UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

UNIVERSITY OF PADOVA

Department of Information Engineering

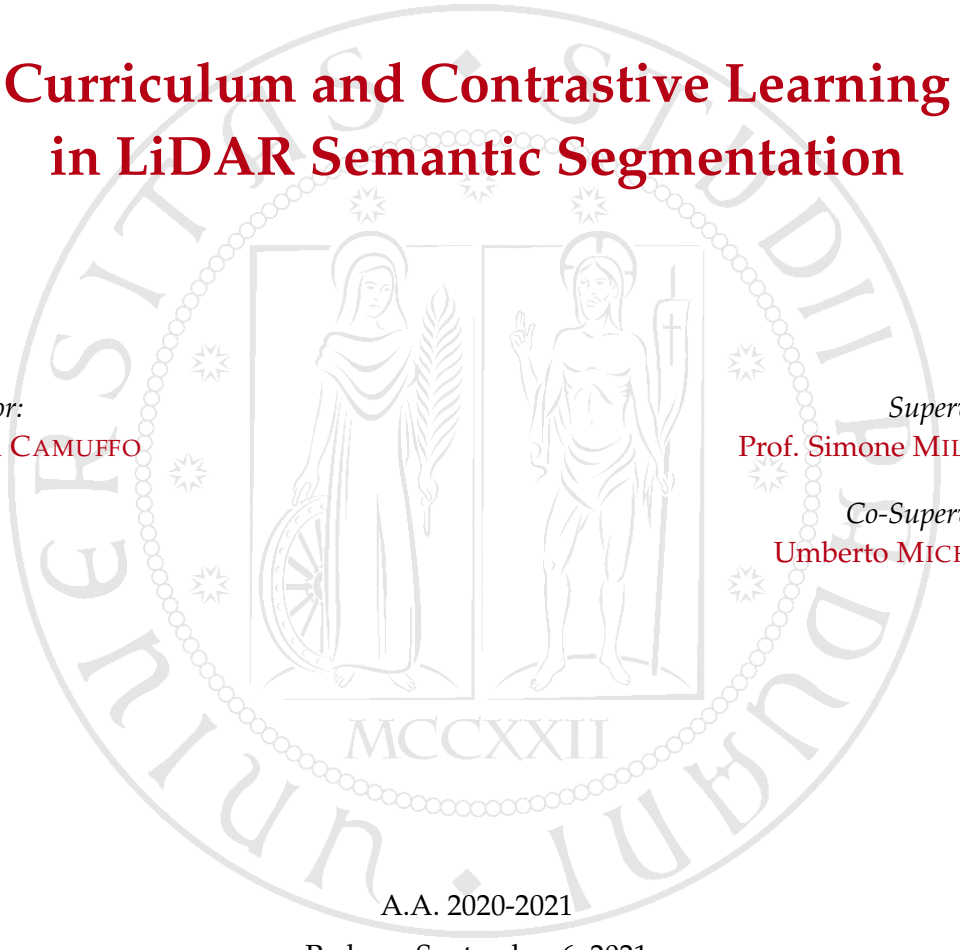
Master Degree in ICT for internet and multimedia

Curriculum and Contrastive Learning in LiDAR Semantic Segmentation

Author:
Elena CAMUFFO

Supervisor:
Prof. Simone MILANI

Co-Supervisor:
Umberto MICHIELI



A.A. 2020-2021

Padova, September 6, 2021

Alla mia famiglia
e a chi mi ha sostenuto in questo percorso
dandomi la forza di non mollare mai.

"Simplicity is the end result of long, hard work, not the starting point."

- Frederick Maitland -

"I am enough of an artist to draw freely upon my imagination. Imagination is more important than knowledge. Knowledge is limited. Imagination encircles the world."

- Albert Einstein -

Abstract

Recent advances in technologies such as autonomous driving and robotics have highlighted the growing need for precise environmental perception. LiDAR Semantic Segmentation has recently attracted the attention of industrial and academic research due to its ability to accomplish fine-grained scene understanding and act directly on raw content provided by sensors. The task has been approached in a variety of ways, so far. RandLA-Net architecture has been selected for this work as it represents a powerful and lightweight solution to deal with large-scale data. The goal of developing solutions that aim at optimizing the learning process rather than focusing on the architecture has been considered, showing how different learning techniques can be used to improve the performance of the model. The methods include Curriculum Learning and Contrastive Learning, which aim at having a better separation between different classes, providing a better understanding of the scene content. Coarse-to-Fine strategies have been developed as well as regularization methods in order to take into account a hierarchical organization of the classes. The results we obtained outperform the state of the art, achieving an improvement of 1.5% in terms of mIoU with different methods. They proved their efficiency by providing a better balance of classes and a faster convergence to optimal performance.

Sommario

Lo sviluppo recente di tecnologie come la guida autonoma e la robotica hanno evidenziato la crescente necessità di una più accurata percezione dell'ambiente da parte dei dispositivi. La segmentazione semantica LiDAR ha recentemente attirato l'attenzione della ricerca industriale e accademica, grazie alla sua capacità di comprendere in modo preciso il contenuto della scena percepita e agire direttamente sui dati grezzi forniti dai sensori. A tale scopo sono state proposte diverse soluzioni algoritmiche. RandLA-Net è stata scelta come architettura per questo lavoro, in quanto rappresenta una soluzione potente e leggera per gestire dati su larga scala. L'obiettivo della tesi è sviluppare una procedura per ottimizzare il processo di apprendimento di RandLA-Net invece di andare a modificare la struttura della rete, mostrando come diverse tecniche di apprendimento possano migliorare le prestazioni a parità di modello, dataset e risorse di calcolo. I metodi includono l'apprendimento curriculare (Curriculum Learning) e l'apprendimento contrastivo (Constrastive Learning), i quali mirano ad ottenere una migliore separazione tra le diverse classi, basandosi su un raggruppamento gerarchico a posteriori e fornendo una migliore comprensione del contenuto della scena. Strategie Coarse-to-Fine e metodi di regolarizzazione sono stati impiegati per tenere conto di un'organizzazione gerarchica delle classi. I risultati dei nostri metodi superano lo stato dell'arte, ottenendo un miglioramento di 1.5% in termini di mIoU con diversi metodi. Hanno dimostrato la loro efficienza, fornendo un migliore equilibrio delle classi e una convergenza più rapida verso le prestazioni ottimali.

Contents

List of Figures

List of Tables

1 Introduction

In this chapter a brief overview of the problem and the contribution of our work is given.

* * *

Recent advances in technologies such as autonomous driving and robotics have highlighted the growing necessity for precise environmental perception. Moreover, the widespread use of such technologies as well as the rapid rate of development have pointed out the need for fast and precise solutions that can accommodate even scenarios in which the safety of people and living entities must be taken into account.
In parallel, the quick evolution of deep learning technologies and the increasing power of Graphic Processing Units (GPUs), raised thanks to the spread of graphics and rendering solutions, have given birth to a solid framework for research and has enabled us to deal even with burdensome Computer Vision (CV) tasks in a straightforward way.

Several computer vision tasks providing semantic scene understanding are used in robotics and autonomous driving applications (Figure 1.1). Specifically, Point Cloud Semantic Segmentation (PCSS) task has recently attracted the attention of industrial and academic research due to its ability to accomplish fine-grained scene understanding and act directly on the raw content provided by sensors; in fact, the most common data employed are point clouds generated through LiDAR sensors.
However, when dealing with this kind of data, there are several difficulties that must be addressed. For example, point clouds are often sparse data structures, in contrast to images where pixels are regularly distributed. This peculiarity makes point cloud processing quite challenging because of high redundancy, uneven sampling density, and lack of explicit structure. Moreover, the utilization of LiDAR-generated point clouds has revealed other quality-degrading problems, such as occlusions, lighting, and weather conditions.

Also problems related to datasets size and compositions represent a limitation in point cloud processing. In fact, LiDAR datasets sizes highly affect the performance of deep learning models jointly with the heterogeneity and classes balancement in their scenes. Moreover, due to the difficulties in acquiring and labeling three-dimensional data, the number of LiDAR datasets now accessible is significantly lower than that of image datasets. In this context, SemanticKITTI dataset has been chosen for this work, providing large-scale data with a sufficient amount of diversity in its scenes.

The task of LiDAR semantic segmentation has been approached in a variety of ways so far. Unsupervised algorithms that simply grouped points into regions according to spatial criteria were among the earliest techniques proposed. Later, supervised approaches, such as conventional machine learning and Deep Learning (DL) techniques, were introduced.

Among the currently employed techniques are methods that simplify the task by projecting point clouds onto bidimensional images. Other largely adopted techniques involve the discretization of point clouds using three-dimensional volumetric structures, like voxels or octrees, that resemble the image structures in a three-dimensional environment. However, both these methods imply the discretization of the data, which introduces undesired artifacts and causes an information loss. These issues represent relevant problems for applications where a considerable level of precision is required, for example in self-driving cars, where people's safety needs to be ensured. Therefore, a class of methods that act directly on the raw point clouds have been chosen for this work. In particular, RandLA-Net [15] architecture has been selected, as it represents a powerful and lightweight solution to deal with large-scale data.

The challenging task of optimizing point cloud processing is still an open problem, since algorithms are extremely computationally heavy, and data require large memory space. LiDAR semantic segmentation has been enabled only during the last few years thanks to the optimization in the GPUs computational capabilities; nevertheless, it is still in its early stages, and further developments need to be studied to improve its applicability.

Most of the existing approaches improved the performances of LiDAR semantic segmentation by enhancing the deep learning architecture. However, this would need a significant investment of resources, which could not be feasible in most scenarios. The objective of this work is to develop solutions that aim at optimizing the learning process, rather than focusing on the architecture.

(a) Robotics. (b) Self-driving cars. (c) Remote sensing.

Figure 1.1: Examples of application of semantic scene understanding methods.

1.1 Proposed Contribution

Several learning techniques will be considered in this research work to improve LiDAR semantic segmentation performance. Some initial analyses have been led on the dataset, highlighting some interesting insights related to the dataset itself and the dataset related with the model outcome. According to these analyses, the strategies to be adopted later have chosen.

Among them, Curriculum Learning techniques are explored. This approach consists in emulating the natural human learning process presenting first the easiest examples and refining the knowledge on more difficult ones at the advanced stages. This way the model is carried to gradually learn the content of the data in a natural manner.
Moreover, also Contrastive Learning methods have been used. This latter approach concerns techniques that aim at having a better separation among different classes, providing a better understanding on the scene content.

In this work novel contributions exploiting these learning strategies are presented. We show that our approaches outperform the state of the art both in terms of overall performance and convergence time.
Particulary, all the methods are founded on the idea of a hierarchy that sees semantic classes grouped depending on their category: people are separated from road as well as from vehicles and vegetation, in order to provide first a broad labelling and then a refinement, because a totally wrong misclassification, like confusing a *traffic-sign* with a *person* is more dangerous than confusing a *car* with a *truck*. The developed approaches include Coarse-to-Fine (C2F) strategies and training regularization methods.

The thesis is organized in the following way. In Chapter 2 an overview of the theoretical background concerning semantic scene understanding and semantic segmentation is provided. A detailed explanation of the metrics and components of the task is given, to move finally the attention on the specific task of LiDAR semantic segmentation. Chapter 3 describes the structures used to deal with three-dimesional data, introducing also the LiDAR datasets and the problematics related. Chapter 4 provides the analyses led on SemanticKITTI dataset [2] for our work while Chapter 5 introduces all the stategies employed. Finally results are given in Chapter 6 and conclusions are drawn in Chapter 7.

2 Semantic Scene Understanding

In this chapter an overview of the methods and the state of the art approaches for semantic scene understanding is presented. LiDAR semantic segmentation and the main architectures employed are introduced. Particular attention is given to the architecture used in this work.

* * *

Semantic scene understanding is an essential task for many computer vision applications. The main aim is to identify and classify each element of an acquired scene within a general framework, i.e., analyze objects in context with respect to the three-dimensional structure of the scene, its layout, and the spatial, functional, and semantic relationships between objects. Among its major applications are autonomous driving systems, where perceiving the overall content of the environmental scene is essential for decision making and safety control [19].

The task includes several computer vision disciplines that provide scene understanding at different levels of resolution with different purposes. Particularly, fine-grained road scene understanding provided by semantic segmentation (Section 2.1.3) is necessary to distinguish drivable and non-drivable surfaces and to infer its functional properties, like parking areas and sidewalks. Currently, such understanding is primarily generated in advance using surveying vehicles and represented in so-called high resolution maps [2].

2.1 Disambiguation

Various levels of scene understanding can be achieved using different computer vision approaches. The choice mainly depends on the application, the scale, the level of detail and properties of objects. Most of the architectures (e.g., PointNet [30] for LiDAR semantic segmentation, Section 2.3.3) provide solutions to more than one of these tasks at the same time. The main categories are overviewed in the following sections.

2.1.1 Image Classification

The most general and simple task for computer vision is image classification, which has found wide diffusion thanks to its applicability in various fields. The task consists in determining the content of an image (or point cloud in the case of three-dimensional data) and categorizing it with a single label (Figure 2.1b).

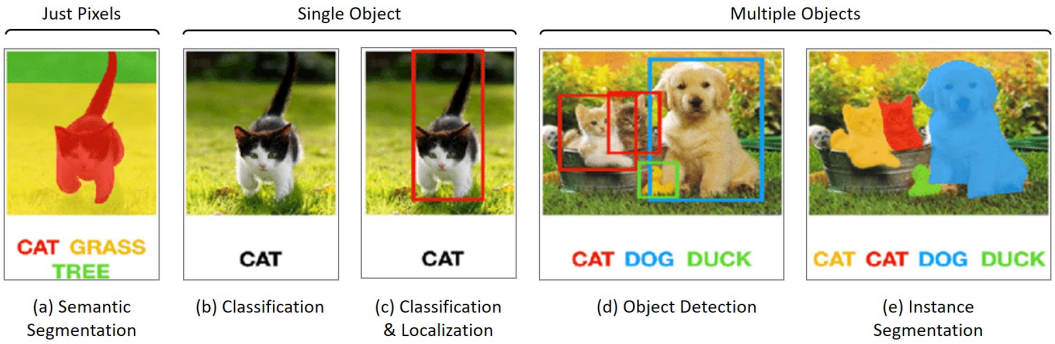


Figure 2.1: Computer vision tasks in bidimensional space.

Formally, the classification problem can be defined as follows. Given an image, i.e., a set of pixels, $X = \{x_1, x_2, x_i, \ldots, x_N\}$ with $x_i \in \mathbb{R}^2$ and a candidate label set $Y = \{y_1, y_2, \ldots, y_k\}$ assign the whole point set X to only one of the k labels. The same definition can be applied in a three-dimensional context, where instead of an image composed of pixels we have a point cloud, i.e., a set of ordered three-dimensional points where $x_i \in \mathbb{R}^3$.

On the other hand, effective classification requires the image to be uniform in content and contain a single object to be classified. If a more accurate understanding of the scene is required, some other computer vision techniques need to be applied. For example, classification can be done jointly with *localization* (Figure 2.1c). This technique is widely employed in face recognition systems, where a precise analysis can be led if the person's face is localized in advance (e.g., by means of algorithms such as *Viola and Jones* [42]).

2.1.2 Object Detection

When we want to classify more than one element in a scene, we need to employ object detection techniques. In particular, object detection is an extension of *classification and localization* task, i.e., it localizes all the objects in the scene, encapsulating each into a bounding box (Figure 2.1d). Therefore, it provides their geometric location and orientation in addition to their semantic instance label.

Each box is commonly represented as (x, y, h, w, θ, c). The parameters (x, y) denotes the object (bounding box) center position, (h, w) represents the bounding box size with width and height, and θ is the object orientation. Finally, c represents the semantic label of this bounding box (object).

Object detection is a fundamental task that allows to have a more precise understanding of the scene and all the objects within it. However, the encapsulation of objects in their bounding boxes represents a limitation, as they contain also part of the background and several bounding boxes can overlap if the respective objects are close one another. If a more

precise shape of the objects is required, segmentation techniques have to be employed.

2.1.3 Semantic Segmentation

The task of semantic segmentation is a fundamental research challenge in image processing. It is one of the finest approaches towards semantic scene understanding as it provides a classification at the pixel level, i.e., each pixel in the image is associated to an object, identified by a semantic label (Figure 2.1a).

