improve segmentation on those regions. Although the publication of this network was only five years before to this document, is no longer considered state of the art. The reason is that the original architecture of U-Net does not provide sufficiently accurate results to compete to modern architecture. However, this network is commonly used as inspiration and even backbone of recent algorithms as it will be shown in section 5.

4.4 Metrics

The methods to rate a segmentation algorithm can variate depending on the application and reference dataset, but there are two that are mostly used for multi-class semantic segmentation:

4.4.1 Intersection over Union

Intersection over Union (IoU) is a metric commonly used when a bounding box is predicted around an interest object. It shows the relation between the true positives and false positive of the output (figure 12). This can also be extended to segmentation in the same way, where the areas are defined by the number of points in the class.

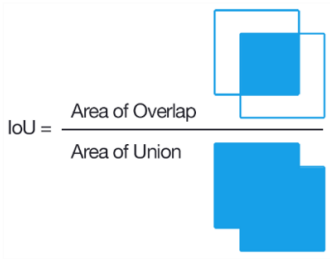


Figure 12: Intersection Over Union graphical representation

For multiclass problems, this metric is extended to mean intersection over union (mIoU), defined by the next formula:

$$\frac{1}{C}\sum_{c=1}^{C}\frac{TP_c}{TP_c + FP_c + FN_c} \tag{1}$$

where TPc, FPc, and FNc correspond to the number of true positive, false positive, and false negative predictions for class c and C is the number of classes [7]. There are some peculiarities of this metrics when is analyzed under different circumstances. For instance, small and big objects in the scene could have same intersection over union if the values of TP, FP and FN are proportional. Therefore this metric alone might not be a good indicator for decision making in real life applications, but is useful and widely used to score and compare segmentation algorithms.

4.4.2 Accuracy

Considering the deficiencies of the intersection over union metric, is useful to use a different metric to score the model. Pixel accuracy (point accuracy in this case) is a more traditional scoring method

that checks what points were labeled correctly in proportion to the total number of points:

$$Accuracy = TP/N \tag{2}$$

Where N is the total number of points. The combination of this two metrics offers the possibility of a deeper comprehension of the results easing the task of understanding the errors in the predictions. Mean IoU reflects the shapes and borders of the predictions in relation to the ground truth while accuracy shows a more global score.

5 State of the Art

5.1 Datasets

Although there are several datasets with lidar information, here we list the ones that are relevant for our research.

5.1.1 KITTI

KITTI dataset is a collection of data obtained through different sensors adjusted to a vehicle, capturing its path while driving in and around Karlsruhe, Germany [1]. The available data includes camera images, laser scans (including Velodyne HDL-64E), high-precision GPS measurements and IMU accelerations from a combined GPS/IMU system. Along with the raw data, the KITTI team also offers several algorithms based on the raw data, such as 3D object detection [2] or optical flow analysis [3]. This simplified the use of the benchmark and quickly became a reference dataset for the autonomous driving research community.

KITTI benchmark quickly became a reference for the research for advance Advanced driver-assistance systems (ADAS) and autonomous mobiles research for several reasons:

- Different sensors coexist in same scene, providing information that can be combined in different ways. Examples about this are the papers [24] and [26] using information from the lidar, camera and GPS all together.
- The dataset offers recordings in different locations like city, road, and suburban areas, covering scenarios for a typical vehicle day to day.
- It is open source, free of cost.
- Having an universal and popular dataset is convenient for comparing the accuracy of different algorithms, knowing that the data is the same and variations in scores will be purely on the processing of it.

These characteristics are sufficient to purse a research on the field saving time and resources for the data acquisition process. [24] and [26] present algorithms applicable in the automotive field, tested under different scenarios and conditions but still were developed entirely indoors

Working only with the data and scripts available in the KITTI benchmark, an extension for lidar segmentation was adapted. The idea is to use scripts for 3D segmentation in images and combine it

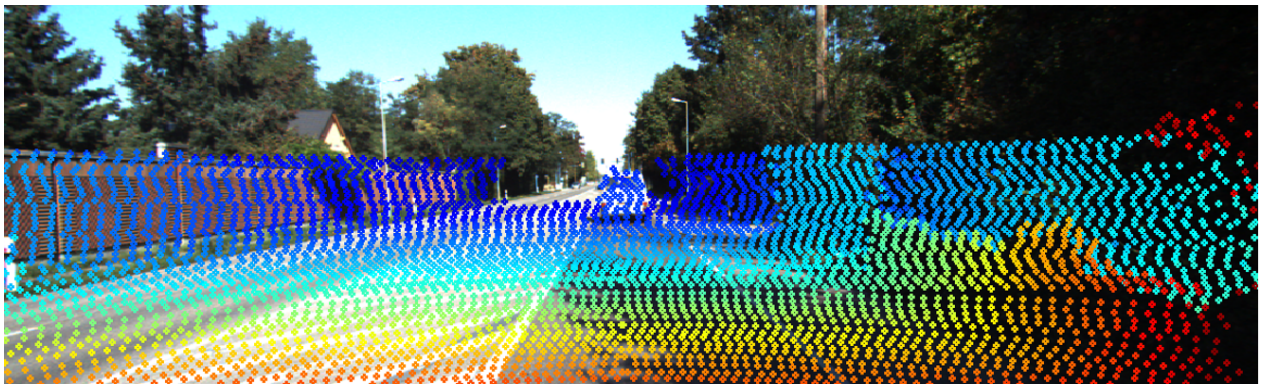


Figure 13: Graphical representation of the lidar-camera registration using KITTI APIs

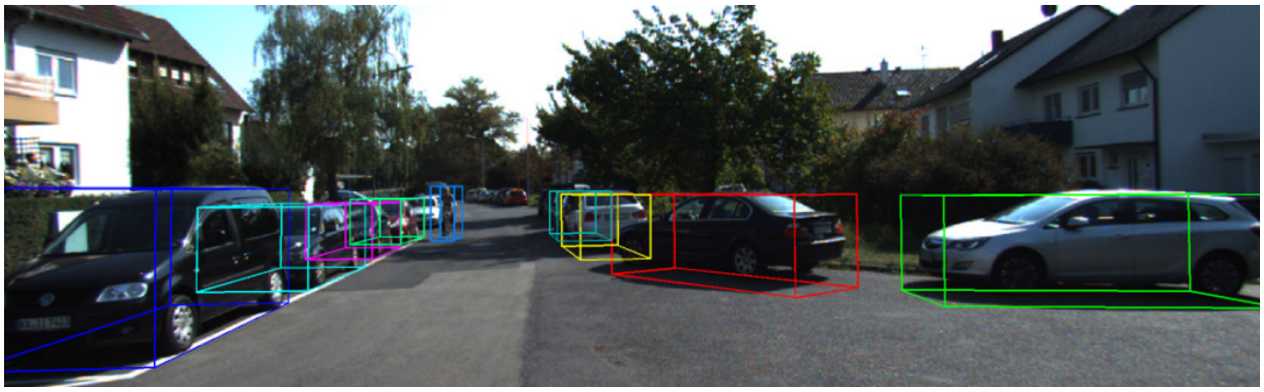


Figure 14: Bounding Box prediction using KITTI APIs, combining lidar and camera information

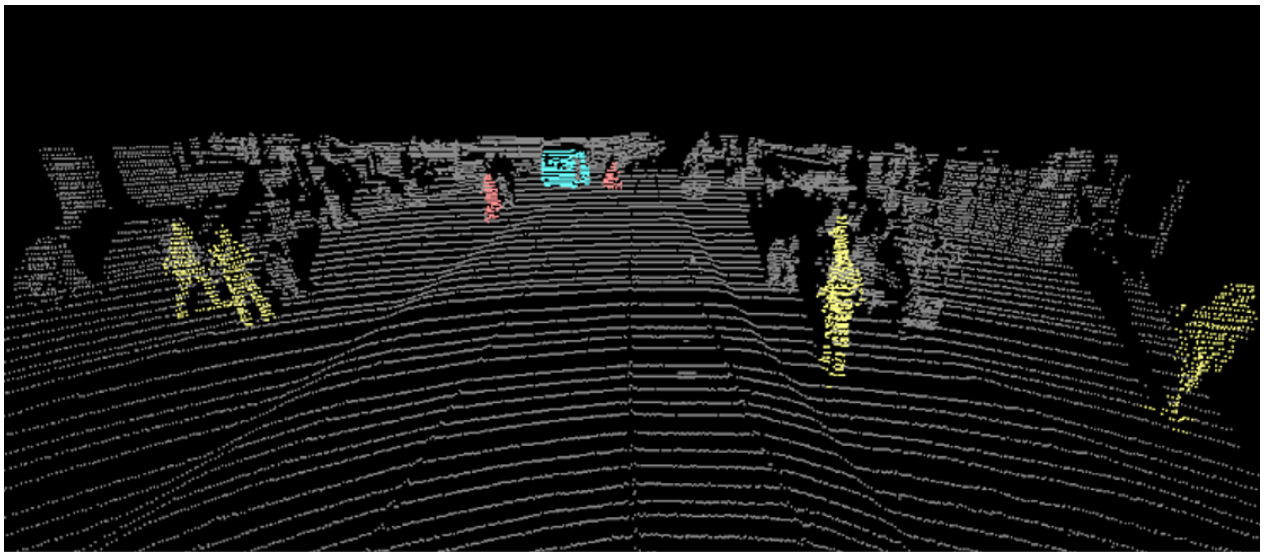


Figure 15: Point Cloud Segmentation using 4 classes: Unlabeled, Car, Pedestrian and Cyclist

with the registration between image and lidar (figure 13 and 14). This way was possible to label a front view of the velodyne pointcloud on four different classes: Unlabeled, Car, Pedestrian, Cyclist

as shown in figure 15. [9]. This is the dataset used by the segmentation algorithm SequeezeSeg [4] and others that work on lidar segmentation like [5] or [9]. Relevant limitations of this dataset are listed below:

- The region of segmentation is limited to the image field of view and in real life applications it is relevant to segment the entire point cloud, without discarding any information.
- The number of classes is limited to four, and this might not be sufficient for robust applications.
- The accuracy on the registration between point cloud and image depends of the accuracy of the translation algorithm.

5.1.2 SemanticKITTI

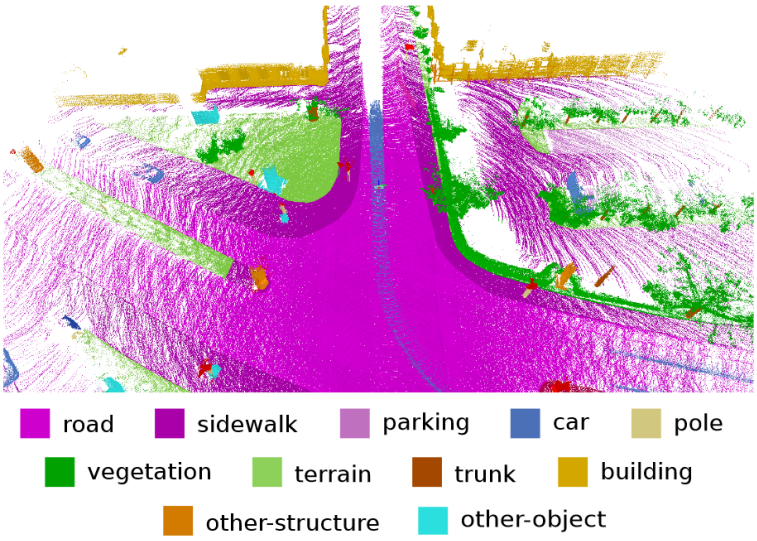


Figure 16: Point Cloud Segmentation by SemanticKITTI

As described in their paper [6], the SemanticKITTI dataset annotated the sequences of the KITTI Vision Odometry Bench-mark and provides dense point-wise annotations for the complete 360 degrees field-of-view of the employed automotive LiDAR. They present a point-wise annotated dataset of pointcloud sequences with an unprecedented number of classes and unseen level-of-detail for each scan. With this, the main limitations of previous KITTI dataset were solved and an extension of the dataset is offered.

The dataset contains a total of 34 different classes as shown in table 1. We included the class frequency (content, as called by the authors) to show differences in what extend the classes are

Table 1: Classes available in SemanticKITTI with corresponding frequency

Number	Class Name	Content	Number	Class Name	Content
1	unlabeled	0.018889855	18	fence	0.07235922
2	outlier	0.00029372	19	other-structure	0.00239513
3	car	0.040818519	20	lane-marking	4.71E-05
4	bicycle	0.000166095	21	vegetation	0.26681502
5	bus	2.79E-05	22	trunk	0.00603501
6	motorcycle	0.000398386	23	terrain	0.07814222
7	on-rails	0	24	pole	0.0028555
8	truck	0.002063361	25	traffic-sign	0.0006156
9	other-vehicle	0.00162182	26	other-object	0.00992313
10	person	0.000176986	27	moving-car	0.00178931
11	bicyclist	1.11E-08	28	moving-bicyclist	0.0001271
12	motorcyclist	5.53E-09	29	moving-person	0.0001606
13	road	0.198749387	30	moving-motorcyclist	3.75E-05
14	parking	0.01471717	31	moving-on-rails	0
15	sidewalk	0.143922984	32	moving-bus	0.00011352
16	other-ground	0.003904855	33	moving-truck	0.00010158
17	building	0.132686194	34	moving-other-vehicle	4.38E-05

present in the sequences and how they relate to each other. The number column is provided as reference for an easy counting of the classes and is not related to class numbering in the APIs.

Per point in the scan, a class is given along with the instance number. These classes can be mapped to a simpler version of 20 classes for a static analysis. Static or single frame analysis is done when the algorithm considers only a frame at a time for the segmentation. This way is not possible to know if the objects on the scene are moving or not. The mapping of the classes leave with the classes listed in table 2.

Test challenge.

From the 22 lidar sequences available in KITTI benchmark, semanticKITTI made public 11 of them: sequence 0 to 10 and 11 to 21 remained unrevealed so that researches can make predictions for these sequences and be submitted in the public online challenge for evaluation:

`https://competitions.codalab.org/competitions/20331`

The amount of total submissions is limited to 10 to participant so that the users do not learn from the results. For this research, the team work on the metrics and evaluation method of this challenge, considering the published papers as baseline.

Table 2: SemanticKITTI with classes mapped for static analysis

Number	Class Name	Content
0	unlabeled	0.031501833
1	car	0.042607829
2	bicycle	0.000166095
3	motorcycle	0.000398386
4	truck	0.00216494
5	other-vehicle	0.001807055
6	person	0.000337583
7	bicyclist	0.000127111
8	motorcyclist	3.74611E-05
9	road	0.198796471
10	parking	0.01471717
11	sidewalk	0.143922984
12	other-ground	0.003904855
13	building	0.132686194
14	fence	0.072359223
15	vegetation	0.266815021
16	trunk	0.006035012
17	terrain	0.07814222
18	pole	0.002855498
19	traffic-sign	0.000615596

5.2 Deep Learning Segmentation

Analysis of a point cloud is of particular complexity in the automotive field, considering the range of values that are encountered in the city or the road. For this reason, in the area of Semantic segmentation, preprocessing and data preparation is not straight forward and researches have tried different representations depending on the application. As stated in [19], some of these approaches include Point-wise multilayer perceptron (MLP), Multiview representation, Graph based and Spherical representation to name a few. Our line of research is the Spherical representation, more specifically, Range image, a representation of the pointcloud in an image-like fashion. This concept will be explained in detail in the methodology section.

Relevant algorithms for our research are listed below:

5.2.1 SequeezeSeg and SequeezeSegV2

SqueezeSeg is a pointcloud segmentation network that works on a spherical representation of the scan, using conditional random field (CRF) as a postprocessing on the prediction [4]. SqueezeSegV2 is the second version of SqueezeSeg that offers and update in the training loss, batch normalization

and additional input channel (figure 17). This architecture became relevant after offering an accessible public version of the KITTI scans already formatted as range images. Their results are often used as reference for new segmentation methods.

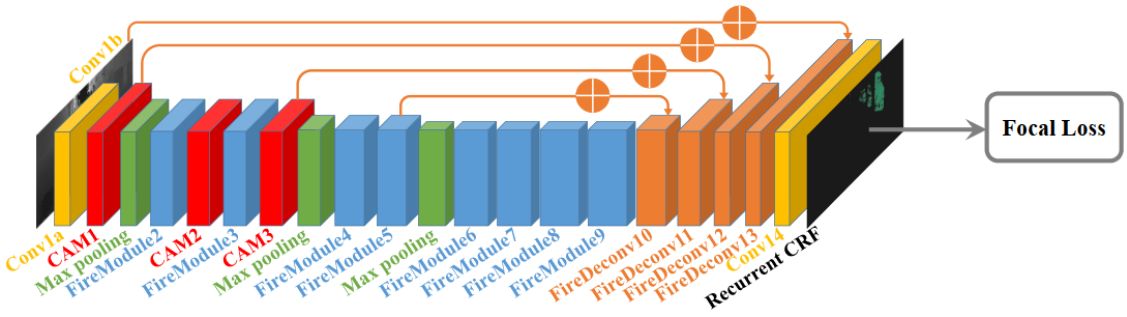


Figure 17: SqueezeSeg_v2 architecture. CAM stands for Context Aggregation Module and CRF are Condition Random Fields used in the refinement of the results

5.2.2 RangeNet++

Coming from the authors of semanticKITTI, this network works as the first reference for the public online competition. Their proposal is a network with 53 layers (inspired in YOLOv3), a loss function that considers class frequency and postprocessing. Unlike the range image dataset proposed by SqueezeSeg, semanticKITTI works on pointclouds, which means that any architecture that outputs a range image still needs to do postprocessing to map it to a pointcloud to have results on the scan segmentation. For this purpose the authors include a fast GPU K-Nearest Neighbour process after the Range Image prediction.

Since this architecture also works with Range Images and comes from the SemanticKITTI team, is considered to be a reference method for pointcloud segmentation [19].

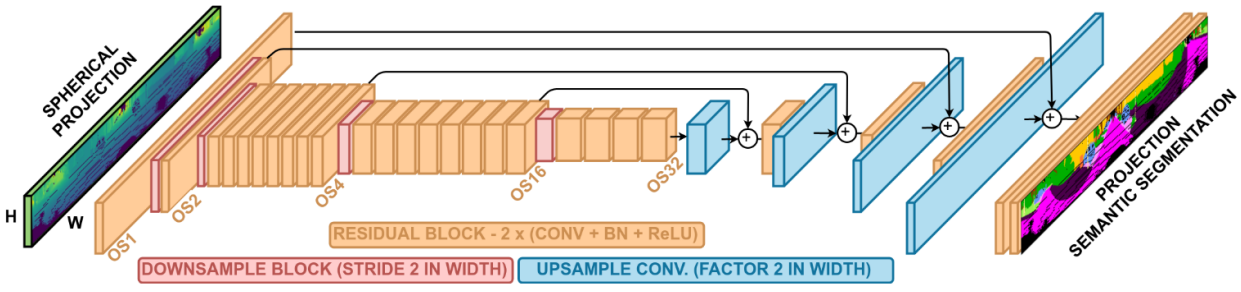
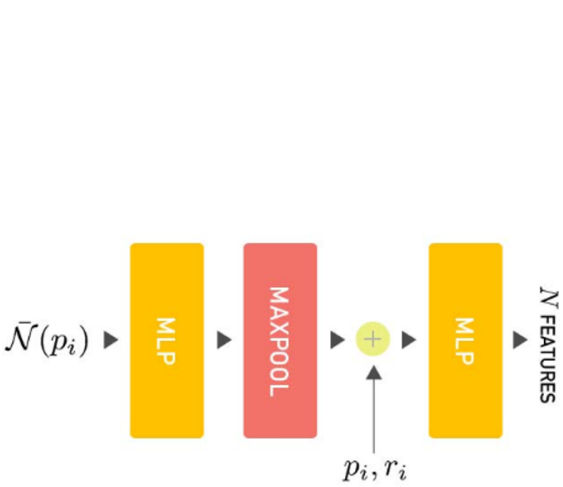