Formally, semantic segmentation is the process to cluster the input data into several homogeneous regions, where points in the same region have some identical attributes. Each input point is predicted with a semantic label, i.e., given a set of N ordered pixels $X = \{x_1 x_2, \ldots, x_N\}$, $x_i \in R^2$ and a candidate label set $Y = \{y_1, y_2, \ldots, y_k\}$, assign each input point to one of the k semantic labels.

Semantic segmentation provides a labelling that is equal for whatever object is in the scene. For example, if there are two *cats* like in Figure 2.1d, semantic segmentation provides that the pixels associated with the two cats have the same label (i.e., *cat*) and belong to the same category. Conversely, another computer vision task, named *instance segmentation*, provides a label for each instance of each class within a scene (Figure 2.1e).

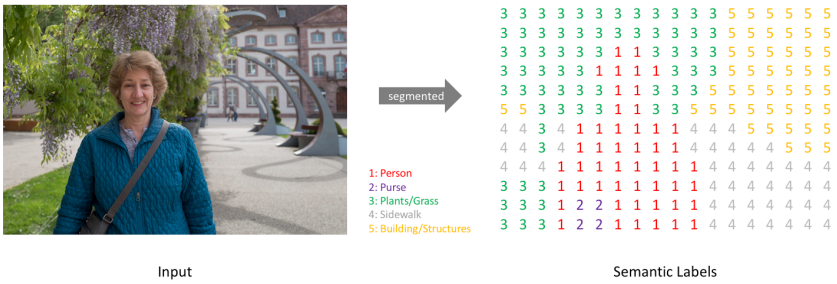
As for image classification and object detection, semantic segmentation can be extended to a third depth-related information, taking into consideration three-dimensional points instead of pixels.
Moreover, when dealing with objects in a three-dimensional environment (e.g., representing people) semantic segmentation can be applied to a single object with the task of *part segmentation* [20]. Part segmentation produces pixel-level semantic annotations that indicate fine-grained object parts instead of an overall object label.

2.2 Semantic Segmentation

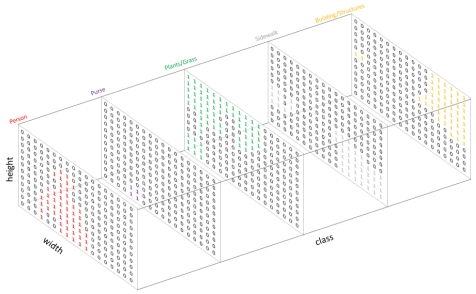
In this work, we perform semantic segmentation applied to three-dimensional data in an autonomous vehicles' scenario. This section provides an overview of the task, firstly applied to images and later to three-dimensional data.

2.2.1 Image Semantic Segmentation

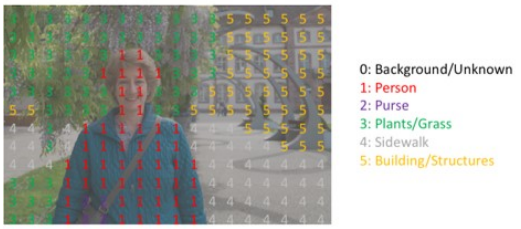
As anticipated, image semantic segmentation provides pixel-wise labelling, often referred to as *dense prediction*. The goal is to take either a RGB color image ($h \times w \times 3$) or a grayscale image ($h \times w \times 1$) and output a segmentation map where each pixel contains a class label represented as an integer ($h \times w \times 1$).
The target is produced by one-hot encoding the class labels and the prediction can be collapsed into a segmentation map (Figure 2.2c) by taking the *argmax* of each channel-wise pixel vector (Figure 2.2b) [27] [11].



(a) Semantic segmentation task representation.



(b) Semantic segmentation mask for each class.



(c) Semantic segmentation pixel labelling.

Figure 2.2: Semantic segmentation process.

2.2.1.1 *Architecture*

The task can be tackled both with classic computer vision algorithms and deep learning architectures, which are usually preferred for their efficiency.
A typical architecture to accomplish the task is composed simply stacking convolutional layers and producing a final segmentation map. This model directly learns a mapping from the input image to its corresponding segmentation through the successive transformation of feature mappings. However, it is quite computationally expensive to preserve the full resolution throughout the network. In fact, earlier layers tend to learn low-level concepts while later layers develop more high-level specialized feature mappings. In order to maintain useful information, the number of feature maps needs to be increased getting deeper in the network.

A popular approach that permits image segmentation models to keep the full image resolution is to follow an *encoder/decoder structure* (Figure 2.9) where a lower-resolution feature map is developed through successively downsampling of the input image and then upsampled to obtain a full-resolution segmentation map.

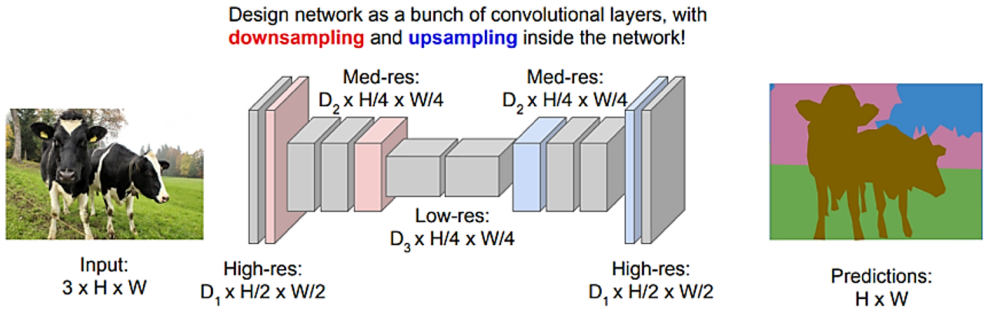


Figure 2.3: Semantic segmentation encoder/decoder structure [23].

2.2.1.2 *Fully Convolutional Network*

Long *et al.* [23] developed in late 2014 the concept of utilizing a fully convolutional network (FCN) trained end-to-end, pixel-to-pixel for image segmentation. The authors propose popular image classification networks (e.g., AlexNet [18]) to serve as the network's encoder module, with a decoder module containing transposed convolutional layers to upsample the coarse feature maps into a full-resolution segmentation map. A pixel-wise cross entropy loss (Section 2.2.3) is used to train the entire network.

However, because the encoder module reduces the resolution of the input, the decoder module struggles to produce fine-grained segmentations.

> *Semantic segmentation faces an inherent tension between semantics and location: global information resolves "what" while local information resolves "where". Combining fine layers and coarse layers lets the model make local predictions that respect global structure* [34].

This conflict is solved by gradually upsampling the encoded representation in stages, adding *skip connections* from prior layers, and summing the two feature maps. Skip connections should provide the necessary details in order to reconstruct accurate shapes for segmentation boundaries and allow us to recover more fine-grain details.

Ronneberger *et al.* improved the fully convolutional architecture by expanding the capacity of the decoder module of the network. They propose the U-Net architecture (Figure 2.4), which consists of a contracting path to capture context and a symmetric expanding path that enables precise localization [34]. This simple architecture has grown to be very popular and has been adapted for a variety of segmentation problems. A similar structure can be found even in models for LiDAR semantic segmentation (Section 2.3).

On the basis of U-Net, many other models have been proposed. For example, convolution blocks have been replaced with *residual blocks* [7], introducing short skip connections

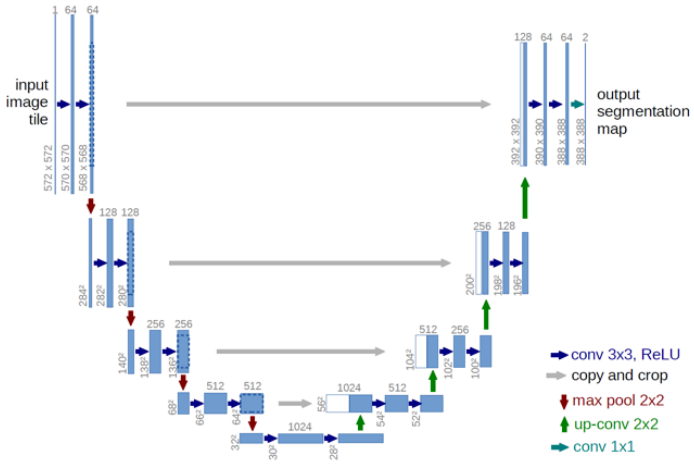


Figure 2.4: U-Net architecture.

(within the block) alongside the existing long skip connections of U-Net. This allows for faster convergence when training and deeper models to be trained.

2.2.1.3 Datasets

A number of common datasets for semantic segmentation has been created in order to help researchers to train new models and benchmark against the state of the art. For indoor environments, PASCAL VOC [8], published in 2012 for a competition, contains 20 object categories and more than 3000 images for training, validation and private testing sets. In 2018 COCO [22] was published, holding 163K images with 91 classes. For the outdoors environment, Cityscapes [6] is worth mentioning, since it is a dataset for autonomous driving, whose classes resembles the ones of SemanticKITTI (Section 3.2.3).

2.2.2 Metrics

When evaluating a standard machine learning model, predictions are usually classified into four categories[1]: true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). However, for the dense prediction task of image segmentation, it is not immediately clear how to evaluate the generated map. In this section the metrics used specifically for this task are overviewed [27] [11].

[1] A true positive is observed when a prediction-target mask pair has an IoU score (Section 2.2.2.3) which exceeds some predefined threshold.
A false positive indicates a predicted object mask had no associated ground truth object mask.
A false negative indicates a ground truth object mask had no associated predicted object mask.

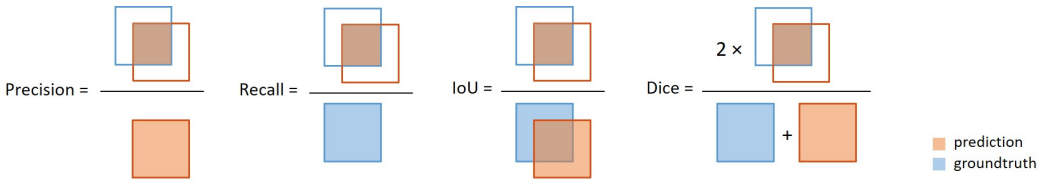


Figure 2.5: Precision, Recall, IoU and Dice coefficient metrics.

2.2.2.1 *Execution time and memory footprint*

Speed or runtime is an extremely valuable metric since the vast majority of systems must meet hard requirements on how much time in performing inference.
Memory usage is another limiting factor for segmentation methods, even if it is less constraining than execution time.

2.2.2.2 *Pixel accuracy*

The main metric, used also for classification and other computer vision tasks (Section 2.1) is the accuracy. In particular for segmentation we referred to it as pixel accuracy.

Pixel accuracy simply reports the percent of pixels in the image which were correctly classified. For $K+1$ classes (typically, K foreground classes and the background) it is defined as:

$$PA = \frac{\sum_{i=0}^{K} p_{ii}}{\sum_{i=0}^{K} \sum_{i=0}^{K} p_{ij}} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

where p_{ij} is the number of pixels of class i predictied as belonging to class j.

When considering the per-class pixel accuracy we are essentially evaluating a binary mask; a true positive represents a pixel that is correctly predicted to belong to the given class (according to the target mask) whereas a true negative represents a pixel that is correctly identified as not belonging to the given class. This metric can sometimes provide misleading results when the class representation is small within the image.

Mean Pixel Accuracy (MPA) is a slightly improved PA in which the ratio of correct pixels is computed in a per-class basis and then averaged over the total number of classes.
Other metrics used also for other tasks are useful in semantic segmentation are precision and recall.

Precision effectively describes the purity of positive detections relative to the ground truth, i.e., of all of the objects that we predicted in a given image, how many of those objects actually had a matching ground truth annotation.

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

Recall effectively describes the completeness of positive predictions relative to the ground truth, i.e., of all of the objected annotated in our ground truth, how many were captured as positive predictions.

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

Usually we are interested into a combined version of precision and recall rates, the **F1 score** metric, which is defined as the harmonic mean of precision and recall:

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{2.4}$$

Dice coefficient is another popular metric for segmentation, which can be defined as twice the overlap area of predicted and groundtruth maps, divided by the total number of pixels in both images.

$$Dice = \frac{2 \cdot |A \cap B|}{|A| + |B|} \tag{2.5}$$

When applied to boolean data the Dice coefficient is essentially identical to the F1 score (Equation 2.4).

2.2.2.3 Intersection over Union

The **Intersection over Union (IoU)** metric, also referred to as the **Jaccard index**, is one of the most commonly used metrics in semantic segmentation. It is defined as the area of intersection between the predicted segmentation map and the ground truth, divided by the area of union between the predicted segmentation map and the ground truth:

$$IoU = J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.6}$$

where A and B denote the ground truth and the predicted segmentation maps, respectively. It ranges between 0 and 1.

This metric is closely related to the Dice coefficient (Equation 5.2) which is often used as a loss function during training (Section 2.2.3). Quite simply, the IoU metric measures the number of pixels common between the target and prediction masks divided by the total number of pixels present across both masks.

Mean-IoU (mIoU) is another popular metric, which is defined as the average IoU over all classes. It is widely used in reporting the performance of modern segmentation algorithms:

$$mIoU = \frac{1}{K+1} \sum_{i=0}^{K} \frac{p_{ii}}{\sum_{j=0}^{K} p_{ij} + \sum_{j=0}^{K} p_{ji} p_{ii}} \tag{2.7}$$

This metric will be widely employed later in this thesis (Chapter 6), in order to evaluate the performances of our methods.

Frequency weighted intersection over union is an improved mIoU which weights each class importance depending on their appearance frequency:

$$FWIoU = \frac{1}{\sum_{i=0}^{K}\sum_{j=0}^{K} p_{ij}} \sum_{i=0}^{K} \frac{\sum_{j=0}^{K} p_{ij} p_{ii}}{\sum_{j=0}^{K} p_{ij} + \sum_{j=0}^{K} p_{ji} p_{ii}} \tag{2.8}$$

2.2.3 Loss functions

Machine learning and mathematical optimization problems are characterized by a loss function (or cost function), which is a function that maps an event or values of one or more variables onto a real number intuitively representing some "cost" associated with the event. An optimization problem seeks to minimize a loss function.
The most common loss function for the task of image segmentation is a pixel-wise cross-entropy loss (Figure 2.6a). This loss examines each pixel individually, comparing the class predictions to the one-hot encoded target labels and then averages over all pixels in the image:

$$\mathcal{L}_{CE} = -\sum_{classes} y_{pred} \cdot \log y_{pred} \tag{2.9}$$

This way the cross entropy loss ensures equal learning for each pixel. This can be a problem if dealing the pixels for each class in the image are unbalanced since training can be dominated by the most prevalent class. To fill this gap, many weighting schemes have been proposed, in order to provide balancement among all the classes.

Another popular loss function is known as the *soft Dice loss* (Figure 2.6b) and based on the Dice coefficient (Equation 5.2). In order to formulate a loss function which can be minimized, we'll simply use:

$$\mathcal{L}_{Dice} = 1 - Dice \tag{2.10}$$

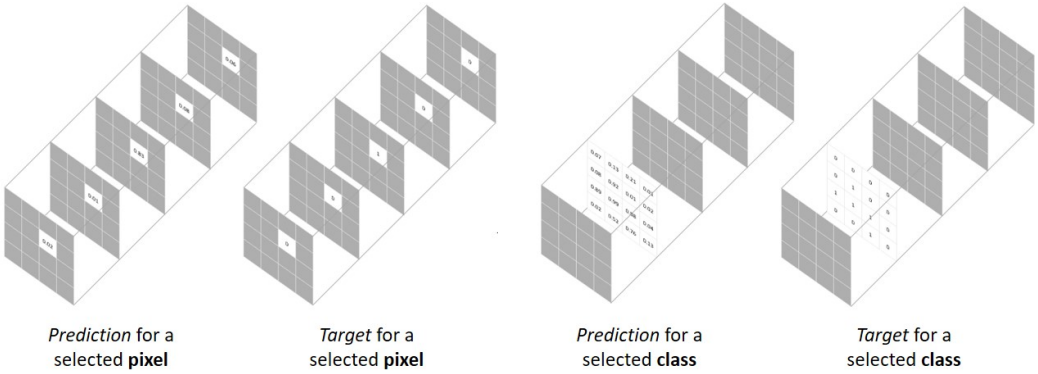
This loss directly uses the predicted probabilities instead of thresholding and converting them into a binary mask.