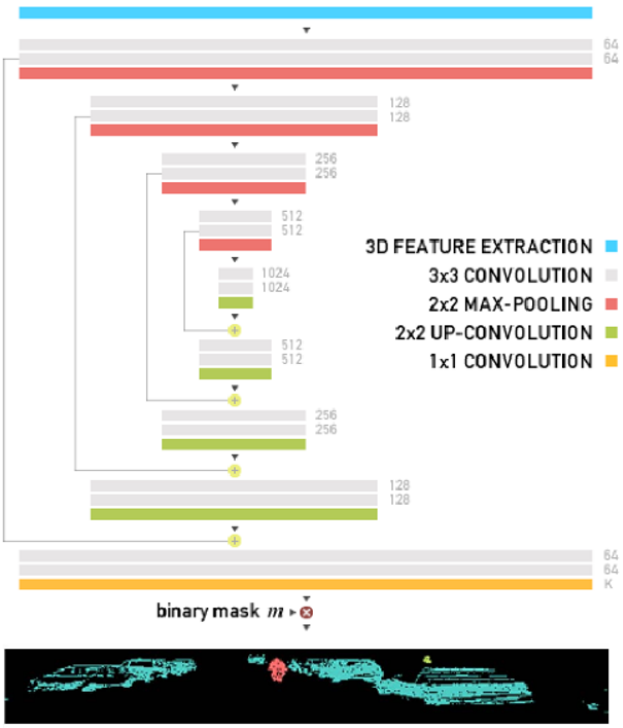
Figure 18: RangeNet53 from RangeNet++

5.2.3 LU-Net

This architecture was developed using the dataset from SqueezeSeg, and is composed of a channel learning part inspired in PointNet and a regular U-Net [9]. The first one is a pointcloud segmentation method based on multilayer perceptron inspired by [22] and [13]. The second one is U-Net, a robust segmentation network famous in the medical image field [8] as explained in section 4.3.2. Figure 19 shows the workflow of the network. With a small architecture and no post-processing, this method manages to score higher and average mean IoU higher than SqueezeSeg2 [9]. This architecture is the baseline for this research, trying to make it functional for more demanding datasets.



(a) LU-Net 3D Feature Extraction Module



(b) LU-Net full architecture

Figure 19: LU-Net Architecture includes, (a) a feature extractor module that learns N features from the input $\bar{\mathcal{N}}(p_i)$ that is neighbouring information with respect of every point to be later concatenated with actual points. (b) Later, segmentation is performed by a conventional U-Net

The first part of the LU-Net architecture is a 3D feature extraction, as the author named the small network to learn N number of features to represent each point. The idea is to have a Network that combines all the points and their channels and learn N features that represent well the data. The justification for this process is that is not clear the impact of each of the channels as input for a deep learning process, therefore a network can learn how to weight and combine all this information.

The input of this architecture is different than other Range Image methods, since it includes

the neighboring points as a reference to the point analyzed. This is in addition to the actual point information. With this, there is a larger number of elements representing each point on the Range Images, that contains information about the neighborhood. A detailed description of these process can be found in [9].

6 Methodology

In this section, we list the algorithms and concepts relevant for our line of research.

6.1 Range image computation

Building range image from point cloud is a pre-processing step that selects and orders points in an image-like fashion.

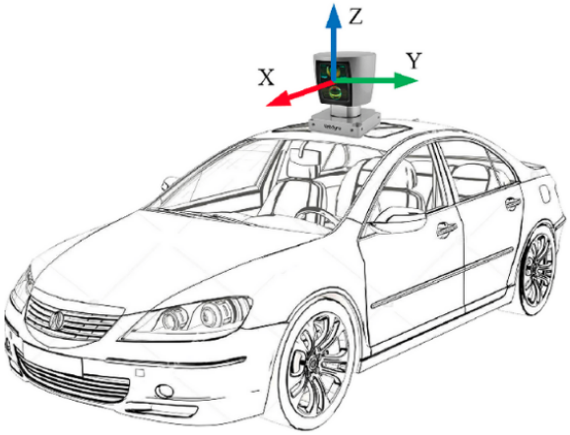


Figure 20: Lidar location in a car for the KITTI dataset. The arrows show the coordinate system definition needed for the Range Image formulas

From the point of view of the sensor, every point in the point cloud can be represented as two angles:

1. Vertical angle θ: Elevation from the Horizontal plane.
2. Horizontal angle ϕ: Angle from the Y axis, around the car. Also called azimuth [6]

The operations to calculate each of the angles is:

$$\theta = arcsin\frac{z}{\sqrt{x^2+y^2+z^2}} \tag{3}$$

$$\phi = arcsin\frac{y}{\sqrt{x^2+y^2}} \tag{4}$$

Where x, y and z are the point coordinates.

Considering that the lidar contains 64 vertical beans, the RI height should be 64 to be dense. The vertical range is also known from the specs: 26.9 degrees. With this we know that the vertical resolution should be

$$v_{res} = 26.9°/64 = \sim 0.42° per\ pixel \tag{5}$$

On the other hand, the horizontal resolution of 0.08 degrees can not be treated the same way because the resulting width dimension of the range image would be large and disproportionate to the height. Therefore, values of 512, 1024 and 2048 and typically selected, in an empirical way, to have dense and undistorted image. The horizontal view can be trimmed to any desired angle depending on the application, but to have a full view of the scene, 360 degrees should be used. With these values, the resolution of the horizontal discretization using 1024 value, should be:

$$h_{res} = 360°/1024 = \sim 0.35° per\ pixel \tag{6}$$

A tensor with dimension (64,1024,n_channels) is created, filled with zeros. For every point in the point cloud vertical and horizontal angles are found and all the point information is added to the matrix in the corresponding index. Due to the discretization process, some points might reside in the same pixel of the range image, so only the closest point is chosen and stored. In this algorithm, the number of channels is 5: 4 original values x,y,z,i (intensity) and 1 extra used in range images called 'range' (r) which is just the euclidean distance of the point:

$$r = \sqrt{x^2 + y^2 + z^2} \tag{7}$$

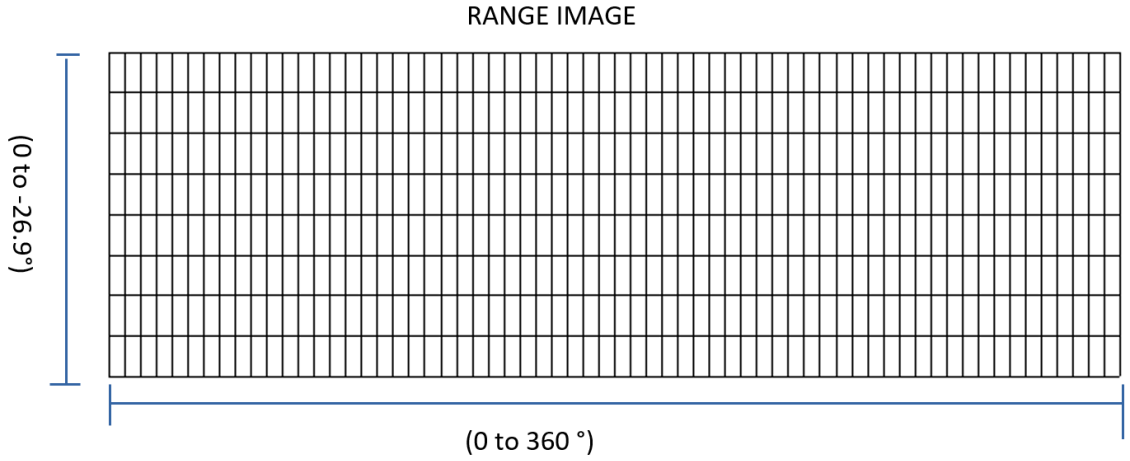


Figure 21: Representation of the Range Image as a matrix that sorts the points of the cloud based on their angle with respect to the sensor. Angles limited by the actual range of the lidar but width and height of the Range Image are defined by the user

A visualization of a point cloud using the colors per label provided by SemanticKITTI is shown

in figure 22. This corresponds to the sequence 08, scan 000000. The corresponding Range Image of this scan is in figure 23, created with a widht of 1024. Figure 24 shows the zoom of the RI in the center region (front of the car) for better visualization.

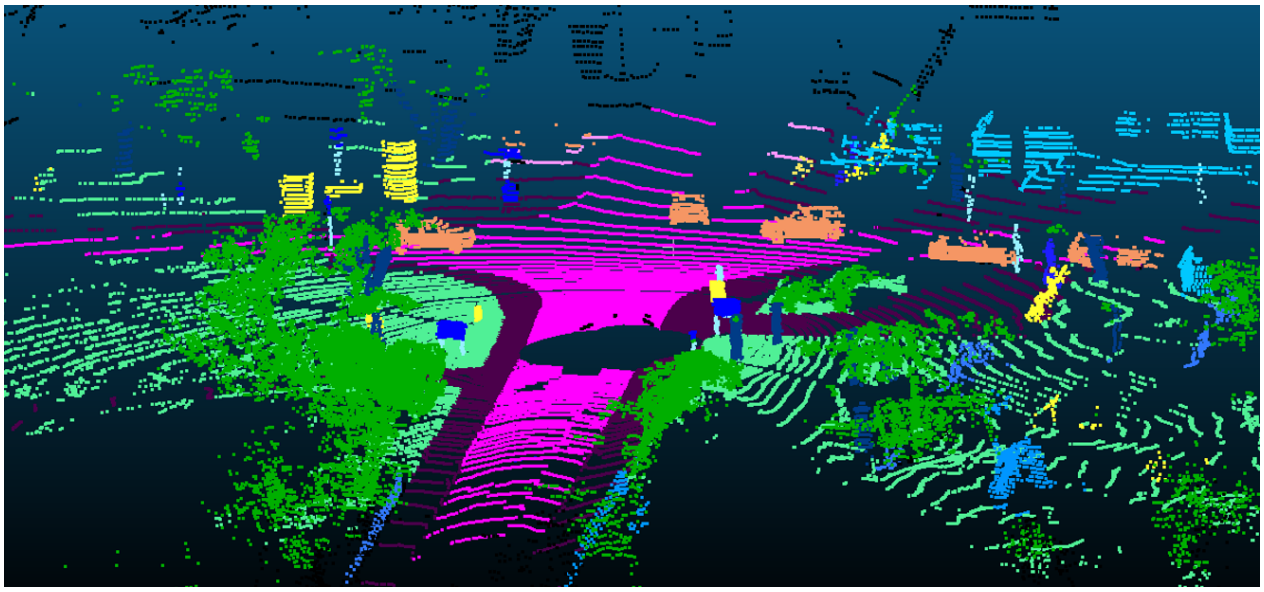


Figure 22: Point cloud visualization with classes, sequence 8 scan 00, from semanticKITTI dataset