2.2.4 Three-dimensional Semantic Segmentation

Point cloud semantic segmentation is the three-dimensional form of semantic segmentation, in which regularly or irregularly distributed points in three-dimensional space are used instead of regularly distributed pixels.

Even if point clouds and generally three-dimensional data (dealt in detail in Chapter 3) present many differences with respect to images, the procedures and architetures adopted in 3D semantic segmentation have a very similar behaviour. The main methods are dealt in detail in the following section. Also, the metrics and loss functions are more or less the same of those used in image semantic segmentation (Sections 2.2.2, 2.2.3).

3D semantic segmentation represents a basic and critical task for many applications, such as robotics and autonomous driving: Simultaneous Localization And Mapping (SLAM)



(a) Cross-entropy loss. (b) Dice Loss.

Figure 2.6: Loss functions for semantic segmentation.

need detailed indoor objects for mapping; an autonomous driving system requires object detection, segmentation, and classification to construct a high definition map.

2.3 LIDAR SEMANTIC SEGMENTATION

Light Detection and Ranging or Laser Imaging Detection and Ranging (LiDAR) are particular detection systems which work on the principle of radar, but uses light from a laser. These sensors produce a sparse prediction of the environment composed of point cloud data. LiDAR semantic segmentation refers to 3D semantic segmentation of LiDAR data and has in literature been tackled with different methods and deep learning architectures.

Before effective supervised learning methods were widely applied in semantic segmentation, unsupervised Point Cloud Segmentation (PCS) was a significant task for 2.5D and 3D data (Chapter 3). This task aims at clustering points with similar geometric characteristics without considering semantic information and can be included in the PCSS workflow as a pre-segmentation, influencing the final result [20].

In general, all the approaches adopted can be broadly divided into three categories depending on the way the LiDAR data are dealt. These methods are briefly analyzed in the following sections and some distinctive architectures are presented.

2.3.1 Discretization based models

The first category of methods is based on a discretization procedure. This procedure transforms point clouds into discrete data structures, before feeding them to the deep learning architecture. The data structures employed can be dense like voxels or octrees

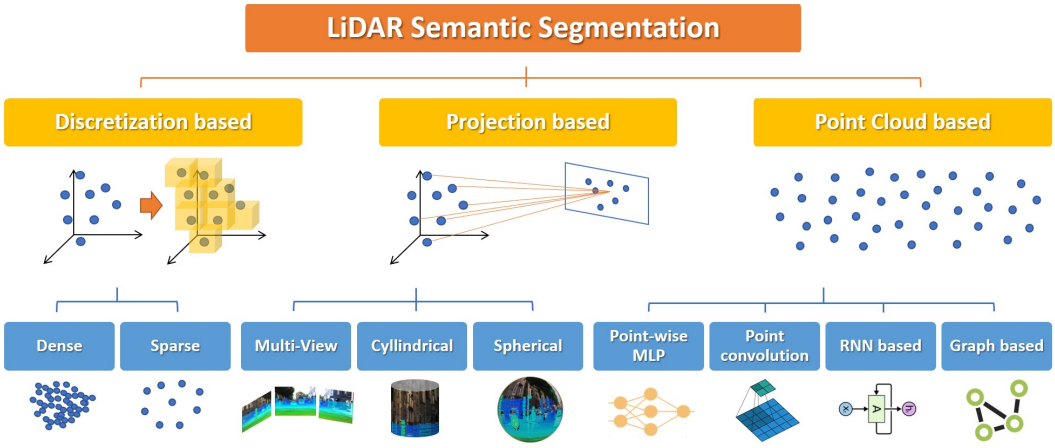


Figure 2.7: Taxonomy of the methods used in LiDAR Semantic Segmentation.

(Section 3.1.1), or sparse like permutohedral lattices [2].

The main advantage of discretization based models is that these kind of structures can be treated as three-dimensional images and dense or sparse convolutions can be easily applied. In the following sections some approaches employing dense and sparse convolutions are dealt.

Dense. The basic idea behind these methods is to divide the space occupied by point clouds into occupancy grids, like voxels (Section 3.1.1), and assign the same label to all the points belonging to the same volume. This procedure allows to produce voxels at different levels of resolution. The advantage of using these data structures is that both the three-dimensional shape and viewpoint can be encoded and voxels can be classified according to the particular condition in occlueded, self-occluded and visible voxels. However the performance is severely limited by the granularity of the voxels and the boundary artifacts caused by the point cloud partition.

Sparse. On the other hand, LiDAR point clouds are naturally sparse, and their respective volumetric representation accounts only for a small percentage of occupied cells. Therefore, many approaches adopted sparse instead of dense convolutional networks.

2.3.2 Projection based models

Another category of methods for LiDAR semantic segmentation consists in projecting the point cloud on a bidimensional structure to infer predictions and remap it back in a later stage. The projection methods are based either on multi-view, spherical or

[2] In mathematics, the permutohedron of order n is an $(n-1)$-dimensional polytope embedded in an n-dimensional space.

cylindrical projections. The deep learning architectures used in these cases are usually well-established convolutional neural networks (CNN) models and pre-trained networks on image datasets[3].

Compared with discretization based models, these methods are able to improve the performance for different tasks (Section 2.1) by taking multi-views of the interest object or scene and then fusing or voting the outputs to make the final prediction. In addition they are efficient in terms of computational complexity. However, projection based methods are approximations of the input data, similarly to discretization-based architectures, and this implies information loss.

2.3.3 Point Cloud based models

To avoid limitations posed by both projection and discretization based methods, e.g, loss of structural information, many methods based on direct processing of point cloud data have emerged and are still emerging nowadays.

Pointwise MLP methods. This kind of methods have been introduced by Qi *et al.* in 2017 with PointNet [30]. This framework represented a novelty in LiDAR semantic segmentation as it achieved considerable results avoiding the use of convolutional networks, but employing a simple architecture, able to learn per-point features by means of shared multi-layer perceptrons (MLPs) and global features using symmetrical pooling functions.

On the base of PointNet, a series of other works have been proposed, e.g., PointNet++ [31] was introduced to tackle the problem PointNet had in capturing local structures introducing a hierarchical network that applies PointNet recursively.
Another pioneer that has developed the method started by PointNet, is RandLA-Net [15]. RandLA-Net represents here a fundamental framework for large-scale point clouds and it has been chosen as a model for this thesis. A more detailed explanation of RandLA-Net architecture is given in Section 2.4.

Point Convolution methods. These methods propose effective convolution operators specifically for point clouds. The pioneer work is here PointCNN [5] that uses a χ-transformation instead of symmetric functions to canonicalize the order of points; this way it performs a generalized convolution operation to learn features from unorderd and unstructured point clouds.

RNN-based methods. RNN-based methods have been introduced to model the interdependency between point clouds. One of the major work here is PointRNN [9] which proposes the network in two different versions depending on the recurrent module introduced, i.e., PointGRU and PointLSTM.

[3]For example important models are AlexNet [18], VGG [38], GoogLeNet [40], ResNet [14].

Graph-based methods. The last methods use networks based on graphs, that are really suitable data structures to represent point clouds as they can capture the geometric structure and shape of objects. When representing point clouds with graphs, each node corresponds to an input point and each edge represents the relationship between the point and its neighbors. A milestone in graph-based methods is DGCNN [43] that constructs a local neighborhood graph to extract the local geometric features and applies convolution-like operations to capture local geometric structures.

2.4 RandLA-Net

One of the main point based architectures for LiDAR semantic segmentation is RandLA-Net [15] proposed by Hu *et al.* in 2019. RandLA-Net is one of the most lightweight solutions available which is able to outperform many of other state-of-art models. For this reason, it has been chosen as base model for this work.

RandLA-Net tackles the problem PointNet had in capturing context information of each point, proposing an effective local feature aggregation module. This module is able to automatically preserve complex local structures by progressively increasing the receptive field for each point. More details on the architecture and this module are given in following sections.

In general, RandLA-Net fundamental features are:

1. It only relies on random sampling (RS) within the network, thereby requiring much less memory and computation.
2. The proposed local feature aggregator (LFA) can obtain successively larger receptive fields by explicitly considering the local spatial relationship and point features, thus being more effective and robust for learning complex local patterns.
3. The entire network only consists of shared multi-layer perceptrons without relying on any expensive operations such as graph construction and kernelisation, making it ideal for large-scale point clouds.

2.4.0.1 Architecture

The architecture of RandLA-Net is shown in Figure 2.8. It is structured as a classic encoder-decoder architecture for semantic segmentation with skip connections (Section 2.1.3). The input point cloud is first fed to a shared MLP layer to extract per-point features. Four encoding and decoding layers are then used to learn features for each point. At last, three fully-connected layers and a dropout layer are used to predict the semantic label of each point (Figures 2.8 and 2.10). The details are providided below.

Network Input: The input is a large-scale point cloud with a size of $b \times N \times d_{in}$, where b is the batch size, N is the number of points in the input point cloud and d_{in} is the feature dimension of each input point. For SemanticKITTI dataset (Section 3.2.3) $d_{in} = 3$ since

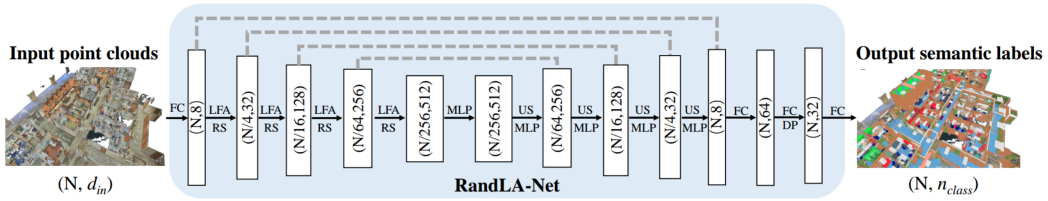


Figure 2.8: RandLA-Net original architecture [15].

each point is represented by its 3D spatial coordinates (x, y, z).

Encoding Layers: Four encoding layers, composed of a local feature aggregation module and a random sampling operation, are stacked. These layers progressively reduce the size of the point clouds and increase the number of features associated to each point in order to extract even more information.

Decoding Layers: On the other side, four decoding layers use the k-nearest neighbours (KNN) algorithm to find one nearest neighboring point for each query point and upsample the point feature set through a nearest-neighbor interpolation. Next, the upsampled feature maps are concatenated with the intermediate feature maps produced by encoding layers through skip connections, after which a shared MLP is applied to the concatenated feature maps. More details will be given in sections 2.4.0.2.

Final Semantic Prediction: The final semantic label of each point is obtained through three shared fully-connected layers and a dropout layer with ratio $p = 0.5$.

Network Output: The output of RandLA-Net is the predicted semantics of all points, with a size of $b \times N \times n_{class}$, where n_{class} is the number of different classes that can be predicted. Figure 2.9 shows the detail of RandLA-Net architecture.

2.4.0.2 Local Feature Aggregation module

The main characteristic of RandLA-Net is its local feature aggregation module, shown in figure 2.10. It is applied to each 3D point in parallel and it consists of three neural units: local spatial encoding (LocSE), attentive pooling, and dilated residual block.

Local spatial encoding: In this module all the points of a given point cloud P are processed. In particular, k-nearest neighbours algorithm is used to find per each point p_i its neighboring k points according to point-wise euclidean distance measure. Then, the position of each of the k neighboring points is encoded as follows:

$$\boldsymbol{r}_i^k = MLP\Big(p_i \oplus p_i^k \oplus (p_i - p_i^k) \oplus \|p_i - p_i^k\|\Big) \tag{2.11}$$

where p_i and p_i^K are the x-y-z positions of points, $\oplus$ is the concatenation operation, and $\|\cdot\|$ calculates the Euclidean distance between the neighbouring and center points. Finally, for each neighboring point p_i^k, the encoded relative point position $\boldsymbol{r}_i^k$ is concatenated with its corresponding point features $\boldsymbol{f}_i^k$ obtaining an augmented feature vector $\hat{\boldsymbol{f}}_i^k$. This passage is not performed with SemanticKITTI (Section 3.2.3) as its point clouds are provided only with x-y-z coordinates.

Attentive pooling: This neural unit is used to aggregate the set of neighbouring point features $\hat{\boldsymbol{F}}_i$. Instead of the widely used max or mean pooling, an attention mechanism is used to automatically learn important local features avoiding information loss.
Specifically, a shared function $g(\cdot)$, consisting in a shared MLP followed by softmax, learns a unique attention score for each feature, that is used to weight the feature vector $\hat{\boldsymbol{f}}_i^k$ and learn the important features. The attention score and the final feature vector can be expressed as follows:

$$\boldsymbol{s}_i^k = g(\hat{\boldsymbol{f}}_i^k, \boldsymbol{W}), \qquad \tilde{\boldsymbol{f}}_i = \sum_{k=1}^{K} (\hat{\boldsymbol{f}}_i^k \cdot \boldsymbol{s}_i^k) \tag{2.12}$$

where $\boldsymbol{W}$ is is the learnable weights of a shared MLP.
Summarizing, given the input point cloud P, for the i-th point p_i, LocSE and Attentive Pooling units learn to aggregate the geometric patterns and features of its K nearest points,

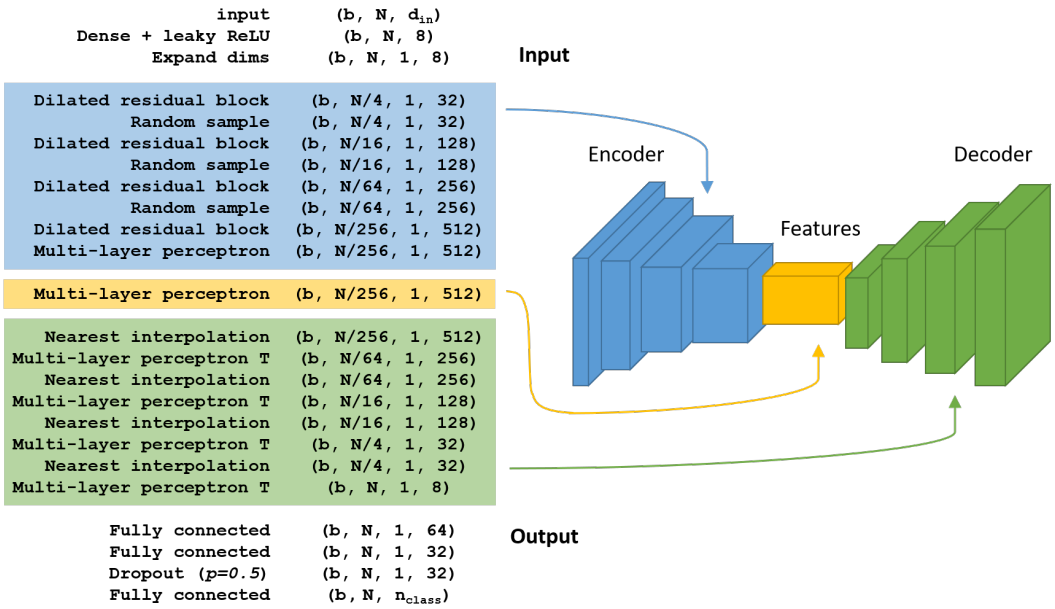


Figure 2.9: RandLA-Net detailed architecture structure.

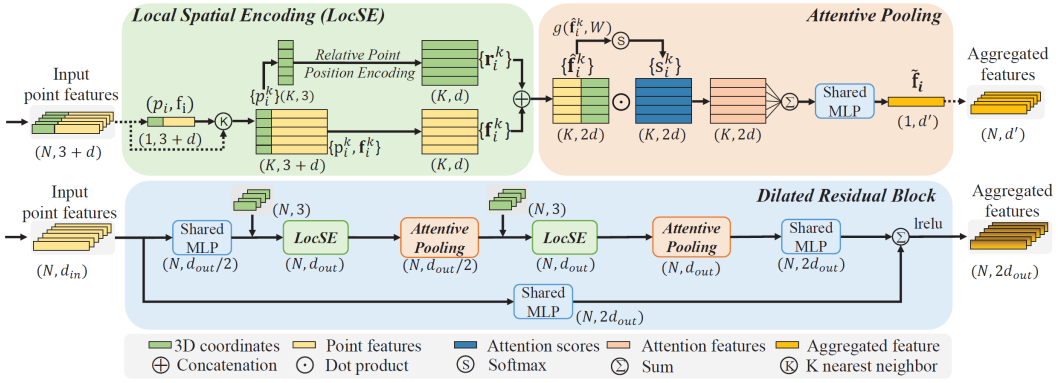


Figure 2.10: RandLA-Net [15] feature aggregation module.

and finally generate an informative feature vector $\tilde{f}_i$.