Figure 23: Range Image visualization with classes, sequence 8 scan 00, from semanticKITTI dataset

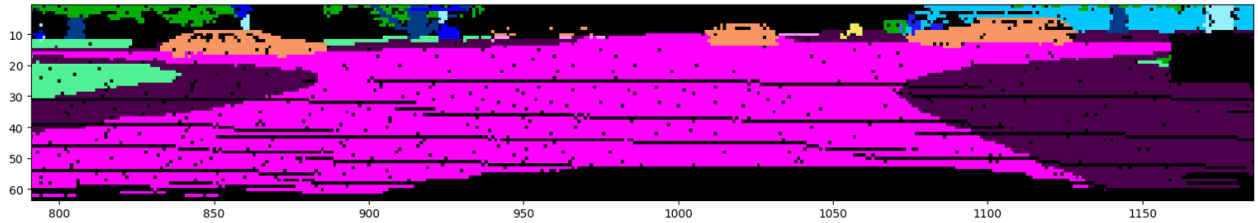


Figure 24: Zoom in of the center part of figure 23, which is the Range Image visualization with classes, sequence 8 scan 00, from semanticKITTI dataset

6.2 Loss Function

6.2.1 Cross Entropy

The loss function is also called error function or cost function and in simple terms it measures in what extend we got the results right and if not, by how much we missed [20]. A traditional method

is the cross entropy. Equation 8 shows how to apply it for binary classification:

$$CE(p, y) = \begin{cases} -log(p), & \text{if } y = 1 \\ -log(1-p), & \text{otherwise} \end{cases} \tag{8}$$

In equation 8, $y \in \{\pm 1\}$ specifies the ground-truth class and $p \in [0, 1]$ is the model's estimated probability for the class with label y = 1. [10]

6.3 Focal Loss

Although cross entropy is sufficient to represent the cost of the prediction, there is an update of this function that focuses on making classes that are scoring low during training, giving them more weight than those with high score (equation 9). This is achieved by adding the modulating factor $(1 - p_t)^\gamma$:

$$FL(p_t) = -(1 - p_t)^\gamma log(p_t) \tag{9}$$

In practice is suggested to use an α-balanced variant which value depends on the application and should be tunes along with the gamma parameter [10].

6.3.1 U-Net weighted loss

The loss function proposed in the U-Net paper includes a weighting method for the pixels that are on the edge of the elements, to focus on prediction in the contours. This term multiplies the cross entropy and shown in eq. 10:

$$w(x) = w_c(x) + w_0 \cdot exp(-\frac{(d_1(x) + d_2(x))^2}{2\sigma^2}) \tag{10}$$

Where $w_c : \Omega \to R$ is the weight map to balance the class frequencies, $d_1 : \Omega \to R$ denotes the distance to the border of the nearest cell and $d_2 : \Omega \to R$ is the distance to the border of the second nearest cell. As reference, U-Net authors set $w_0 = 10$ and $\sigma \approx 5$ pixels. The distances are found using morphological operators [8].

6.3.2 LU-Net loss

An important innovation in LU-Net is to combine focal loss function with the contour weighting term of U-Net, proposing a function that weights predictions on the edges of objects but also classes that are scoring low during training.

$$E = \sum_{x \in \Omega} -l_{\{m(x)>0\}} w(x)(1 - p_{l(x)})^{\gamma} log(p_{l(x)}(x)) \tag{11}$$

Equation 11 shows the elements of this loss function, Where Ω is the domain of definition of u, $m(x) > 0$ are the valid pixels, $\gamma = 2$ is the focusing parameter and $w(x)$ is a weighting function introduced to give more importance to pixels that are close to a separation between two labels [9].

6.3.3 Dice Loss

There is a connection between the loss function and the evaluation method of the system. Since this function will define and measure error in the predictions while training, this same definition of error is the one that should be use in the evaluation of the system. Therefore, there is a loss function definition that resembles the intersection over union metric and pushes the system to optimize on that direction.

Dice loss is the proposal for this problem, a function that specializes in overlap of the prediction and groundtruth [11].

$$DL_2 = 1 - \frac{\sum_{n=1}^{N} p_n r_n + \epsilon}{\sum_{n=1}^{N} p_n + r_n + \epsilon} - \frac{\sum_{n=1}^{N} (1 - p_n)(1 - r_n) + \epsilon}{\sum_{n=1}^{N} 2 - p_n + r_n + \epsilon} \tag{12}$$

For binary segmentation the formula is defined in eq. 12, with $r_n \in R$ being the reference foreground segmentation and $p_n \in P$ the prediction. ϵ is added to the equation to ensure stability to the system.

This function can be extended to a categorical segmentation with the Generalized Dice Loss (GDL, equation 13):

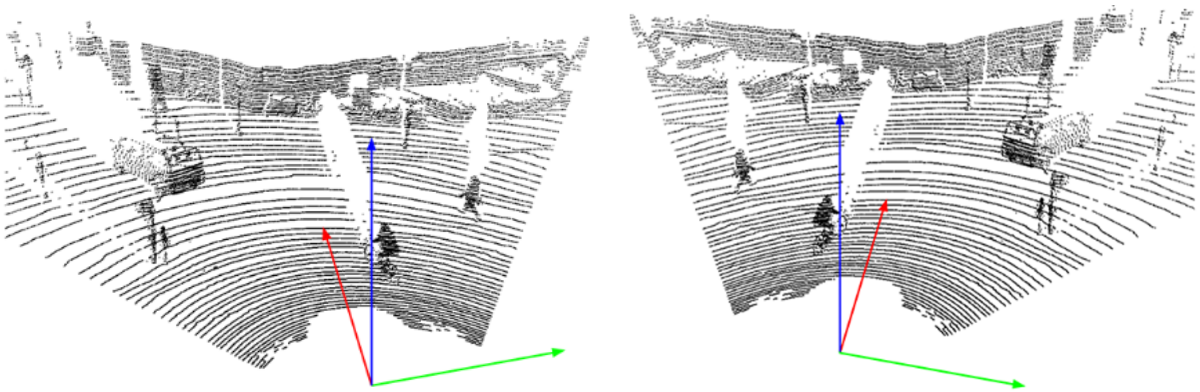
$$GDL = 1 - 2\frac{\sum_{l=1}^{2} w_l \sum_n r_{ln} p_{ln}}{\sum_{l=1}^{2} w_l \sum_n r_{ln} + p_{ln}} \tag{13}$$

Where w_l is used to provide invariance to different label set properties.

As proposed by [11] this weighting parameter is the inverse of elements present in the scene per class (figure 14).

$$w_l = \frac{1}{(\sum_{n=1}^{N} r_{ln})^2} \tag{14}$$

This way the classes that are less frequent will have a stronger impact on the loss. All these characteristics came mainly from medical imaging field where the pictures analyzed contain objects of different sizes and nature, and to have a robust segmentation algorithm is necessary to train with a loss function that is design to deal with these obstacles, something that traditional Cross entropy does not accomplish sufficiently.



(a) (b)

Figure 25: Visualization of the Random Flip algorithm for data augmentation. (a) Original point cloud. (b) Point cloud after flip on x axis. If this frame is selected randomly all points will be flipped using x axis as reference

6.4 Data Augmentation

For data augmentation the team followed the paper "Quantifying Data Augmentation for LiDAR based 3D Object Detection" [21] that goes over possible augmentations techniques for our input data.

Firstly the authors state that there is a large gap between the baselines with no augmentation compared to the augmentation policy, some cases with more than 17% of accuracy increase.

The paper mentions that the most effective data augmentation techniques for pointclouds are:

1. **Global Rotation**. This means to rotate every point $p(x, y, z) \in P$ around the upright yaw axis by an angle α drawn from a uniform distribution. We consider this data technique not suitable for our research, since in original data the location and dimensions of the road is stable thoughout the scans. Doing a global rotation will change this fact.

2. **Global Translation**. This means to translate every point $p(x, y, z) \in P$ such that an augmented point $p*$ has the form $p(x+\Delta x, y+\Delta y, z+\Delta z)$. Similar to last point, it would represent to changing location of road with respect to the car, which we consider to be undesirable.

3. **Global Global Scaling**. This is scaling every point $p(x, y, z) \in P$ in every direction by a scalars drawn from a uniform distribution. This technique is risky from the Range Image point of view, since a scaling could cause dark areas when creating it.

4. **Random Flip**. This means that every point$p(x, y, z) \in P$ is flipped by a 50% chance on the forward facing x-axis [21]. For all the four suggested methods this is the one that fit our demands for mantaining the morphology of the data.

For all these methods, the same logic for augmenting the data should be followed for the labels to have consistency in the annotations.

Since the authors of [21] were working on a dataset where labels were only on the front part of the scene, facing the car (most likely same dataset than SqueezeSeg and LU-Net), there was no sense for a y-axis flip. In SemanticKITTI all the pointcloud is segmented, so flipped on y-axis was also included for some tests. However, in our case, there was not much improvement with this extra flip and it only caused longer converging time. Therefore, only random flip of 50 percent chance on the forward facing x-axis was considered.

6.5 Validation

Usually in deep learning for image processing the real-time validation consist in selecting randomly a batch from the validation set and do forward pass on our model every certain amount of iterations to compute the loss. With this we could know the behavior of our model when faced to data that is not part of training and commonly is used to check when to stop training. In our case this process was not representative since some scans only contain few classes present and does not generalize if picked randomly. An example of this is shown in figure 26, showing the behaviour of one of our configurations. On light transparent blue, we can see the behaviour of the loss without smoothing the graph and how unstable it gets. Even when smoothing the graph, validation behaviour is still not representative for choosing an overfiting area. To select the best checkpoint in the model, a "full validation" needs to take place, where the whole validation sequence is forward passed and evaluated. This is time consuming but efficient in terms of selection of the best state of our model. In this scenario the figure 27 shows the mIoU scores of all validation set at every checkpoint. In

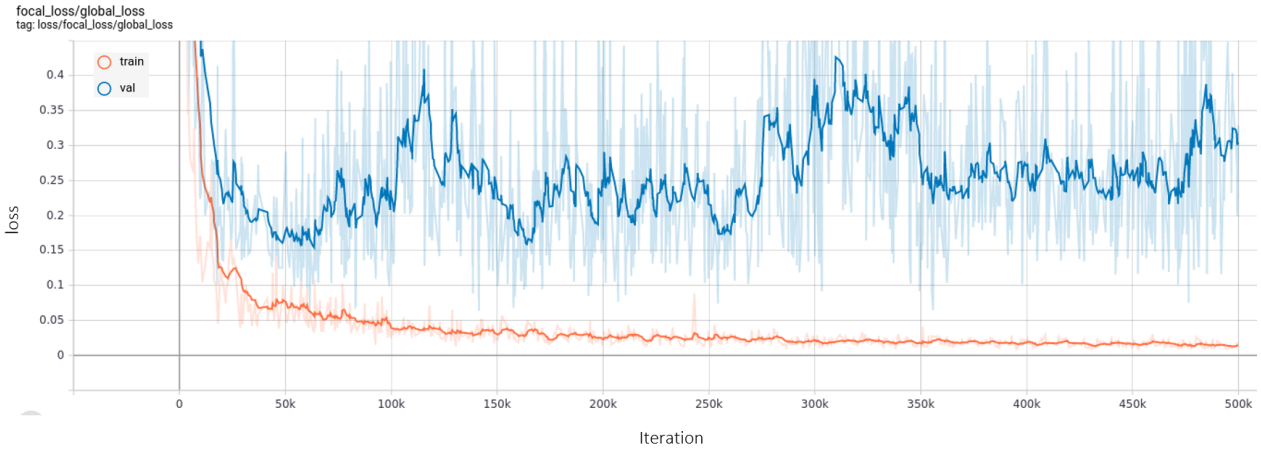


Figure 26: Loss graph showing train and validation scores

this case we recorded every 20000 iterations. The number of iterations between each checkpoint can variate depending on the stability of the results through the graph.

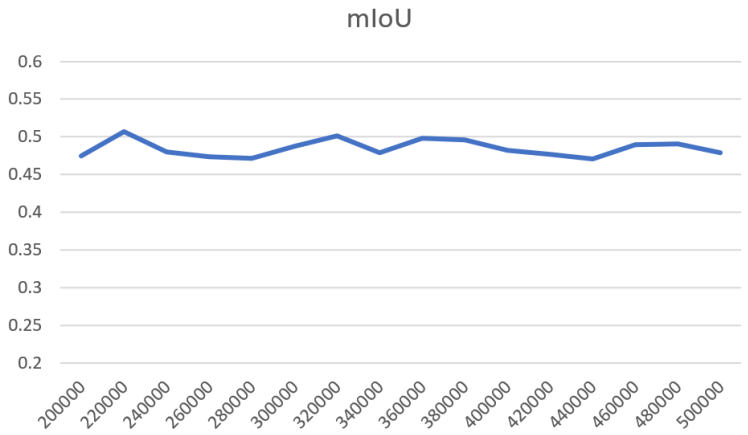


Figure 27: Full validation every 20000 iterations

The difference between figures 26 and 27 shows how in our scenario we can not rely on validation loss to select the best state of our model that does not over-fit and generalizes well the data.

6.6 Propagation

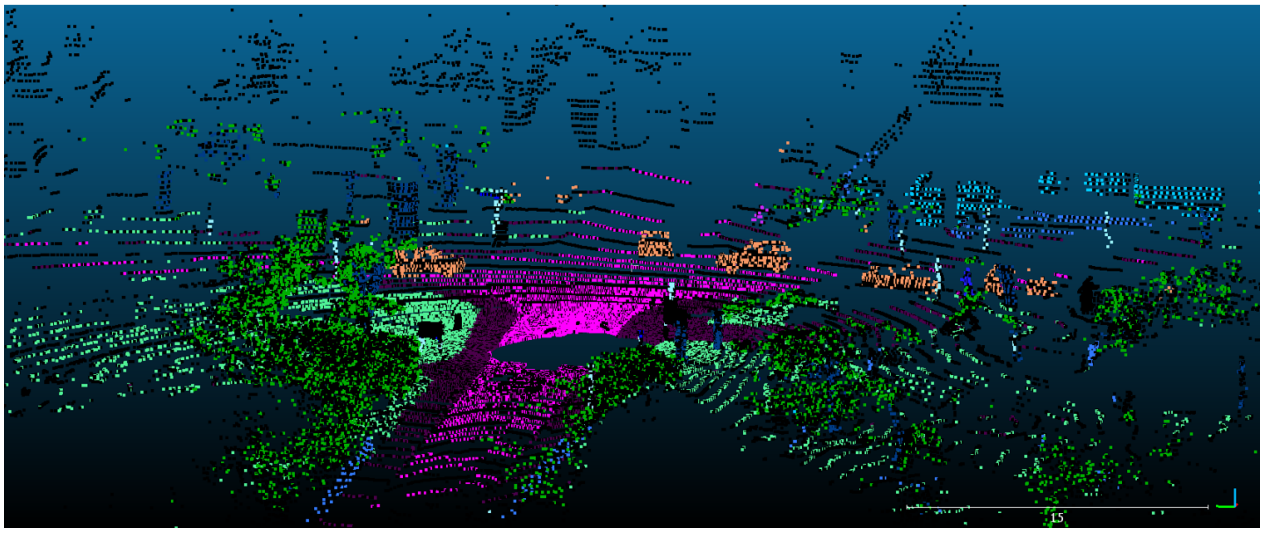
The dataset that is offered by the SqueezeSeg [4] team is purely range images. Therefore, LU-Net [9], that was developed with this dataset uses a Range-Image-only training and validation process. To extend this model to semanticKITTI dataset, firstly we need to go from pointcloud to range image. We followed the method explained in section 6.1 that is embedded in the SemanticKITTI APIs. Secondly, after the model makes predictions for the Range Image, there should be a propagation

algorithm, where the Range Image is projected back to the point cloud to label all points. Following the lead of RangeNet++ [7] we do this process using K-Nearest Neighbor algorithm, a non deep learning-based unsupervised method using the information of spatially closest neighbors. Up to this point we use python libraries to do this process with the idea of expanding to fast GPU-enabled KNN later on as in [7]. Configuration K=3 and neighbors weighted by distance gave us the best results. The range image representation is the discretization of the pointcloud. As a consequence, it produces occlusions between points on different scales depending of the image width, so when going back to the pointcloud it could have errors on objects edges. In addition, if the prediction is done wrong in a region, this error will be propagated to all the points in the same range.

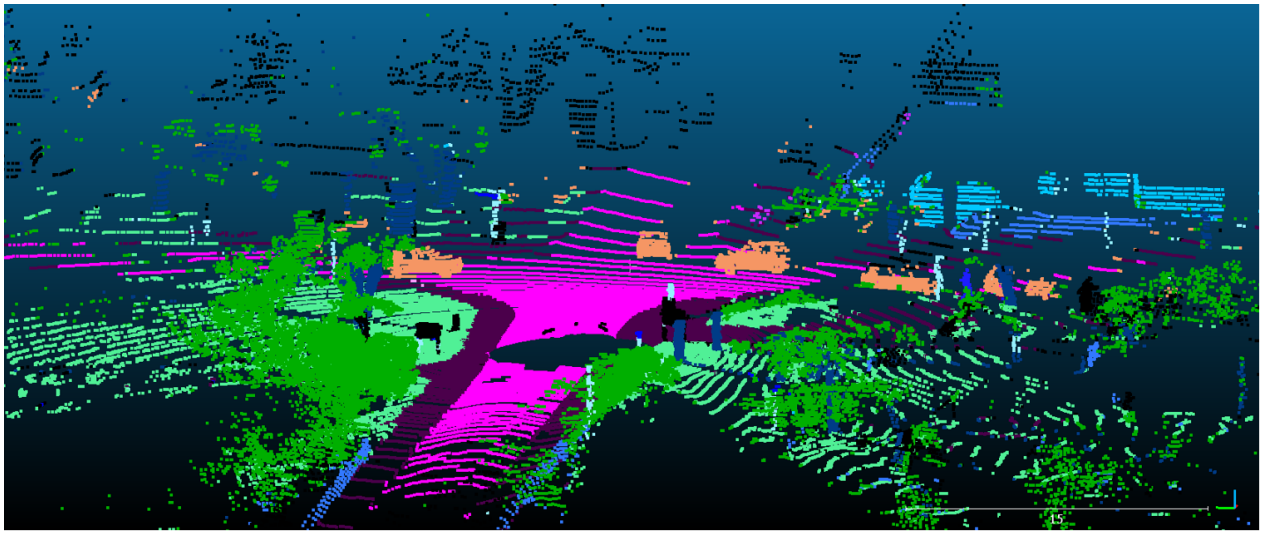
Figure 28: Showing 3/5 of the predicted RI (front view)

Figure 23 shows the front view of the Range Image in sequence 08 scan 000000, using the labels predicted by one of our models. In this example, we show a commonly problematic class in this data set which is "other object" displayed in yellow in the ground truth (figure 23 and 22). In this scan, our prediction classifies these points as "unlabeled". Figure 29a shows how the points that were selected for creating the Range Image relate to the rest of the point cloud (colored in black as unlabeled) and figure 29b shows the cloud after the KNN propagation process. It is noticeable how the error can be projected to farther point in the cloud when doing the propagation by observing the colors.