Dilated residual block: Since RS continously down samples the input point cloud, the receptive field of each point needs to be increased significantly at each step to preserve the geometric details of each input point cloud, even if some points are dropped. Connecting multiple LocSE, Attentive Pooling and skip connections, a Dilated Residual Block is built (Figure 2.10).
In RandLA-Net two sets of LocSE and Attentive Pooling are stacked as the standard residual block, achieving a satisfactory balance between efficiency and effectiveness. The overall module effectively preserves complex local structures considering neighbouring geometries and significantly increasing receptive fields, while being computationally efficient.

3 Three-dimensional data

In this chapter a general overview of LiDAR point clouds is provided. Three-dimensional data structures are described and the SemanticKITTI dataset is presented.

* * *

Powerful computers and image acquisition devices have made it possible to capture reality through serial sets of data and store these digitally into three-dimensional objects. These three-dimensional data sets can be processed and rendered much like two dimensional data sets. However, adding a third dimension augments the amount of information to be processed, which is consequently greater than for two dimensional objects.

In the previous chapter we have analyzed the task of semantic segmentation in computing images. But what happens when the data are not regularly distributed like pixels? What are the differences whilst computing those data instead? What are the problems that arise? As we have seen, 3D semantic segmentation methods employ different data structure to deal with three-dimensional data, using even diverse architectures.
The following sections will provide a general overview of these three-dimensional data structures: the LiDAR data, their properties and the problematics related are then specifically analyzed.

3.1 3D DATA STRUCTURES

A three-dimensional scene can be modeled in different ways. In general, the three-dimensional information acquired through sensors is represented in different forms that vary in both their structure and properties.
The most direct way to represent them is through point clouds, as they are the form in which raw sensor data are naturally stored. However there are many other representations that can be used, which can be more or less suitable for a specific task.

3.1.1 Volumetric Models

Volumetric models are regular data structures that represent the three-dimensional volume of an object or 3D structure. They are the most common and intuitive representation of 3D data as they are just an extension of bidimensional images to the third dimension.

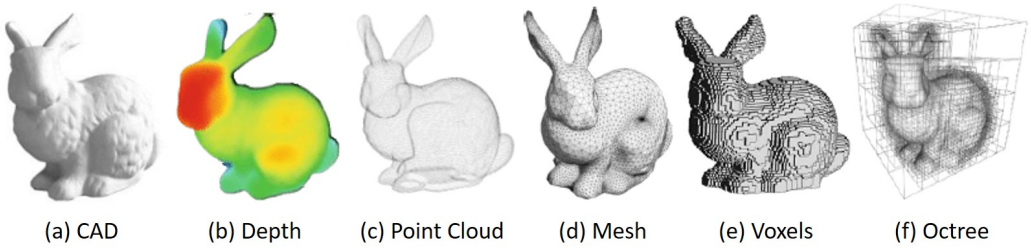


Figure 3.1: The Stanford Bunny [41] model in six different three-dimensional representations.

Volumetric representations are practical for rendering and smooth visualization. However they come to approximating the initial geometry and introduce aliasing artifacts. In addition, their highly structured grid layout makes them suitable for processing using 3D convolutional neural networks.

Voxels (Figure 3.1e) are the equivalent three-dimensional representation of an image pixel. A voxel is formally a three-dimensional cubic unit block that represents a naive extension of occupancy grids[1] to the three-dimensional space. A sparse voxellized representation can be obtained directly from point clouds, by discretizing the space and filling voxels where one or more points are present. Unfortunately this representation is rough and bulky, hence octrees are often preferred.

Octree (Figure 3.1f) is a tree-based data structure which provides a finer representation of the 3D space by recursively subdivide voxels into octants while keeping only the occupied cells.

3.1.2 Shell or boundary models

Shell or boundary (B-Reps) models represent the boundary of the object, not its volume. Almost all visual models used in reality capture workflows, games and film are boundary representations.

The main B-Rep model is **3D mesh** (Figure 3.1d). A 3D mesh is a geometric data structure, commonly used in computer graphics, which encodes a 3D object geometry in terms of a combination of edges, vertices, and faces. A mesh represents the surface of a 3D object using polygon (e.g., triangles or quadrilaterals) shaped faces is termed as the *polygon mesh*. A *regular mesh* is composed of only a single type of polygons, like the most common *triangular mesh*.

[1] An occupancy grid is a bidimensional space subdivided through a grid system representing an environment. In this grid system the occupied cells are filled while the free space is not.

By extending this representation it is possible to obtain different B-Reps, which are always composed of vertices, edges and faces but with different shapes but representing both the interior volume along with the object surface (volumetric mesh).

Meshes are a great way to explicit the geometry of a point cloud, and they frequently allow for a significant reduction in the number of points required as vertices. On top, they permit to get a sense of the relationship between objects through the faces' connectivity. However, meshing is an interpolation of the base point cloud geometry, and can only represent the data to a certain degree, which is determined by the mesh's complexity. There are a variety of ways for meshing a point cloud, but the best results typically necessitate some a priori knowledge of the object's shape.

3.1.3 Parametric models

Parametric models use constructive solid geometry (CSG) technique to define complex objects as a composition of feature-based simple objects such as cubes, spheres, cones, and cylinders with a set of operations such as union, intersection, addition, and subtraction. This representation is often used for CAD models (Figure 3.1a) in computer vision and graphics. Building parametric models form point clouds is however a hard task since it demands a very smart structuration of the object.

3.1.4 Depth maps

A depth map (Figure 3.1b) is an image that encodes depth information about related to the 3D scene from a single viewpoint. This information is encoded using height above the ground, horizontal disparity and angle with gravity for each pixel. This representation is fine if surface information linked to a known point of view is enough. For example, in real-time autonomous driving scenarios, depth maps allow to map the environment at 360° very quickly. Depth maps are good data structure for low memory requirements, but they suffer for week topology and cannot describe a full 3D scene on their own.

3.1.5 Point Clouds

A point cloud (Figure 3.1c) is a set of data points in the 3D space. Each point is spatially defined by a triplet of coordinates (x, y, z) and a combination of such points can be used to describe the geometry of an object or the complete scene.
Point clouds are easily obtained as a map of the external surface of objects through reality capture devices, like LiDAR sensors. However, they can also be the result of Photogrammetric Scanning, depth sensing, and more recently deep learning through Generative Adversarial Networks (GANs).

Point clouds provide simple yet efficient and precise 3D data representation and can be subjected to operations like fast linear transformations, objects combinations and fast rendering, i.e., projects and draws the points onto an image plane.
These benefits, on the other hand, must contend with memory constraints and free space

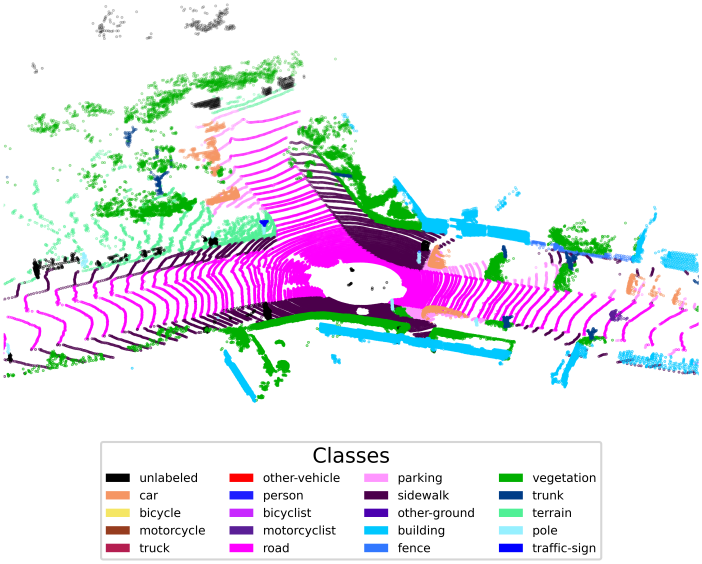


Figure 3.2: Example point cloud from the SemanticKITTI dataset.

representation issues: detecting the same position several times results in multiple overlapping dense points, while non-measured space is treated similarly as free space. In addition, even if fast rendering is appliable directly on point clouds, in the most of the cases it is preferred to utilize a solid representation of the three-dimensional geometry, deriving a mesh with a suitable surface reconstruction technique.

3.2 LiDAR Point Clouds

LiDAR point clouds are the most commonly used data in point cloud semantic segmentation. With the increasing application of this method in multiple fields, such as autonomous driving, remote sensing, photogrammetry, there is a rise of large-scale datasets with more than one million of points. Nevertheless, developing fine-annotated 3D LiDAR datasets is extremely labor intensive and requires professional skills.

3.2.1 Datasets

Data and algorithm selections are strongly driven by the requirements of specific applications. There exist many kinds of point cloud data, depending on the technologies used for the acquisition.

In general, LiDAR datasets (Table 3.1) can be divided into three groups:

1. **Static datasets:** data collected from static viewpoints by terrestrial laser scanners or using MLS (Mobile Laser Scanning) systems that capture mainly static scene objects for applications such as street view, 3D modeling, and virtual realities.

2. **Sequential datasets:** data collected as sequences of frames from vehicular platforms for ADAS (Advanced Driving Assistance System) or autonomous driving applications, which can be further divided into datasets with pointwise or 3D bounding box annotations. These datasets usually contain more frames and moving objects with respect to static datasets, but the sparsity in their points is higher especially on the vertical direction.
3. **Synthetic datasets:** data collected in a virtual world by simulating any of the above data acquisition systems. The problem of using such datasets is caused by the large gap between synthetic and real scenes. Synthetic scenes can generally be very realistic, but they lack accuracy in detail.

Image and RGB-D datasets (Section 2.2.1.3) have much larger scales and their number of pixels/frames are more sufficient than 3D LiDAR ones. Although the studies on image and RGB-D still face the data hunger problem (Section 3.2.2), it is more serious in the domain of 3D LiDAR datasets.

Among the most popular dataset we can identify *Semantic 3D* [13] as the existing largest LiDAR dataset for outdoor scene segmentation tasks. It is a static dataset composed of more than 4 billion points, split into training and test sets with nearly equal size, and labeled with 8 classes. *Paris-Lille-3D* [35] is another popular MLS point cloud dataset with more than 140 million labelled points, acquired with a mobile LiDAR in two French cities, Paris and Lille. Despite it is a static dataset, it suits autonomous vehicles' applications.

SemanticKITTI [2] is a dataset built on KITTI dataset [12], acquired from an autonomous driving platform and records six hours driving using digital cameras, LiDAR, GPS/IMU inertial navigation system on European streets. The large data size makes SemanticKITTI very helpful for training deep learning models. Indeed, it is used as a benchmark in RandLA-Net (Section 2.4) and also in this work. A detailed description is provided in Section 3.2.3.

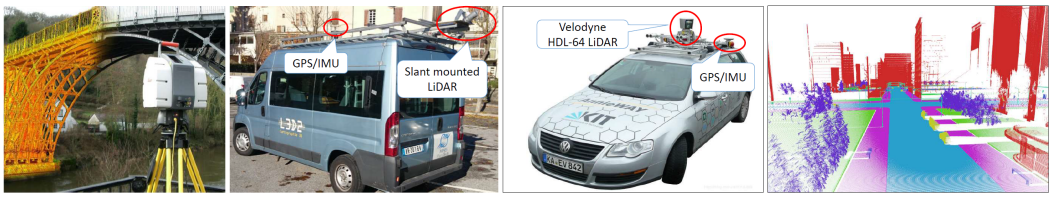


Figure 3.3: Acquisition systems for (a, b) static datasets, (c) dynamic datasets, (d) synthetic datasets [10].

	Dataset	Year	Frames	Points	Scene	Instance	Sequential
Static Data	Oakland	2009	17	1.6M	outdoor	✗	✗
	Paris-rue-Madame	2014	2	20M	outdoor	✓	✗
	TerraMobilita/Iqmulus	2015	10	12M	outdoor	✓	✗
	S3DIS	2016	5	215M	indoor	✗	✗
	Semantic3D	2017	30	4009M	outdoor	✗	✗
	Paris-lille-3D	2018	3	143M	outdoor	✓	✗
Synthetic Data	GTA-V	2018	/	/	outdoor	✗	✓
	SythCity	2019	75000	367.9M	outdoor	✗	✓
Sequential Data	Sydney Urban	2013	631	/	outdoor	✓	✓
	SemanticKITTI	2019	43552	4549M	outdoor	✓	✓
	SemanticPOSS	2020	2988	216M	outdoor	✓	✓

Table 3.1: The main datasets for LiDAR semantic segmentation and their features [10].

3.2.2 Data hunger

The performance limitation caused by insufficient training data is called the *data hunger effect*, which is reflected on both data size and diversity.
The LiDAR semantic segmentation datasets (Section 3.2.1) are annotated entirely or partially by humans, which takes time, is labor intensive, necessitates specialized knowledge and software and is more difficult than annotating images since 3D data are more difficult to be interpreted. Because of these challenges, publicly accessible datasets for LiDAR semantic segmentation are far smaller and less diverse than those for 2D images. As a result, 3D semantic segmentation may face a serious data hunger problem.

Many datasets for LiDAR semantic segmentation have been developed, and used as benchmark for deep learning models comparison, so far. But each of these dataset has its own size and composition of scenes.
On this concern, Gao *et al.* in 2020 [10] led experiments on such different datasets and deep learning models for semantic segmentation, pointing out that the problem is strictly connected to two factors: scene diversity and dataset size.

The *scenes diversity* represents a problem for deep models, since the accuracy decreases whenever the training is made on a dataset and the testing on a different one. In addition, a better understanding happens if the scenes content is heterogeneous, but mixing datasets negatively impacts the model learning due to the domain gap between different data. Furthermore, some other factors also affect model performance, such as the sensor difference, especially in image-based approaches. The data hunger problem in scene diversity is still challenging for 3D LiDAR semantic segmentation models to improve their generalization ability.

On the other hand, also the *dataset size* highly impacts the performances of deep learning models. In general, a large-scale dataset is more effective on whatever model, but the

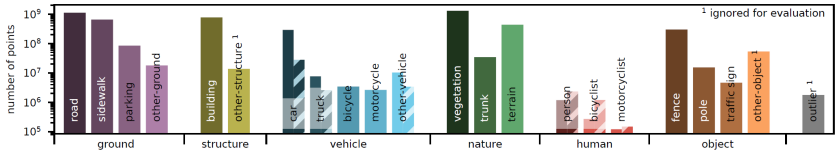


Figure 3.4: Classes of SemanticKITTI [2].

impact the size has highly depends on the model construction. For example, Gao *et al.* [10] shown that in PointNet [30] and derived models (Section 2.3.3) the performance, compared with the amount of data required saturates with 25% training data. The most of deep learning models continuously improve their performances if fed with more data, and would require more than existent datasets.

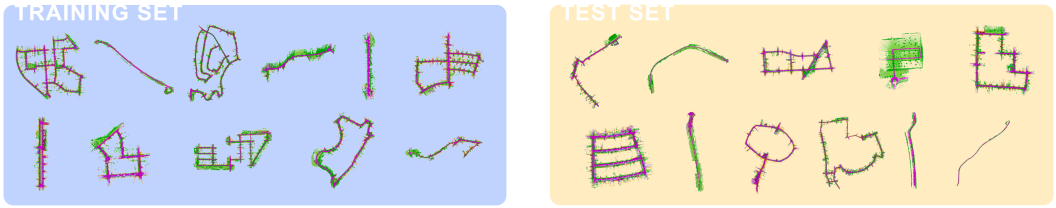
Finally, even the imbalance of classes can represent a problem (as pointed out in Section 2.2.3) for LiDAR semantic segmentation, as well as the distance from the sensor and the number of involved points. In chapter 4 many analyses to further investigate those problems are led on the selected dataset, to have more knowledge on the techniques to be adopted in order to face them.