Due to this propagation process, comparing Range Image Segmentation methods is problematic, since the results on the SemanticKITTI platform are made only at a point cloud level. There is no information available about RangeNet++ of SqueezeSeg before propagation in the SemanticKITTI dataset. The results on a Range Image level evaluation is relevant in order to understand how the score is affected by the propagation into the point cloud under different image sizes.



(a)



(b)

Figure 29: (a) Semi propagation in point cloud, (b) Fully propagated using KNN

7 Configuration Proposals

Throughout the research, many configurations were tested using LU-Net architecture as start point, trying to adapt it to the new SemanticKITTI dataset and to increase the score of the predictions. Although more than 30 different configurations were tested, here we will list only those that offered a significant improvement and then were used as baselines for next trials. The scores obtained on these configurations will be presented in section 8. Configuration that remained stable are the following:

- Batch size = 2
- Sequence 8 as validation set
- Range Image of size 64x1024
- Neighbors defined using a window of 3x3 in the Range Image

7.1 Original LU-Net Starting Point

This is the original LU-Net adapted to SemanticKITTI dataset, the reference starting point. Parameters defined by the authors remained the same, except for batch size. Originally was set to 4 but since the Range Image size was doubled, this value was too high for the computational point of view.

7.2 V2 - Data Augmentation

Following the research about lidar and data augmentation in [8], we expanded the data using random flip in x axis of 50% of the scans, as explained in section 6.4. First attempt for this data augmentation technique we did it by selecting randomly the scans to be augmented. On a different test we augmented every two scans in the sequences and obtained better results with this approach. This is because the scans that we work with were taken continuously. With this in mind, in random selection there is a chance to have consecutive frames that does not offer relevant differences, or there could be also large gaps between scans selected. With this we improved mean intersection over union by 2 percent.

7.3 V3 - Update Loss Function

Loss function has a relation between the evaluation metrics and we noticed the original model was not including Dice loss when mIoU is the scoring formula.

As stated in [12], the Dice loss stability depends on the initial parameters of the system such as learning rate. To avoid making a full study of Dice loss for pointcloud segmentation, we simply add the Dice loss to the LU-Net loss to create a more stable error function but with the influence of Dice measure on the metrics. In addition, the KITTI dataset offers the class frequency throughout the training data, so that the user can know in advance the probability of a class to appear during training. With this, we decided to include this value in the loss function so that the system can weight towards uncommon classes by a constant factor. This is different to the weight in Dice loss, since there the weight is based on the current batch analyzed and not in the whole dataset. The final loss model looks as following:

$$loss = [GDL] + (-log(f))\,[LU\ Net\ loss] \tag{15}$$

Where f is a vector with the class frequencies, informations offered in the SemanticKITTI dataset. The Multimodal Dice Loss and the LU-Net loss was explained in loss function section 6.2.

For this step the learning rate was also updated. Original LU-Net had a learning rate decay algorithm to reduce this parameter through training. Unfortunately this was no functioning correctly in the released code and this failure was making the system to work always with the value $lr = 0.0001$. This value was too small for our model so it was increased to $lr = 0.001$. With this the system shows faster convergence.

7.4 V4 - Feature extraction redefinition

As mentioned in 5.2.3, the first part of LU-Net architecture is a feature extraction module that learns N channels to represent the data. Originally this process is inspired in Point-Net [13] that uses multilayer perceptron that are typically simulated with 1x1 convolutions as shown in section 5.2.3.

While reviewing the code of LU-Net, some errors where encountered in the adaptation of Point-Net into LU-Net architecture and data. For example:

- First max pooling layer was discarding information of most neighbors and processing just one of them in next steps
- Fixing the kernel size of this pooling layer will merge the channels in a way that is not justifiable and does not follows the approach of Local Point Embedder, presented in [9], which is the inspiration for LU-Net.

The feature extraction module is adapted to the input data by the use of a reshape method in the range image. In the input and excluding the batch size, the data shape is $64 \times 1024 \times 8 \times 3$, first two elements defining the range image dimensions, and the last two element are the neighbours relative information. The author then reshaped the tensor to $65536 \times 8 \times 3$ to be able to make the 1×1 convolutions throughout the neighbours. Problem with this operation is that the analysis is done by turning the image into a vector of points and loosing its 2D properties, needed for understanding the input as an image.

Without the possibility of adapting the current architecture to maintain the tensors dimensions for the rest of the Network, we decided to do a redefinition of the 3D feature extraction module. The motivation for these changes also came from the understanding of the range image, which is a different data representation compared to all other predecessors of the extraction module. Our proposal for the feature extraction module states that:

1. Range Image has the ability to be treated as an image, therefore a Convolutional Neural Network (CNN) is advisable instead of a Multilayer Perceptron (MLP) to perform an analysis that considers the neighborhood and pixel location, instead of as an unordered list of points.

2. The concatenation of actual point and relative neighbors should be done at the beginning, since the point information should have more weight than the neighbors. If actual point is concatenated later in the network, the number of features can dim out its relevance.

With these ideas, the proposal for the updated feature extraction module is shown in figure 30

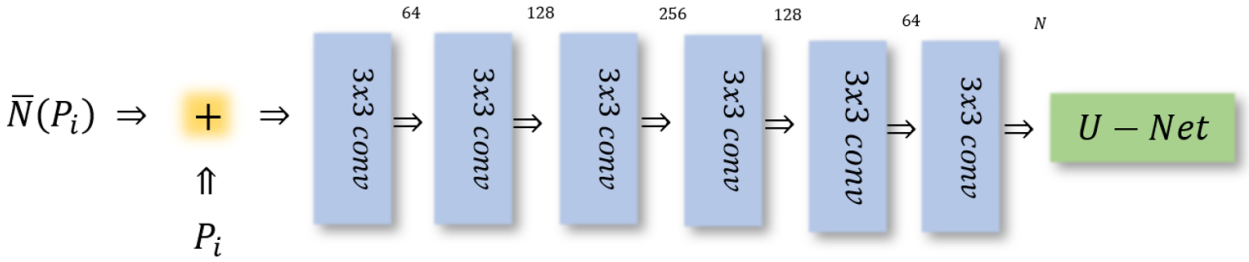


Figure 30: Architecture of the proposed feature extraction module

Since there cannot be a dimension reduction before entering U-Net, there are no pooling layers in the feature extraction module. The number of features learned in each layer are similar to original LU-Net and the input considers all 5 channels of points and relative neighbours: x,y,z, reflectance and distance.

8 Results

8.1 Study of parameters

Table 3: Results on an RI level evaluation in validation set. Best scores are marked in bold and second best are underlined

	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign	Mean
Original	91	4.1	21.8	45.6	23.2	23.2	43.7	0	92.8	36	78	0	83.7	48.3	82.6	53.4	70.5	53.1	19.9	45.8
V2	92	12.1	19.6	**61.6**	25.9	33	52.1	0	92.5	31.1	78.2	**2.2**	83.2	43.8	82.1	55.1	69.2	54.5	20.7	47.8
V3	91.3	6.8	27.8	52.6	27.6	38.8	59.8	0	92.7	32.4	78.8	0.2	84.1	48.2	81.9	57.2	68.3	57.6	26.1	49.1
V4	**92.3**	**40.9**	**45.4**	60.9	**30.2**	**56.5**	**64.3**	0	**94**	**33.2**	**80.8**	1.4	**85.3**	**52**	**85.9**	**63.1**	**74.8**	**60.8**	**37.4**	**55.7**

The Table 3 shows the validation scores for all the classes and mean Intersection Over Union doing a Range-Image level comparison. IoU for motocyclist is zero because this class is not present in the validation sequence, but we still included it in the mean calculation to keep the same classes when analyzing the results in the test set.

V4 column in table 3 is the results of the best configuration achieved up to the publication of this document, which is a sum of several small improvements throughout the first two thirds of the internship period, including V2 and V3 proposals.

For being able to compare our results to other state-or-the-art methods that use SemanticKITTI banchmark, is necessary to propagate predictions of the Range Image and label the point cloud. This will also give information about how accurate is the propagation process. This evaluation was performed through the SemanticKITTI online competition, making predictions for sequences 11 to 21.

Table 4: Results on test set comparing with other algorithms. In bold are the highest results per class and underlined the second highest. RangeNet21 corresponds to a version of this algorithm with 21 layers. RangeNet53++ has 53 layers and a post-processing for propagation.

	width	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign	Mean
SqueezeSeg	2048	68.3	18.1	5.1	4.1	4.8	16.5	17.3	1.2	84.9	28.4	54.7	4.6	61.5	29.2	59.6	25.5	54.7	11.2	36.3	30.8
SqueezeSegV2	2048	82.7	21	22.6	14.5	15.9	20.2	24.3	2.9	88.5	42.4	65.5	18.7	73.8	41	68.5	36.9	58.9	12.9	41	39.6
RangeNet21	2048	85.4	26.2	26.5	18.6	15.6	31.8	33.6	4	91.4	57	74	26.4	81.9	52.3	77.6	48.4	63.6	36	50	47.4
RangeNet53++	1024	90.3	20.6	27.1	25.2	**17.6**	29.6	34.2	7.1	90.4	52.3	72.7	22.8	83.9	53.3	77.7	52.5	63.7	43.8	47.2	48
RangeNet53++	2048	**91.4**	25.7	34.4	**25.7**	23	38.3	38.8	4.8	**91.8**	**65**	**75.2**	**27.8**	**87.4**	**58.6**	80.5	55.1	64.6	47.9	**55.9**	52.2
Ours	1024	90.1	**41.7**	**34.6**	16.5	16.6	**50.1**	**43.6**	**7.7**	90.5	59.7	73.9	21.6	87.2	54.9	**83.1**	**61**	**68.1**	**49.3**	51	**52.7**

Table 5: Number of parameters per architecture. Only V4 includes an update in the architecture. Last column shows the training time only on the network (no pre or post-processing)

	Number of parameters	Seconds/RI
Original LU-Net	23402053	0.067055827
V2	23402053	0.067055827
V3	23402053	0.071154159
V4	24168888	0.10997988

8.2 Comparison to state of the art

In table 4 we listed a couple of algorithms that work with Range Image as well and are considered key methods for point cloud segmentation [19]. In this table we included the width of the input Range Image since it is a variable that has a direct impact on the number of parameters of the Network and in general, the amount of data handled by the system. In bold numbers we mark the scores that are an improvement compared to other methods, inclouding the mean Intersection Over Union.