3.2.3 SemanticKITTI

SemanticKITTI is the dataset chosen for this work. It is actually one of the largest and most popular datasets suitable for various tasks, including LiDAR semantic segmentation. It was developed by Behley *et al.* in 2019 [2] and contains detailed point-wise annotations with 28 classes, on 22 different scenes.

The dataset was obtained by annotating the sequences of the KITTI dataset, published by Geiger *et al.* in 2012 [12], acquired with a velodyne system (Figure 3.3c), composed of a LiDAR sensor, a stereo camera rig (RGB + depth), a global positioning system (GPS), and Inertial Measurement Unit (IMU). The KITTI dataset provided many benchmarks, but the only part exploited for SemanticKITTI was the odometry from the SLAM system, on which the labels were placed.
The LiDAR acquisitions, taken in the German city of Karlsruhe in rural areas and on highways, are highly heterogeneous as they present diverse landscapes with variable sequence lengths and number of points per acquisition.

All the data points have been labeled according to the object they belong to, providing a ground truth for the supervised task of point cloud semantic segmentation. However, the semantic labels have been provided only for the training and validation sequences, while the test ones are left as a challenge. In the original setup, the sequences chosen for training are `00,01,02,03,04,05,06,07,09,10`, sequence `08` was chosen for validation, while sequences from `11` to `21` for test.



(a) Training sequences.

(b) Test sequences.

Figure 3.5: Sequences of SemanticKITTI [2].

To make comparison with other benchmark datasets the classes have been reduced to 19. Figure 3.4 shows the overall frequency of each class in the dataset. SemanticKITTI has been chosen as a benchmark for this work for the diversity of its scenes and large-scale of dataset.

4 Dataset Analyses

In this chapter some analyses led on the SemanticKITTI dataset are presented. General features of the dataset and features related to the performance of RandLA-Net on it are investigated.

* * *

As pointed out in the previous chapter, when dealing with LiDAR point clouds some problematics linked to the data properties arise. SemanticKITTI dataset (SemanticKITTI [2], presented in Section 3.2.3) has been chosen since it provides a wide and heterogeneous dataset, suitable for semantic segmentation. In this chapter we are going to analyze its properties in order to have a broader overview of the problem to deal with and a solid understanding of the material available.

Section 4.1 presents some general analyses of the raw LiDAR data, while Section 4.2 reports other analyses, specifically related to the output of the chosen architecture (RandLA-Net [15], investigated previously in Section 2.4).
Notice that the whole study is completely *independent* with respect to both the dataset and architecture: even if SemanticKITTI and RandLA-Net are chosen it can be generally adapted to whatever LiDAR dataset or model.

Also, given that test sequences (from `11` to `21`) are provided with no ground truth labels, we consider in our setup only sequences from `00` to `10`. Hereinafter sequences `00,01,02,03,04,05,06,07,09,10` will be considered as the training data, while sequence `08` as the test one.

4.1 General analyses

General analyses are led to understand how the point clouds involved are structured and distributed in the space. In particular, the properties analyzed are the following:

- The distribution of classes within a certain distance from the LiDAR sensor.
- The distribution of classes at a certain height.
- The distribution of number of points assigned to each class in each sequence.

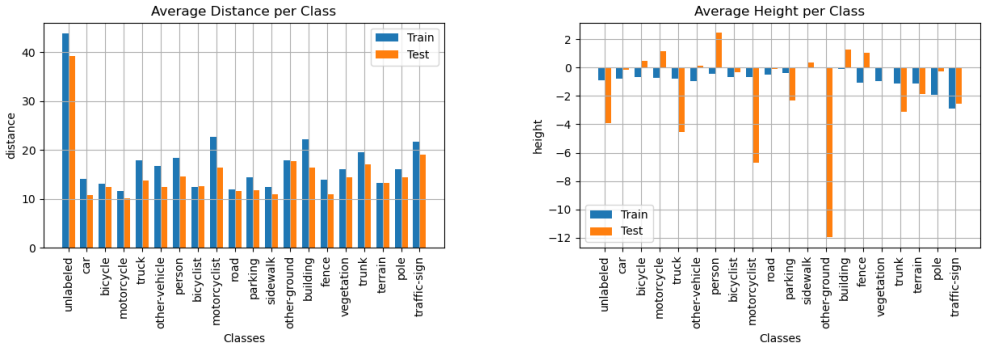
4.1.1 Distances

The distribution of the distance of the classes with respect to the LiDAR sensor position is shown in Figure A.1. It follows a *power law* distribution as the number of points expo-

nentially decrease with increasing values of the distance. This means that the number of points is greater for closer objects with respect to farther objects, which is reasonable as the point clouds are acquired with a LiDAR sensor.

The figure shows also the *unlabeled* class since this analysis is made a priori considering the ground truth labels. It is interesting to notice that this distribution has a curious behaviour as it starts again to rise when a threshold of 50m is overcome. This is due to the fact that after a certain threshold (50m in this case) all the points are classified as *unlabeled* because they are considered too far from the LiDAR sensor to be classified correctly with a sufficient level of confidence.

The average distance per class in the training and test sequences is shown in Figure 4.1a. Training and test sequences do not differ much in general, even if the training samples are on average a little more distant.



(a) Distances.

(b) Heights.

Figure 4.1: Analyses of the distances and heights relative to the sequences in the training set and in the test set. The measures are expressed in meters (m).

4.1.2 Heights

The heights plot of Figure A.2 shows the distribution of the points per class at a certain height from the LiDAR sensor position (0m). As expected the height is greater for objects that are actually higher and lower for shorter ones. Notice that the height is computed with respect to the LiDAR and not the ground, hence also negative heights are resonable.

Figure 4.1b shows the average heights of the points per class in the training and test sequences. In general the result confirms that the vertical resolution is limited, which is one of the problematics related to sequential datasets like SemanticKITTI. The greater variability in the test heights is due to the fact that a single sequence is considered, while for training the result is averaged on 10 sequences.

4.1.3 Number of Points

Table 4.1 presents the total number of points assigned to each class in each sequence, which is represented also in the histograms of Figures 4.3 and 4.2.

The distribution of classes per sequence is pretty uniform. However there is a great imbalance in the overall number of points assigned to some classes with respect to others. For example classes like *building*, *vegetation* and *road* hold a huge number of points (order of 10^8), while classes like *motorcyclist* and *other-vehicle* hold very little points (order of 10^5).

In addition, there are some "holes", i.e., some sequences lack in some classes (e.g., sequence 01 lacks of *bicycle*, *motorcycle*, *truck*, *person*, *bicyclist* and *parking*). These missing class are typically the least frequent ones and many sequences lack in those classes (e.g., *bicyclist* is absent also in sequences 03 and 04).

class	#	00	01	02	03	04	05	06	07	08	09	10	tot
unlabeled	0	1.08E+07	4.13E+06	9.55E+06	2.96E+06	9.37E+05	7.98E+06	8.39E+06	3.05E+06	1.63E+07	5.24E+06	3.32E+06	7.27E+07
car	1	2.48E+07	5.33E+05	7.23E+06	8.92E+05	3.16E+05	5.98E+06	3.53E+06	6.55E+06	1.59E+07	4.01E+06	1.60E+06	7.14E+07
bycicle	2	1.75E+05	0.00E+00	1.21E+04	1.17E+04	0.00E+00	1.18E+04	6.18E+04	4.23E+04	1.98E+05	3.95E+03	3.50E+03	5.20E+05
motorcycle	3	3.15E+05	0.00E+00	3.55E+04	0.00E+00	0.00E+00	9.62E+03	3.28E+04	8.43E+04	1.95E+05	4.62E+04	1.90E+04	7.37E+05
truck	4	2.15E+05	0.00E+00	0.00E+00	4.19E+02	0.00E+00	5.34E+05	7.69E+05	6.80E+05	2.69E+05	3.07E+05	7.39E+04	2.85E+06
other-vehicle	5	7.18E+05	6.31E+04	1.54E+05	9.12E+04	6.30E+04	5.26E+05	1.00E+06	2.32E+05	1.11E+06	4.04E+05	2.25E+04	4.38E+06
person	6	1.06E+05	0.00E+00	4.62E+04	2.97E+03	1.96E+03	7.92E+04	2.53E+04	1.23E+05	3.16E+05	8.27E+04	8.53E+04	8.68E+05
bicyclist	7	9.69E+04	0.00E+00	4.50E+01	0.00E+00	0.00E+00	5.47E+04	1.94E+04	1.31E+04	1.84E+05	0.00E+00	0.00E+00	3.68E+05
motorcyclist	8	2.21E+03	2.76E+03	7.40E+04	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.80E+04	0.00E+00	0.00E+00	9.70E+04
road	9	4.97E+07	2.49E+07	5.89E+07	1.14E+07	6.07E+06	2.91E+07	1.53E+07	1.27E+07	4.52E+07	2.11E+07	1.23E+07	2.87E+08
parking	10	4.29E+06	0.00E+00	6.09E+06	6.94E+05	4.28E+04	2.78E+06	4.97E+05	1.07E+06	3.17E+06	1.32E+06	4.90E+05	2.04E+07
sidewalk	11	3.83E+07	3.26E+02	5.48E+07	6.18E+06	1.40E+06	2.80E+07	8.61E+06	8.60E+06	3.01E+07	1.57E+07	9.03E+06	2.01E+08
other-ground	12	2.48E+03	1.84E+06	4.30E+05	7.00E+01	2.01E+05	4.90E+05	2.90E+06	0.00E+00	3.33E+05	2.15E+05	2.84E+05	6.70E+06
building	13	8.64E+07	2.31E+05	2.71E+07	3.27E+06	3.81E+05	4.61E+07	1.30E+07	2.24E+07	4.39E+07	1.60E+07	1.56E+07	2.74E+08
fence	14	1.09E+07	1.12E+07	2.79E+07	3.36E+06	2.82E+06	2.38E+07	1.88E+06	4.09E+06	6.23E+06	6.07E+06	9.24E+06	1.08E+08
vegetation	15	9.65E+07	2.54E+07	1.49E+08	2.98E+07	9.45E+06	5.96E+07	1.14E+07	1.62E+07	1.12E+08	4.77E+07	3.07E+07	5.88E+08
trunk	16	3.14E+06	4.82E+04	3.06E+06	4.59E+05	8.89E+04	8.68E+05	4.38E+05	4.53E+05	3.89E+06	9.59E+05	3.00E+05	1.37E+07
terrain	17	1.85E+07	1.25E+07	2.25E+07	7.28E+06	2.25E+06	1.53E+07	2.50E+07	4.80E+06	4.41E+07	1.30E+07	8.04E+06	1.73E+08
pole	18	1.13E+06	2.09E+05	7.81E+05	1.88E+05	1.21E+05	4.61E+05	6.16E+05	3.48E+05	1.13E+06	3.41E+05	2.81E+05	5.60E+06
traffic-sign	19	2.08E+05	2.18E+05	1.08E+05	4.55E+04	2.17E+04	1.03E+05	2.40E+05	4.75E+04	3.07E+05	1.05E+05	7.03E+04	1.47E+06

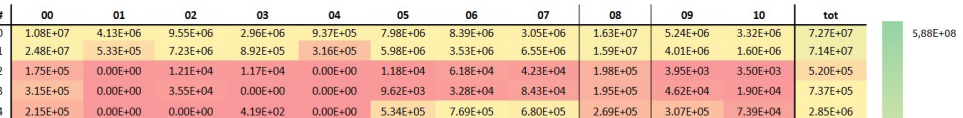


Table 4.1: Heatmap table of the number of points per each class in each sequence.

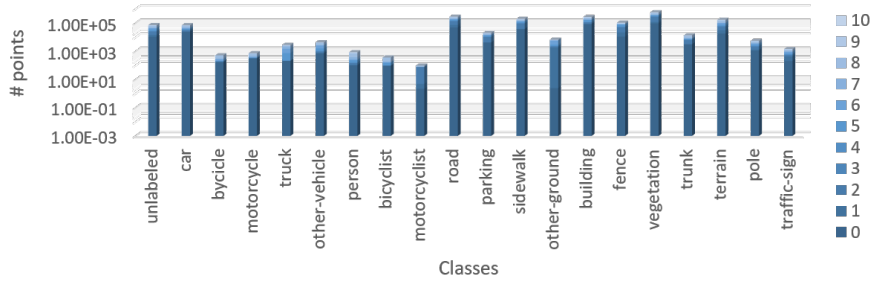


Figure 4.2: Flat plot of the overall number of points per each class in log scale.

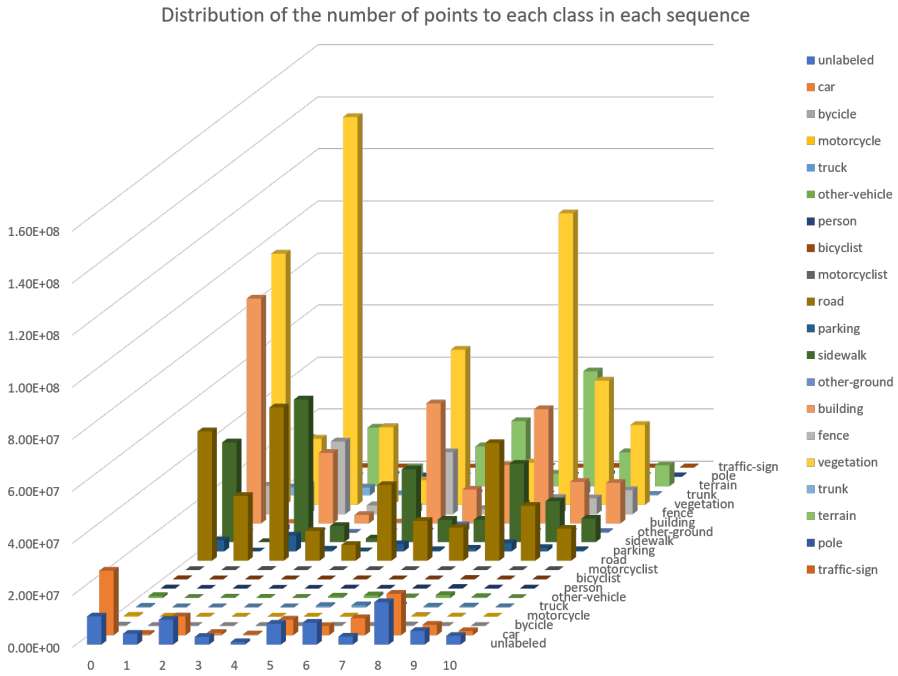


Figure 4.3: Three-dimensional plot of the number of points per each class in each sequence in linear scale.

This imbalance of points per class is important to be considered. Its influence on the outcome will be investigated later in Chapters 5 and 6 and will emerge also in the analyses of the following sections.

Overall, our general analyses confirm the results obtained by Gao *et al.* in [10] where similar analyses were led on the same dataset. In fact, according to them, the dataset has a good diversity of *vehicle* distribution. However, *persons* are scarce, while vegetation takes almost 50% of the whole points.

4.2 MODEL-BASED ANALYSES

Structural properties of the dataset have emerged form the previous section. However the correlation of the dataset properties with the model outcome can constitute a fundamental element to understand how those properties affect the results. Therefore the objective of the analyses led in this section is to relate the dataset properties with the output of the considered model.

The results obtained from the training on RandLA-Net [15] are here taken into account. The network is trained for 20 epochs with batch size $b = 4$ and the the model that gave the

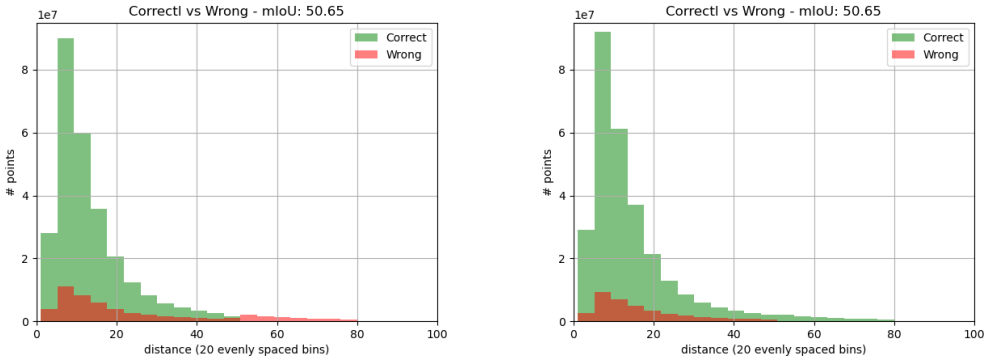
best mIoU is considered. The predictions of that model are taken in order to investigate the following aspects:

- The number of correctly segmented points and not with respect to the distance from the sensor.
- The number of points with respect to the obtained mIoU.

In addition, some Network Science techniques were exploited in order to retrieve other useful information and properly set up the methods.

4.2.1 Prediction vs distance

As shown in figure 4.4 the number of correctly classified points is greater at shorter distances and decreases if the total number of points in the class is small. Note that the class *unlabeled* is never predicted, therefore all the points are reasonably wrongly classified in one of the other 19 classes (Figure 4.4a) but can be considered as correctly classified (Figure 4.4b). For classes with a great mIoU, the distribution of the correctly classified points in less related to the distance measure.



(a) Unlabeled included.

(b) Unlabeled excluded.

Figure 4.4: Overall distribution of correctly and wrongly classified points.

4.2.2 Points vs mIoU

The plot of the mIoU per each class is shown in Figure 4.5. Comparing the diagram with the one of Figure 4.2, it is clear that in general the classes with the largest number of points are able to achieve larger values of mIoU.

There is also a weak correlation between the mIoU and the average height of the points. In fact in the test sequence, classes *motorcyclist* and *other-ground* have very negative distances and achieve the worst values of mIoU.

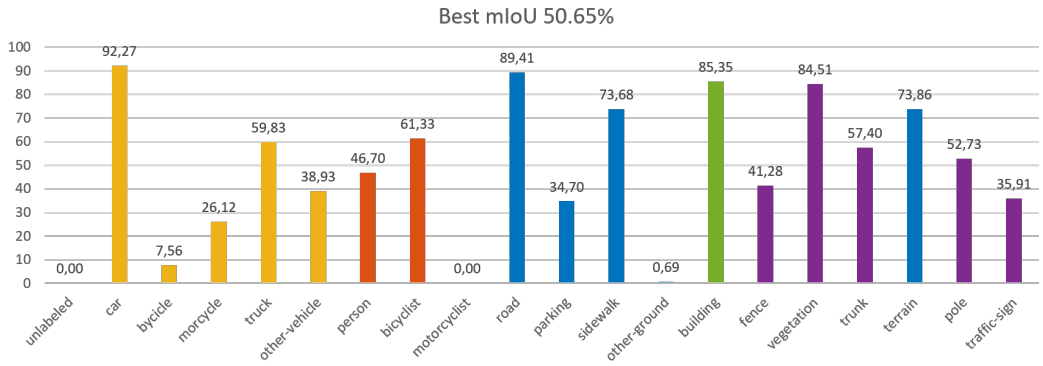


Figure 4.5: mIoU per each class resulting from the vanilla training on 15 epochs. Colors are given according to the subdivision of classes set in section 4.2.3.3.

4.2.3 Network Science analyses

Network Science techniques [1] are employed in order to understand further relationships and properties of the dataset, related to the model outcome. The confusion matrix resulting from the vanilla training (Figure 4.6) is here treated as an adjacecy matrix A, i.e., a matrix representation of a graph, where the entries of the matrix represents the nodes and the scores in the matrix represent the weights of the links between nodes.

In our case the matrix represents a direct graph because it is not symmetric. The entries on the columns are the true labels and the elements in the rows are the predicted labels. The link between a true label and a predicted label is present if a point with that true label is predicted with that predicted label.
The following sections give the details about the algorithms employed and the results obtained from each of them.

4.2.3.1 PageRank

PageRank [29] is an algorithm proposed by Page *et al.* in 1999. It was firstly employed to sort the entries produced by a web research and it was largely applied in ranking pages on Google search engine.

The task of PageRank is to provide a ranking of the entries in the network graph identifying among them the authorities and hubs: the authorities are the nodes with the greatest number of incoming links, while the hubs are the nodes with the greatest number of outgoing links. In our case, the analysis of the authorities provides a ranking among the predicted classes, while the analysis of the hubs provides a ranking among the ground truth classes.

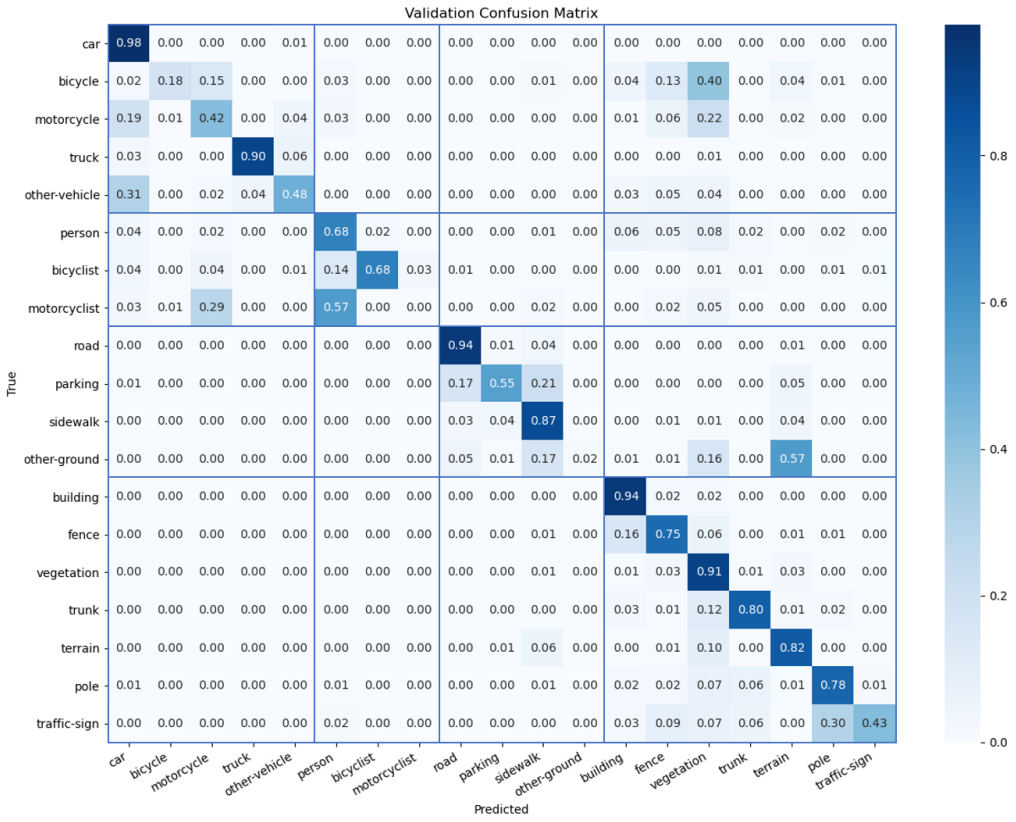


Figure 4.6: Normalized confusion matrix resulting from the vanilla training. The matrix partition is set according to the analysis of Section 4.2.3.3 exception made for classes *building* and *terrain*.

PageRank is based on the equation:

$$\boldsymbol{p_{t+1}} = cM\boldsymbol{p_t} + (1-c)\boldsymbol{q}, \qquad M = A \cdot diag^{-1}(\boldsymbol{d}) \tag{4.1}$$

where $\boldsymbol{d}$ is the degree vector, $c = 0.85$ is the dumping factor and $\boldsymbol{q}$ is the teleport vector. The authorities are given by $\boldsymbol{r} = \boldsymbol{p}_\infty$ and the solution can be found solving a linear system through power iteration. To find the hubs it is sufficient to transpose the adjacency matrix A.

The results of PageRank are given in Figure 4.7. Classes like *vegetation*, *road* and *building* appear in the highes ranked positions both in authorities and hubs scores, while *other-ground* and *bycicle* are low rated in both. Overall we can notice an accordance of both the rankings with the results found before, regarding the mIoU (Figure 4.5) and the number of points per class (Table 4.1 and Figure 4.2).

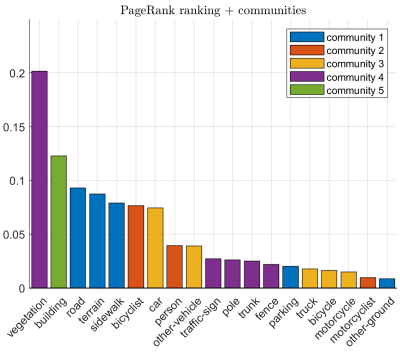
4.2.3.2 *HITS*

For the same purpose another method was employed. HITS [17] is another algorithm similar to PageRank providing authorities and hubs.
HITS is based on the equation:

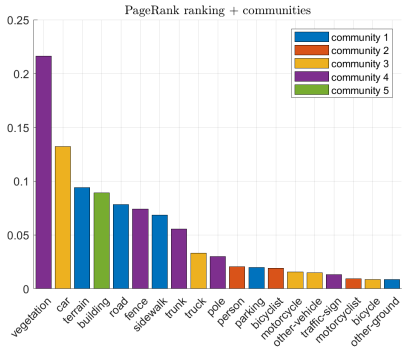
$$\boldsymbol{a_{t+1}} = M\boldsymbol{a_t}, \qquad M = AA^T \tag{4.2}$$

where $\boldsymbol{a}$ are the authority scores.

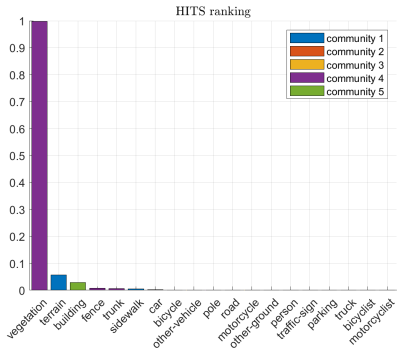
The results of the analyses with this algorithm are shown in Figure 4.7. The *vegetation* class is also here the most rated; the same holds for many other hubs and authorities found previously. Given that the hubs and authorities almost match, the considerations made for PageRank hubs and authorities ranking distribution hold also in this case.



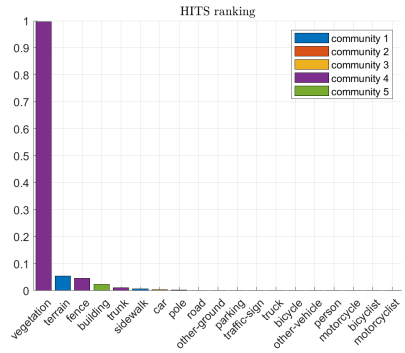
(a) Authorities of PageRank.



(b) Hubs of PageRank.



(c) Authorities of HITS.



(d) Hubs of HITS.

Figure 4.7: Hubs and authorities colored according to their community as in figure 4.8.

Figure 4.8: Resulting communities from Spectral Clustering analysis.

4.2.3.3 *Spectral Clustering*

The last technique used here is Spectral Clustering (SC) [28], an algorithm that provides a subdivision of the entries in the network, represented by the matrix. It proceeds with subsequent subdivision in two communities and iterates until the estimated number of communities is reached.

The number of communities is estimated through the following *quality measures*:

- The **Conductance**:

$$\Phi(k) = \min_{|S|=k} \phi(S), \qquad \phi(S) = \frac{cut(S, S^c)}{\min(assoc(S), assoc(S^c))} \tag{4.3}$$

  where $cut(S, S^c) = \sum_{i \in S, j \in S^c} a_{ij}$ is a partition of the vertices of the graph into the two disjoint subsets, S and S^c. The two subsets represent the two communities. Finally, V is the set of vertices of the graph and $assoc(S) = \sum_{i \in S} \sum_{j \in V} a_{ij}$ is the total number (or weight) of the edges incident with S.

- The **Modularity**:

$$Q = \frac{1}{2L} \sum_{i,j} \left(a_{i,j} - \frac{k_i \cdot k_j}{2L} \right) \cdot \eta(c_i = c_j), \qquad \eta = \begin{cases} 1 & if \quad \text{true}, \\ 0 & if \quad \text{false}. \end{cases} \tag{4.4}$$

  where $a_{i,j}$ are the elements of the confusion matrix, L is the total number of links, k_i is the degree of node i and c_i represents community i.
  The conductance plot has a number of local minima that roughly defines the number of communities. When $Q \in [0.3; 0.7]$ the number of subdivisions is optimal.

In our case, Spectral Clustering has been employed to identify the communities among the classes. The first subdivision identified the community composed by {*road*, *parking*, *sidewalk*, *other-ground*, *terrain*}, the second subdivision gave {*person*, *bicyclist*, *motorcyclist*}, the third {*car*, *bicycle*, *motorcycle*, *truck*, *other-vehicle*}, and the last one the two communities {*fence*, *vegetation*, *trunk*, *pole*, *traffic-sign*} and {*building*} (Figure 4.8).

Observe that the subdivision of the classes obtained with this method fairly matches the a priori categorization made for SemanticKITTI (Figure 3.4). In fact, the elements in each

cluster share peculiarities with the other elements belonging to the same community and result very different with respect to the elements in the other communities.

This subdivision constitutes a fundamental element that will be kept into consideration in the next chapters for choosing the best learning methods to be adopted. These learning methods are presented in Chapter 5, while the obtained results in Chapter 6. From now on, the communities will be identified as *macro-classes* and the elements inside each macro-class as *micro-classes*.

If the PageRank or HITS ranking is integrated with the SC subdivision, we can identify also a leader class in each community, i.e., the class that achieves the highest score as a hub or authority according to a ranking algorithm. As the two ranking algorithms are pretty in agreement and also do the hubs and authorities, the choice of the leader classes is straightforward. Hence, the macro-classes will be identified in the following chapter with *road*, *person*, *car*, *vegetation* and *building* classes.

5 Proposed Methods

In this chapter the proposed learning strategies are explained. All the curriculum learning and contrastive learning techniques are dealt in detail. The methodologies to obtain fair results among the macro and micro classes are introduced.

* * *

The subdivision of the classes introduced in the previous chapter has led to the choice of specifc learning approaches to improve the performances of the model.
The main idea is to rely on this *hierarchical representation* (Figure 5.1) in order to wisely organize the data and choose the learning strategies to employ accordingly. The solutions adopted are presented in the following sections.

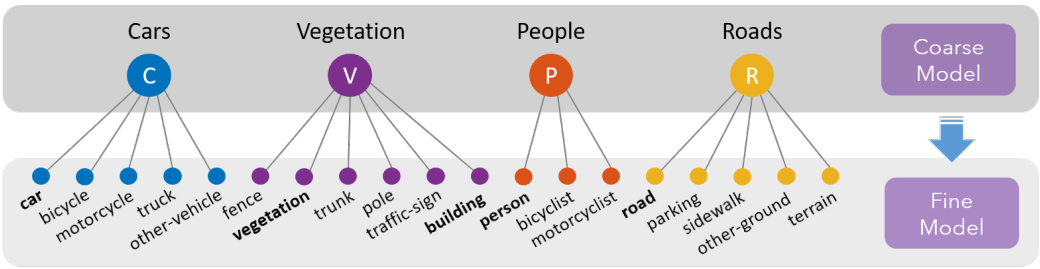


Figure 5.1: Hierarchical organization of the classes based on the analyses of Chapter 4. Note that the communities are reduced to 4 (the last 2 subdivisions are merged) to fit the batch size $b = 4$.

5.1 Curriculum Learning strategies

As stated by Y. Bengio, one of the fathers of deep learning, in 2009 in [3]:

> *Humans and animals learn much better when the examples are not randomly presented but organized in a meaningful order which illustrates gradually more concepts, and gradually more complex ones.*

In fact the idea of Bengio was to apply this graduality to deep learning as well. In this framework the model is trained first with easies examples and then the knowledge is

refined with more complex ones at advanced stages.

The definition seems pretty straightfoward and simple. However the task to determine *what is simple and what is difficult* is a hard work. Curriculum Learning was originally presented to solve a classification problem (see Section 2.1.1). In this work, it is instead applied to the complex problem of LiDAR Semantic Segmentation.
Strategies to organize the samples in a meaningful way are tried. Also, we focus the attention on particular training strategies to improve the performances with a curriculum coarse-to-fine approach [39].