Our method outstands in classes like *bicycle* and *person* having increments of 16 and 11.8 percent respectively. On the other hand, our approach scores inconsistently low in classes like *truck* and *other-ground*, suggesting that there should be a review on how the model is learning these classes. The classes *car* and *road* do not have a score improvement unlike the trend on newer methods. These are the most frequent on the dataset and are present in most of the scans.

8.3 Discussion

Tabel 4 shows that our method out-stands some of the reference algorithms for point cloud segmentation using Range Image, and this is achieved with a smaller size representation. We use as comparison two versions of SqueezeSeg as well as a combination of RangeNet architectures. RangeNet21 is a version with 21 layers and RangeNet53++ has 53 layers and a fast propagation.

Our architecture is considerably smaller compared to the other models (figure 17), since ours is formed by a 6-layer network for the feature extraction module (figure 30) and a 5-levels deep U-Net as shown in figure 19. The feature extraction module is the key on maintaining a smaller network with competitive results. The idea behind this module is to extract features going deeper in a CNN but without pooling layer so that the image dimensions is preserved, therefore, there is no risk of loosing information on down-sampling. Commonly this type of architecture would suggest that the number of parameters to increase considerably, but since the module is only 6 layers deep is not the

case. In addition, the use of this module enables the possibility of using a simple U-Net with no intermediate or extra modules. The number of parameters (trainable parameters in TensorFlow) per version is shown in Table 5. The V4 architecture has an increment of only $\sim 3.27\%$ parameters, but with an increment of $\sim 10\%$ of mIoU as shown in Table 3. Table 5 also shows the seconds per Range Image in training, measured from the start of the feature extraction module to the range image segmentation. No pre or post-processing was added in this measure so these values are useful for an intern comparison of the architectures and they should not be used as a reference for other algorithms. This table highlights the link between the number of parameters and processing time.

Figure 31 shows the confusion matrix of the prediction made with V4 on the validation set, showing the type of mistakes done by our Network. A class that is scoring low is *other-vehicle* but confusion matrix states that is normally labeled as *car*, a class that has similar properties. To improve this result the team would have to analyze the differences between these two classes when projected to the Range Image, making sure that the features that make them different are preserved.

Even if class *motorcyclist* is not present in the validation set, the confusion matrix still shows that the system used this label wrongly in objects like *person*, and *motorcycle*. This class is problematic for other architectures as well, mainly due to the lack of training data for this class and not the features of the motocyclists in the sequences. The proof are the classes *bicycle* and *bicyclist* that are commonly thin (and therefore challenging) regions to label, but since this class is present in most of the sequences, the system learns to identify them.

The update in the loss function as well as the architecture, caused a behavioural change in the global focal loss of training data. Figure 32 shows the train loss for V2 and Fig 33 shows it for V4. Both using smoothness factor of 0.75. V2 loss follows a stable reducing pattern while V4 oscillates between 0.6 and 0.3 without any periodicity. This suggests that the system still needs could stabilize with more iterations and even achieving a lower loss if trained longer.

Loss function is strongly linked to hyper-parameters of the model such as learning rate and batch size. Also, each optimization algorithm has intern variables that the user should tune for a better performance.

Original LU-Net (as well as our update) uses Adam Optimizer [27] that requires input for β_n and ϵ. Until the latest configuration we used the default values for these parameters, but will be studied as well for checking their impact in instability of the loss.