5.1.1 Batch organization

Providing samples in a meaningful order can be interpreted in many ways. However, if we follow the general approach adopted in classification the meaningful order can be seen as a *witty batch organization*.

Formally, given the training set $X_t = \{x_1, x_2, x_3, \cdots, x_N\}$, the batch is a subset of the training set of size b, i.e., $B = \{x_{i_1}, x_{i_2}, \cdots, x_{i_b}\} \subset X_t$. Our model architecture (RandLA-Net, analyzed in Section 2.4) has $b = 4$ as batch size, so it is able to process $b = 4$ LiDAR point clouds at a time. This first curricular approach aims at choosing the $b = 4$ point clouds by sampling the indexes i_k, where $k = 1, \ldots, 4$, according to some criteria, explained in the following sections, and provide a gradual improvement of the network skills.

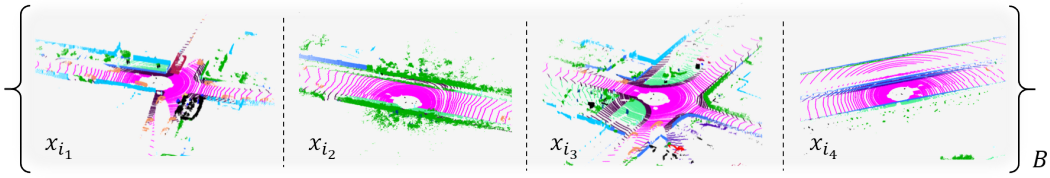


Figure 5.2: Visual example of a heterogeneous batch organization. The $b = 4$ point clouds are here very different one another. The samples presented in this figure are: `0000281x05`, `0000198x04`, `0000345x01`, `0000002x07` from left to right, where the notation `AxB` means point cloud `A` in sequence `B`. The classes colors are the standard of SemanticKITTI (see Section 3.2.3).

5.1.1.1 Random

Observing closely each point cloud x_i, it is composed of a variable number of points. Each point belongs to a different class and can be seen as part of a different object.
However the number of points per each class varies a lot from one point cloud to another (see Section 4.1, Figure 5.2). Therefore, the first idea proposed is not to place in each batch samples completely at random, but in such a way to have *heterogeneous* batches, composed in a certain percentage of classes belonging to each of the macro-classes. The tentatives proposed in this view are listed below.

- A first simple organization is based on the idea of sampling each batch in such a way that in each batch all the macro-classes are present.
- Another trial proposed is shown in Figure 5.3. Here an order based on the number of points per each class is provided in addition to the samples organization.

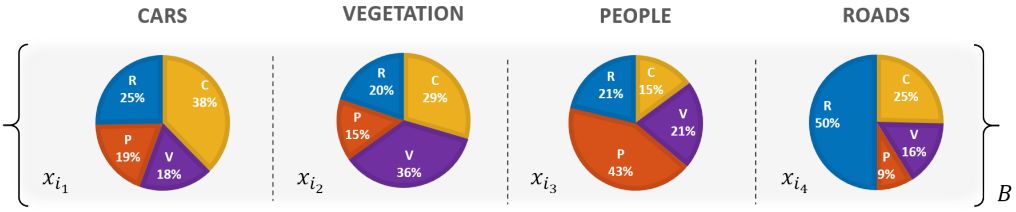


Figure 5.3: Example representation of batch where each class is present and the overall percentage of points per macro-class is fairly balanced.

5.1.1.2 *mIoU-based*

The metric chosen to order the batches is here the mIoU (Section 2.2.2). Differently from the trials based on the number of points, this metric is model-based; therefore a first pre-training of the network is required, which is performed with the vanilla setting. The different sampling methods are presented below.

- In a first trial the samples are ordered by decreasing mIoU (Figure 5.4), without keeping into consideration the batch size.
- In a second trial, they are ordered by decreasing mIoU respecting the batch composition with all the macro-classes (Figure 5.5).

5.1.1.3 *GGBB*

This last approach performs a fairness sampling both in terms of difficulty and classes involved. In fact, here the batches are composed of 2 *good* (type `G`) point clouds and 2 *bad*

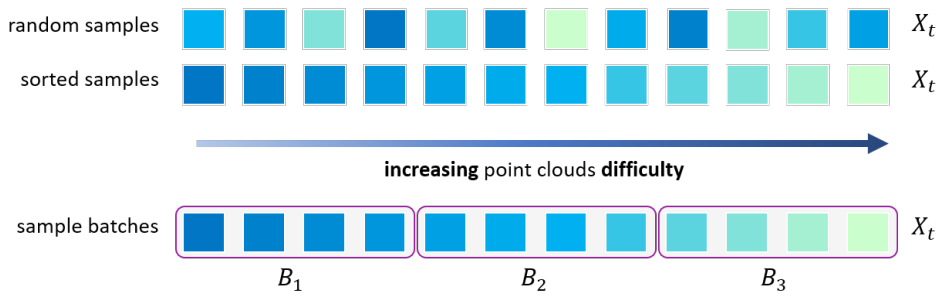


Figure 5.4: Example representation of curriculum learning batch creation. *Difficulty* is measured using mIoU or number of points.

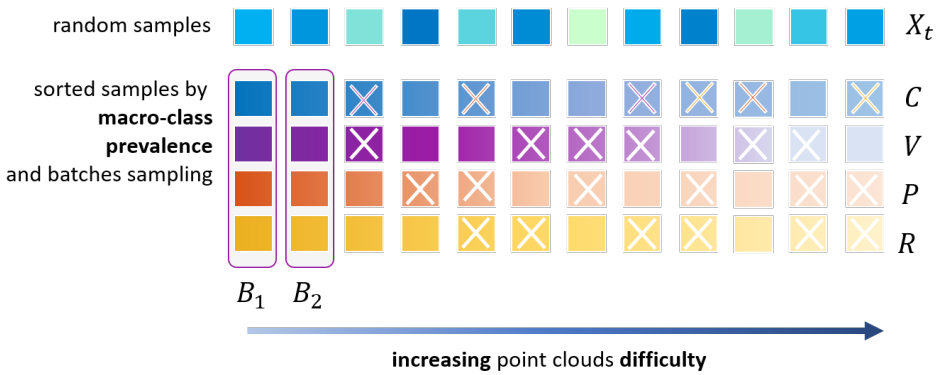


Figure 5.5: Example representation of curriculum learning batch creation respecting macro-classes sampling in the organization of batches (4 different orders within the same set). *Difficulty* measured using mIoU or number of points with respect to macro-classes. The crossed samples indicates the the sample was just selected in other categories.

(type `B`), where good and bad are determined with respect to the previously considered evaluation parameters. The trials within this configuration are described below.

- The former `GGBB` batch configuration is based on the mIoU is sampled respectively from the top and bottom of the lists, according to the ordering per macro-class with respect to the mIoU (Figure 5.5 but sampling 2 classes queue from top and 2 from bottom).
- The latter `GGBB` batch configuration is sampled similarly, but relying on the number of points per macro-class.

5.1.2 Coarse-to-Fine training

The objective of Coarse-to-Fine training strategies is similar to Curriculum Learning via batch organization, i.e., to provide a priori knowledge on the general grouping among similar classes and learn task from the easiest to the hardest. Nevertheless, the curricular approach is here dealt under a different persepective.

The hierarchy of the classes is now used in order to pretrain the model on the coarse task and fine tune it on the fine class (according to Figure 5.1). In this way a general broad separation is achieved first and the knowldege is then refined at advanced stages. The three different methods tested are explained in the following sections.

5.1.2.1 Final dense layer

One of the possible training schemes to perform a pretraining of the network on the macro-classes adds a dense layer at the output of the network. In this way the micro-classes (19 in our case) at the output are mapped to the macro-classes (4 in our case). After the network is trained for N epochs the last layer is discarded and the network is trained for $2N$ epochs on the standard architecture. Figure 5.6 illustrates the procedure.

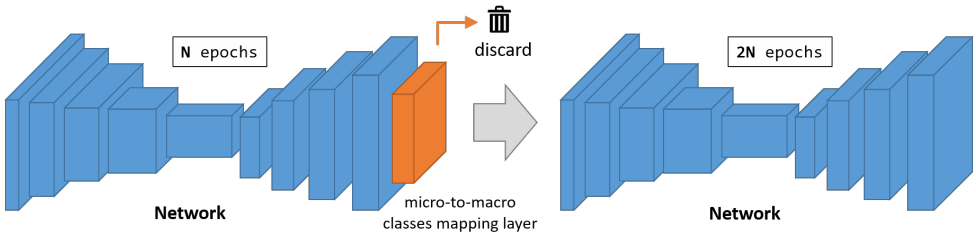


Figure 5.6: Graphical representation of the coarse-to-fine training scheme.

5.1.2.2 *Simple C2F*

Another possible training scheme follows a similar approach. Also here a pretraining of the network on the macro-classes (4 in our case) is performed. However in this training scheme the last layer is reduced to a smaller one, that classifies only the macro-classes. After the network is trained for N epochs this last layer is discarded and the backbone is preserved with all its weights.
A new layer that classifies all the micro-classes (19 in our case) is added at the output replacing the one deleted and the network is fine tuned for other $2N$ epochs. This scheme is shown in Figure 5.7 [39].

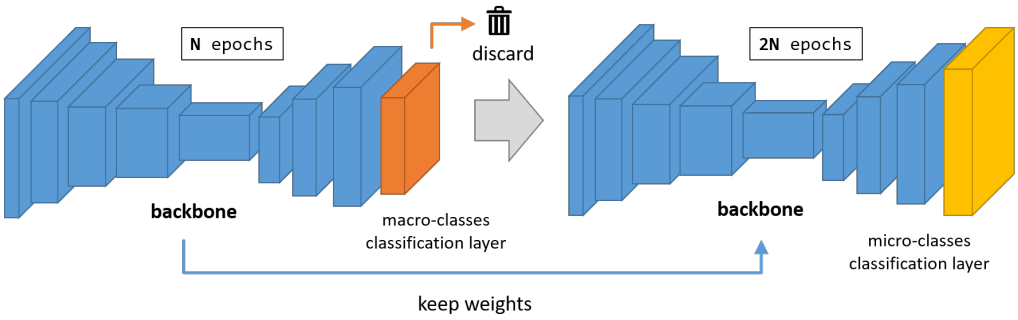


Figure 5.7: Graphical representation of the simple coarse-to-fine training scheme.

5.1.2.3 *Progressive*

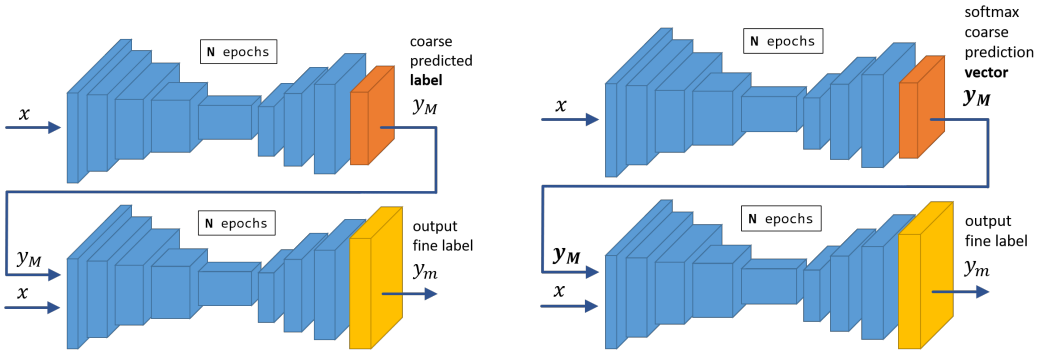
The progressive coarse-to-fine training provides a training scheme where the output of the coarse training is re-feed to the network as input later.
As the scheme in Section 5.1.2.2 a pretraining of the network on the (4 in our case) macro-classes is performed as a first stage and trained for N epochs.

The second stage instead is completely different: here the network weights are re-trained from scratch; what remains from the first stage is the coarse output prediction y_M with is fed to the network together with the input x. The second stage classifies all the (19 in our

case) micro-classes and it is trained for N epochs [32].

Three different versions of this scheme have been tested:

- A single channel progressive curriculum learning, where the *groundtruth* labels are included in the second training stage.
- A single channel progressive curriculum learning, where the *argmax* predictions are included in the second training stage, as shown in Figure 5.8a.
- A multiple channels progressive curriculum learning, where the *softmax* predictions are included in the second training stage, as shown in Figure 5.8b.



(a) Single channel.

(b) Multiple channels.

Figure 5.8: Graphical representation of the progressive coarse-to-fine training schemes. In the single channel version (a) the coarse predictions on the training set are given as an input to the network in the second stage. In the multiple channels version (b) the network is fed with softmax coarse prediction vectors (N channels) instead of coarse labels.

All the above proposed solutions clearly enhance the performances of the model adding some order and a hierarchical organization to the overall structure of classes. However in these the hierarchy and grouping of classes are not yet taken into account when the loss function is computed and the weights of the networks are updated.
In this perspective other training methodologies needs to be introduced to improve and speed up the performances.

5.2 CONTRASTIVE LEARNING METHODS

Contrastive Learning methods are further proposed to obtain a better distinction among different classes.

> *Contrastive Learning is a machine learning technique used to learn the general features of a dataset without labels by teaching the model which data points are similar or different* [4].

This technique can be applied either at the input level, the feature level or the output level. In our case, Contrastive Learning is applied at feature and output level as shown in Figure 5.9. In general the main objective of this method is to cluster similar samples belonging to the same class and put apart samples that belong to different classes. Many ways to achieve that can be employed both at a feature and output level. The following sections provide adetailed explanation of such methods.

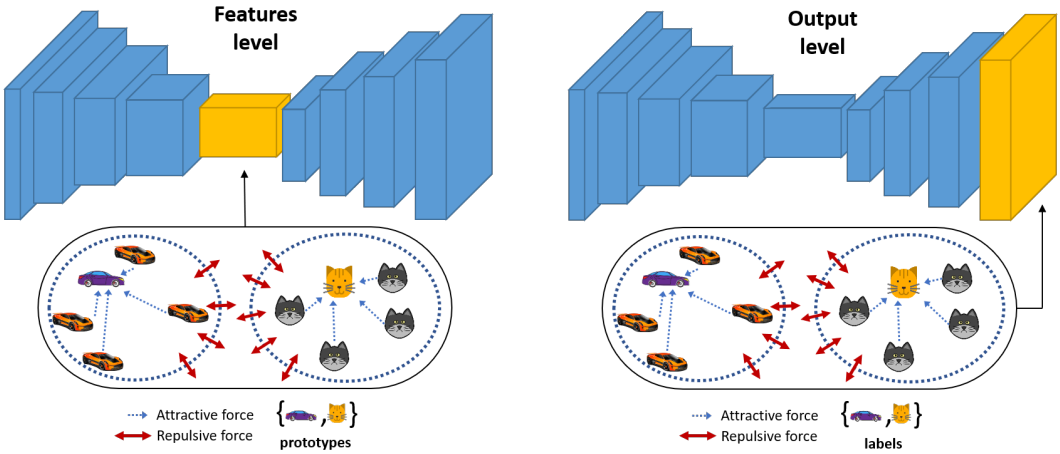


Figure 5.9: Contrastive Learning applied at feature and at output level. Loss terms are computed in order to separate different classes and cluster elements belonging to the same class [26].

5.2.1 Output level

Contrastive Learning at the output level achieves the classes separation directly on the labels; this is obtained by means of particular loss functions or additional terms addeded to the original loss. More details are given below.

5.2.1.1 Weighting schemes

The general loss function employed in the employed model RandLA-Net (Section 2.4) is a classic *categorical cross-entropy* function, i.e.:

$$\mathcal{L}_{CE} = \frac{1}{M}\sum_{k=1}^{M}\mathcal{L}_{CE_k}, \quad \mathcal{L}_{CE_k} = -\beta_k \sum_{i=1}^{N} p_i \cdot \log p_i \tag{5.1}$$

where M is the number of classes, p_i is the output of the output softmax layer and β_k is the weighting factor for class k.

One of the most direct modifications that can be performed on this loss function is to change the β_k factor in order to give different weights to the different classes. In the general

cross-entropy function $\beta_k = 1$. In the original paper of the architecture [15], $\beta_k = \frac{1}{m_k}$ where m_k is the number of points in the considered class k. In our configurations we tried to set also $\beta_k = \frac{1}{\sqrt{m_k}}$.