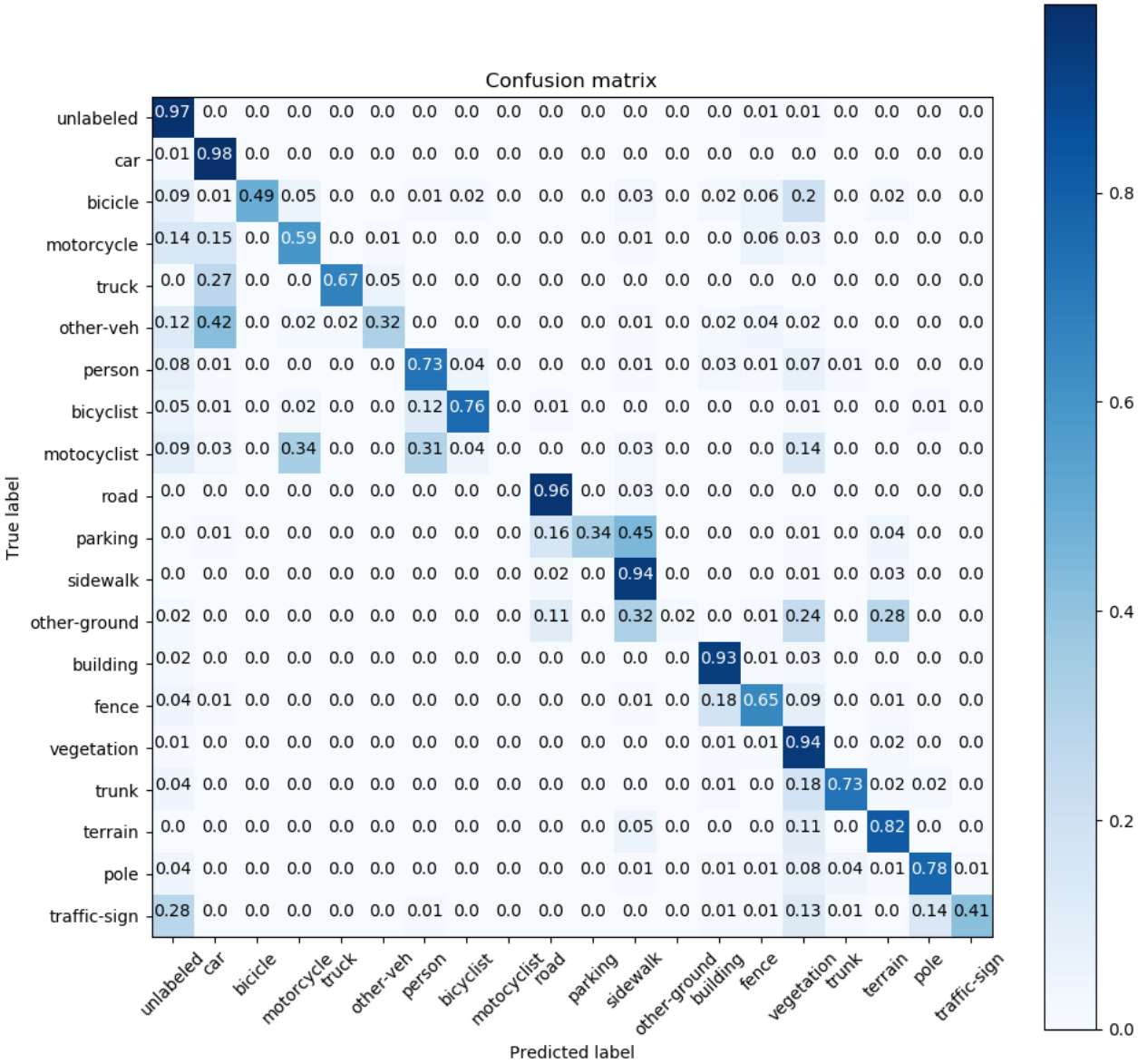


Figure 31: Confusion Matrix for predicitons using V4 architecture

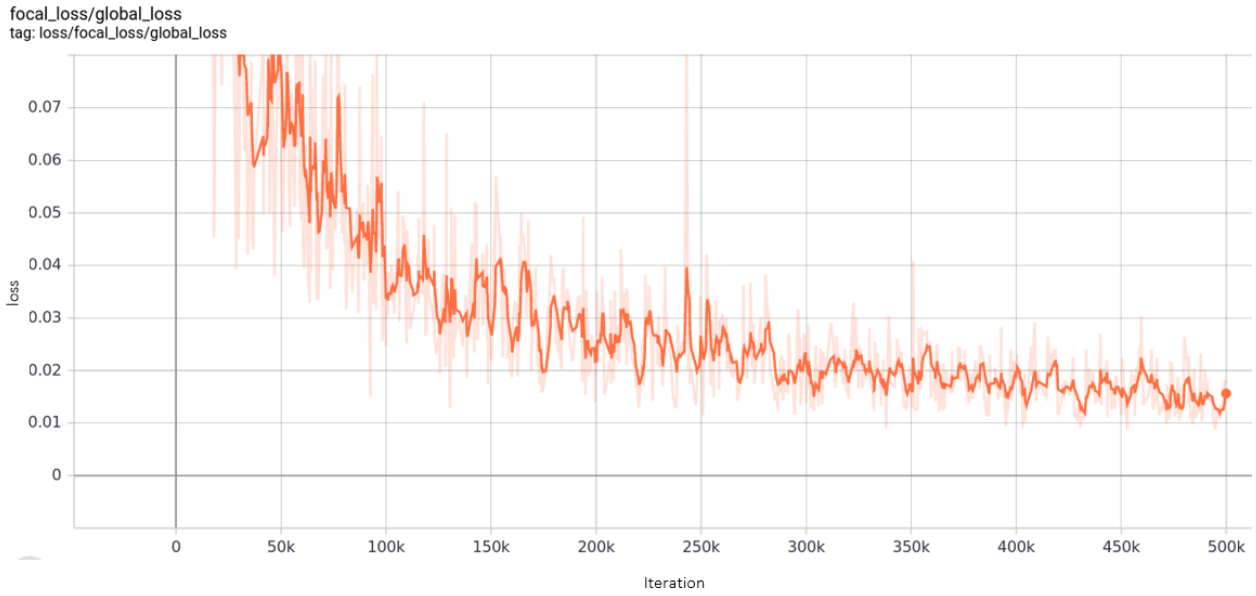


Figure 32: Train loss for V2

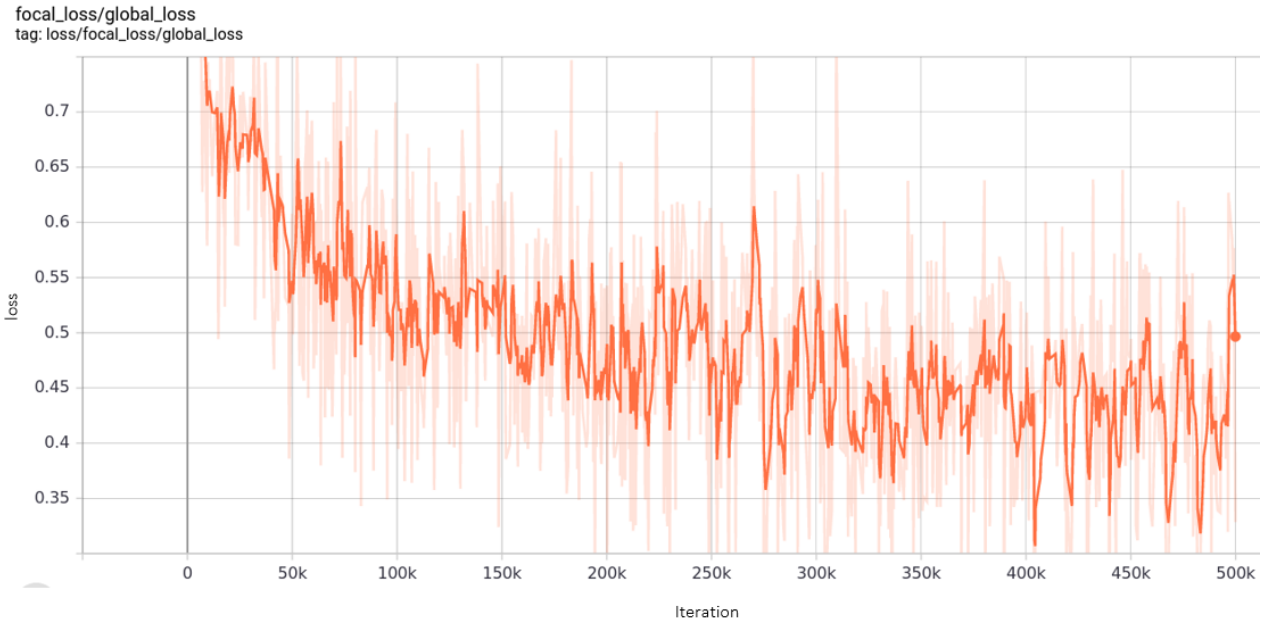


Figure 33: Train loss for V4

9 Future Work

As the internship program continues for more than one month after the submission of this document, the team will focus in specific task considering the current results:

1. Although the idea of this Network is to be small in architecture and input data, a future task is to extend this to Range Image of width 2048 to check the impact on the mIoU and measure if the increment of resources compensate the score improvement.

2. The accuracy of a Deep Learning system depends in great scale to the way the data is presented and used in the learning process. With this in mind the team can do a deeper analysis of the training sequences to:

   - Check the distribution of the training data.
   - Perform data augmentation based on problematic classes frequency.
   - Weight labels considering the errors in confusion matrix.

3. Study the possibility to do a multi-domain analysis, including other sensors available in the benchmark e.g. camera.

4. Test different hyper-parameters of the Network to increase the scores of the V4 architecture, such as U-Net depth, number of features learned by the Feature Extraction Module, learning rate, etc.

5. Explore the expansion of V4 architecture to include intermediate steps in U-Net for higher scores.

The core intention of the research was originally to adapt the LU-Net Architecture into SemanticKITTI dataset and work different proposals to increase its accuracy. The team managed to accomplish that objective and noticed the potential of this network. Figure 34 shows the list of the top scoring methods, extracted from the semantic KITTI website. (`http://www.semantic-kitti.org/tasks.html`). Our results would be listed in 7th place by the publication of this document, above RangeNet++. Some of these methods are very recent and were not published by the start of the internship. This reflects how relevant and dynamic the topic currently is.

The main advantage in our approach compared to the others in the list is the simplicity of the model, the reduced number of parameters/layers and competitive accuracy with range image

of width 1024. Future work will be orientated on reviewing new architectures and include in our model methods that could be compatible. This should be careful reviewed in order to avoid making the network too complex and loosing its simplicity.

Approach	Paper	Code	mIoU	Classes (IoU)
KPConv			58.8	
SqueezeSegV3			55.9	
3D-MiniNet			55.8	
SalsaNext			54.5	
PolarNet			54.3	
RandLA-Net			53.9	
RangeNet++			52.2	
LatticeNet			52.2	

Figure 34: Current State of the art on SemanticKITTI challenge

If the results are sufficient to stand against state-or-the-art methods, the team would consider to work on a publication and do follow up on this architecture for future internship programs.

10 Conclusion

After an exploratory process with different intermediate tests, we propose an architecture update of the Original LU-Net that remains simple and competitive with state-of-the-art methods. We achieved this by adapting the algorithm to SemanticKITTI, a more challenging dataset that is the current reference for point-cloud segmentation in automotive field. By these updates, the network improved by around 10% the score of the original version. The network also manages to score better than the reference network RangeNet++ that uses a larger Range Image representation and amount of layers. The team will continue to research on possible updates of the architecture and training data handling.

References

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

[2] Geiger, A., Lenz, P., & Urtasun, R. (2012, June). Are we ready for autonomous driving? the kitti vision benchmark suite. In 2012 IEEE Conference on Computer Vision and Pattern Recognition (pp. 3354-3361). IEEE.

[3] Menze, M., & Geiger, A. (2015). Object scene flow for autonomous vehicles. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3061-3070).

[4] Wu, B., Wan, A., Yue, X., & Keutzer, K. (2018, May). Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In 2018 IEEE International Conference on Robotics and Automation (ICRA) (pp. 1887-1893). IEEE.

[5] Wu, B., Zhou, X., Zhao, S., Yue, X., & Keutzer, K. (2019, May). Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In 2019 International Conference on Robotics and Automation (ICRA) (pp. 4376-4382). IEEE.

[6] Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., & Gall, J. (2019). SemanticKITTI: A dataset for semantic scene understanding of lidar sequences. In Proceedings of the IEEE International Conference on Computer Vision (pp. 9297-9307).

[7] Milioto, A., Vizzo, I., Behley, J., & Stachniss, C. (2019). Rangenet++: Fast and accurate lidar semantic segmentation. In Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS).

[8] Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention (pp. 234-241). Springer, Cham.

[9] Biasutti, P., Lepetit, V., Aujol, J. F., Brédif, M., & Bugeau, A. (2019). LU-Net: An Efficient Network for 3D LiDAR Point Cloud Semantic Segmentation Based on End-to-End-Learned 3D Features and U-Net. In Proceedings of the IEEE International Conference on Computer Vision Workshops (pp. 0-0).

[10] Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In Proceedings of the IEEE international conference on computer vision (pp. 2980-2988).

[11] Sudre, C. H., Li, W., Vercauteren, T., Ourselin, S., & Cardoso, M. J. (2017). Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In Deep learning in medical image analysis and multimodal learning for clinical decision support (pp. 240-248). Springer, Cham.

[12] Shen, C., Roth, H. R., Oda, H., Oda, M., Hayashi, Y., Misawa, K., & Mori, K. (2018). On the influence of Dice loss function in multi-class organ segmentation of abdominal CT using 3D fully convolutional networks. arXiv preprint arXiv:1801.05912.

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

[14] Czichos, H. (2018). Measurement, Testing and Sensor Technology. Springer International Publishing.

[15] Reymann, C., & Lacroix, S. (2015, September). Improving LiDAR point cloud classification using intensities and multiple echoes. In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 5122-5128). IEEE.

[16] Skarbek, W., Koschan, A., Bericht, T., & Veroffentlichung, Z. (1994). Colour image segmentation-a survey.

[17] Skansi, S. (2018). Introduction to Deep Learning: from logical calculus to artificial intelligence. Springer.

[18] Aggarwal, C. C. (2018). Neural networks and deep learning. Springer, 10, 978-3.

[19] Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., & Bennamoun, M. (2019). Deep Learning for 3D Point Clouds: A Survey. arXiv preprint arXiv:1912.12033.

[20] Skansi, S. (2018). Introduction to Deep Learning: from logical calculus to artificial intelligence. Springer.

[21] Hahner, M., Dai, D., Liniger, A., & Van Gool, L. (2020). Quantifying Data Augmentation for LiDAR based 3D Object Detection. arXiv preprint arXiv:2004.01643.

[22] Landrieu, L., & Boussaha, M. (2019). Point cloud oversegmentation with graph-structured deep metric learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 7440-7449).

[23] Maddern, W., Pascoe, G., Linegar, C., & Newman, P. (2017). 1 year, 1000 km: The Oxford RobotCar dataset. The International Journal of Robotics Research, 36(1), 3-15.

[24] Neto, A. M., Victorino, A. C., Fantoni, I., & Ferreira, J. V. (2013, January). Real-time collision risk estimation based on Pearson's correlation coefficient. In 2013 IEEE Workshop on Robot Vision (WORV) (pp. 40-45). IEEE.

[25] `http://www.cvlibs.net/datasets/kitti/eval_semseg.php?benchmark=semantics2015`

[26] Solis, E. B., Neto, A. M., & Huallpa, B. N. (2015, October). Pearson's correlation coefficient for discarding redundant information: Velodyne lidar data analysis. In 2015 12th Latin American Robotics Symposium and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR) (pp. 116-119). IEEE.

[27] Rummelhard, L., Paigwar, A., Nègre, A., & Laugier, C. (2017, June). Ground estimation and point cloud segmentation using SpatioTemporal Conditional Random Field. In 2017 IEEE Intelligent Vehicles Symposium (IV) (pp. 1105-1110). IEEE.

[28] Ibtehaz, N., & Rahman, M. S. (2020). MultiResUNet: Rethinking the U-Net architecture for multimodal biomedical image segmentation. Neural Networks, 121, 74-87.