5.2.1.2 Dice loss

In semantic image segmentation (Section 2.1.3 one of the most diffused loss functions is the Dice loss, which is based on the Dice coefficient, a widely used metric in computer vision community to calculate the similarity between two images [16]. Dice loss can be written as:

$$\mathcal{L}_{dice} = 1 - \frac{2y\hat{p} + 1}{y + \hat{p} + 1} \tag{5.2}$$

where y is the ground truth and $\hat{p}$ the prediction. This loss function has been tested because it considers the loss information both locally and globally, which is critical for high accuracy.

5.2.1.3 Focal loss

Antoher loss function employed is focal loss [21]. This function has been tested because it down-weights the contribution of the easiest examples and enables the model to focus more on the most difficult ones. This way it works well for highly imbalanced class scenarios, like the one we are taking into consideration (see analyses of the dataset in Chapter 4). Focal loss can be expresed in analytical terms as:

$$\mathcal{L}_{focal} = -\alpha_t(1 - p_t)^\gamma \log p_t \tag{5.3}$$

where $\alpha_t \in [0, 1]$ is a weighting factor (that can be set to $\frac{1}{M}$ as in equation 5.1 and $\gamma > 0$ defines how much the harder examples are weighted more with respect to the cross-entropy.

All these loss functions are traditional losses for semantic segmentation and are employed in case of unbalanced classes. On the other hand all of them do not take into account priors on the classes but the number of points per class.
In this context, the analyses led in Chapter 4 suggest another approach that is able to exploit even the macro-micro classes subdivision and other priors obtained performing a first pretraining of the network with the standard setting.

5.2.1.4 Triplet loss

The first real tentative of applying Contrastive Learning starts from here. Triplet loss is loss function where a baseline (anchor) input x_a is compared to a positive (truthy) input x_p and a negative (falsy) input x_n. The distance from the baseline (anchor) input to the positive (truthy) input is minimized, and the distance from the baseline (anchor) input to the negative (falsy) input is maximized [37].
In mathematical terms it can be described as euclidean distance, i.e.:

$$\mathcal{L}_{triplet} = \max\{\varphi(x_a, x_p) - \lambda \cdot \varphi(x_a, x_n) + \mu, 0\} \tag{5.4}$$

where $\varphi(\cdot)$ is a loss function (we used the categorical cross entropy at the output level, while the mean squared error (MSE) at feature level). $\varphi(x_a, x_p)$ represents the attraction force, i.e., the standard loss part. $\varphi(x_a, x_n)$ instead represents the repulsion force where the samples are put far from others. λ is the coeficient that determines the measure in which this force exert w.r.t. the attractive one. It can be decomposed as $\lambda = \alpha\lambda_d$ where α is determined according to a prior, while λ_d is fixed.

The choice of the negative sample is made here exploiting the prior on the prediction the network is able to do in the vanilla configuration: each point is pushed away from the closer one that belongs to a different class (which could lead the network to a misclassification). For this purpose we referred to the confusion matrix of Figure 4.6 to sample x_n and choose α accordingly.

In general the wrongly-assigned labels assigned are mostly referred to classes within the same macro-category or classes with a huge number of points (more details on the analyses are given in Chapter 4 while a more detailed discussion on the results obtained are provided in Chapter 6). A meaningful example of this is presented in Figure 5.10.

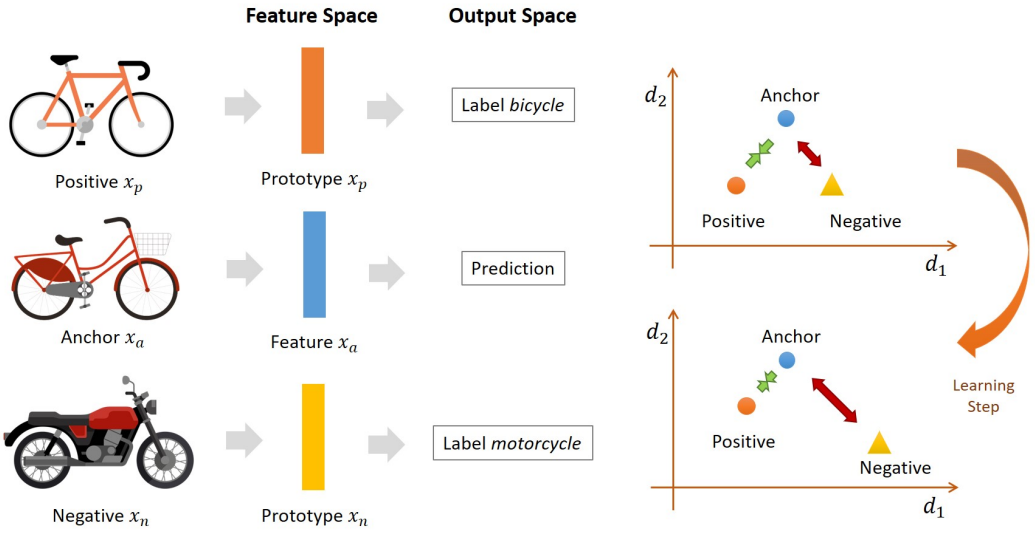


Figure 5.10: Example of triplet loss in d-dimensional space with $d = 2$. The bicycle prediction (anchor) is attracted by the bicycle sample (positive) and repulsed by the morcycle sample (negative).

The positive and negative sample are here (output level contrastive learning) the labels of those selected classes. The anchor sample is the prediction of the network. At a feature level instead the positive and negative samples are some prototype features referred to each class, while the anchor is the feature vector itself. More details will be given in Section

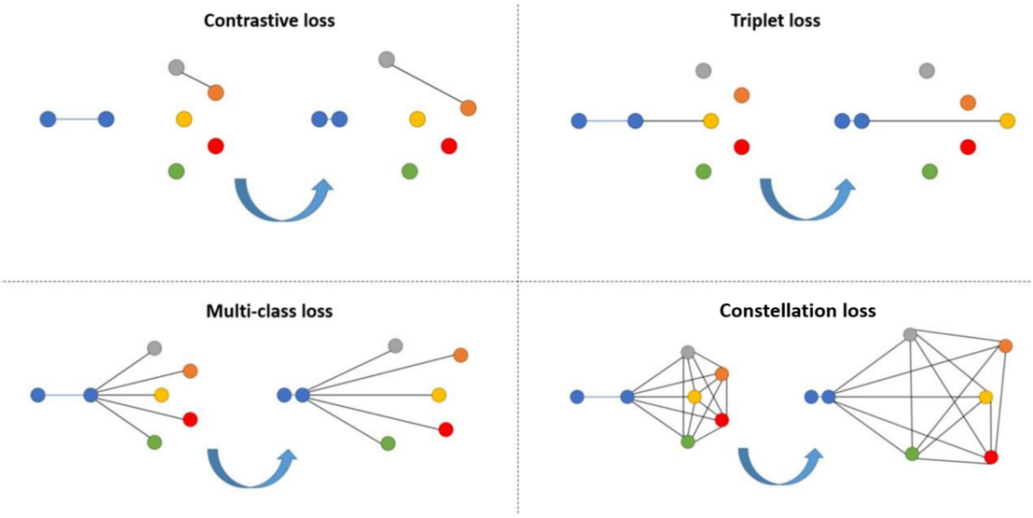


Figure 5.11: Triplet loss and other loss functions similar to triplet loss. The arrow represents the learning step. [24]

5.2.2.

There are many variants of the triplet loss, e.g., the *contrastive loss*, the *constellation loss* and the *multi-class loss* [24] (Figure 5.11). However the losses in triplet loss family are *relative* loss functions, which are not as stable as a classic categorical cross-entropy. This way other solutions have been proposed. Instead of replacing the whole loss function, additional loss terms can be added to the standard one. In this case the additional terms act as regularizers and achieve more stability. The following parargaph explains those methods.

5.2.1.5 Hierarchical loss

The hierarchical loss we introduce here is not a standard loss function as the ones described in the above paragraphs. Our proposed loss includes a loss term that accounts for priors on the classes hierarchy (Figure 5.1). The loss is composed in the following way:

$$\mathcal{L}_{hierarchy} = \mathcal{L}_{micro} + \gamma \cdot \mathcal{L}_{macro} \tag{5.5}$$

where $\mathcal{L}_{micro}$ is the loss function computed on the micro-classes, $\mathcal{L}_{macro}$ is the loss function computed on the macro-classes weighted by a factor γ [25].

5.2.2 Feature level

Contrastive learning at the features level achieves the classes separation by means of particular loss functions or additional terms addeded to the original loss. However, differently from contrastive learning at the output level, some *prototypes* of features per each class are considered, instead of labels.

The idea behind prototypes is to have a target ideal feature per each class, in order to have a measure of how much the feature from the processed sample is close to the target. This way prototypes have been built, one per each class (19 in our case).

Each prototype $\mathbf{\Gamma}_t$ is a $(1, d_{feats})$ feature vector at time t, where $d_{feats} = 512$ is the depth of the feature level of the network (see Figure 2.10), and it is constructed as shown in figure 5.12. A running average is performend at each training step in order to update the prototypes per each class, i.e.:

$$\mathbf{\Gamma}_t = \frac{\mathbf{\Gamma}_{t-1} \cdot k_{t-1} + n \cdot \sum_{i=1}^{n} p_i}{k_t} \tag{5.6}$$

where $k_t = k_{t-1} \times n$ is the number of feature vectors considered to build the prototype at time t, n is the number of points per that class at time t, $k_0 = 0$ and $\mathbf{\Gamma}_0 = \mathbf{0}$.

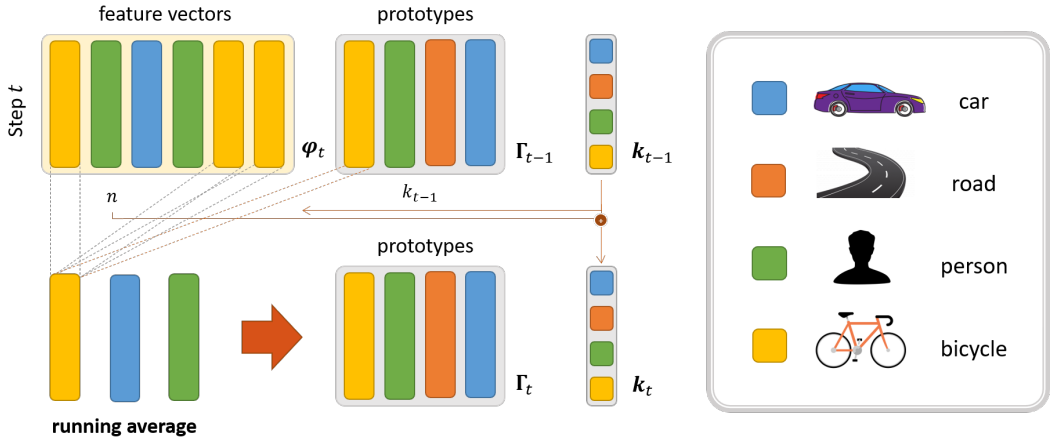


Figure 5.12: Schematic representation of prototypes construction. In the example each color is associated with a different class and $n = 3$ for the yellow class vectors.

Note that in order to understand which feature vectors belong to which ground truth class, the original labels have been downsampled according to the downsampling method used for the features (see Section 2.4). In this way the assignment is univoque. Moreover, the prototypes are built dynamically because each time step more features of the same class are produced and the measurement is adjusted accordingly.

Hence, we compute euclidean distance between the features and the respective prototypes. This distance is then added to the standard cross-entropy loss function, weighted by a factor γ, as it is expressed in the following equation:

$$\mathcal{L}_\Gamma = \mathcal{L}_{micro} + \gamma \cdot \mathcal{L}_{proto}, \quad \mathcal{L}_{proto} = \|\mathbf{\Gamma} - \varphi\| \tag{5.7}$$

where φ_t is a $(1, d_{feats})$ feature vector at time t.

5.3 FAIRNESS

The last concept proposed in this work is the fairness. This term comes from the resources allocation theory and its aim is to give a penalty to the most diffused classes with respect to the rearest ones (Figure 5.13). In mathematical terms the fairness we used can be expressed as:

$$\mathcal{F} = \sum_{k=1}^{M} \mathcal{F}_k, \quad \mathcal{F}_k = \frac{\left(\sum_{i=1}^{n} p_i\right)^2}{n \cdot \sum_{i=1}^{n} p_i^2} \tag{5.8}$$

where p_i is the prediction of the i-th point, and M is the number of macro-classes. With this formula, known in information theory as the **Jain's fairness index** [33], a balancement is introduced inside each cluster.

We can refer to fairness as a regularization term that helps in preserving an homogeneity among the classes in the same macro-class. This term is added to the loss function as expressed in the following equation:

$$\mathcal{L}_{\mathcal{F}} = \mathcal{L}_{micro} + \gamma \cdot (1 - \mathcal{F}) \tag{5.9}$$

where the term included is $(1 - \mathcal{F})$ because the loss function is minimized in the training process.

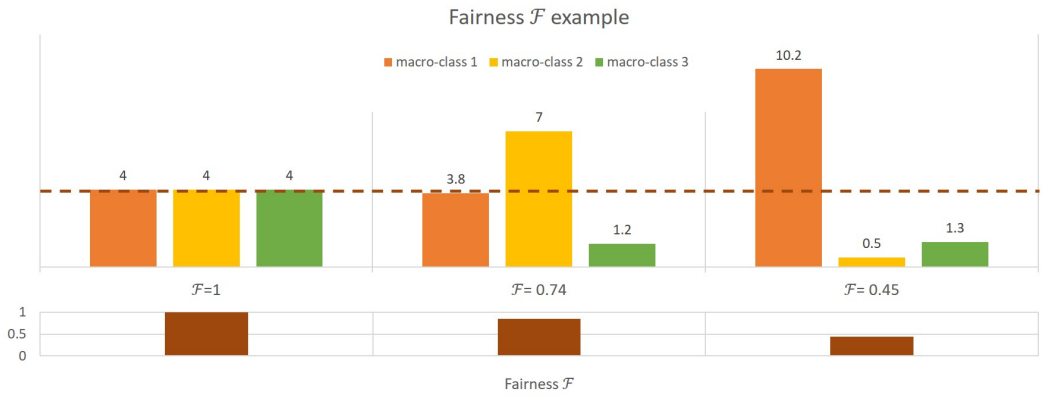


Figure 5.13: The fairness concept in visual terms.

6 Results

In this chapter the results obtained with the proposed learning solutions are reported. Such data are discussed and compared with the state-of-the-art standard framework.

* * *

The following sections provide the results given by the application of the methods explained in the previous chapter. First, some quantitative results are given and each set of methods is broadly discussed. Then, some qualitative examples are presented.

6.1 Implementation details

The dataset has been downloaded from the original online repository of Behley *et al.*[1]. Besides, the repository contains many folders addressed to different tasks and only the parts related to semantic segmentation have been downloaded, i.e., the KITTI Odometry Benchmark Velodyne point clouds, the KITTI Odometry Benchmark calibration data and the SemanticKITTI label data.

As for the architecture, it has been downloaded from the GitHub repository of the original paper from Hu *et al.*[2]. The code is entirely written in Python 3.6 using TensorFlow 1.15 as a framework for training the network. The training has been performed on GPUs, particularly on NVIDIA GeForce RTX 3090 and NVIDIA TITAN RTX devices.

As anticipated, the trainings have been performed using batch size $b = 4$ and validation batch size $b_V = 4$, whilst the original paper used $b = 6$ and $b_V = 20$. The choice was made for memory occupation constraints. However, in the end, the most promising settings with batch size $b = 4$ have been also tested with the original setting to compare our results with the state of the art.

If not stated otherwise, the trainings have been performed for 100 epochs in general, as for the original paper. An exception was made for the curriculum learning trainings with a pretraining stage (Section 5.1.2), where the coarse training was performed on $N = 50$ epochs, whilst the fine tuning on other $N = 100$ epochs for a total of 150 epochs. Another

[1] http://semantic-kitti.org/dataset.html
[2] https://github.com/qingyonghu/RandLA-Net

exception was made for the feature level, where the training was performed only on 40 epochs.

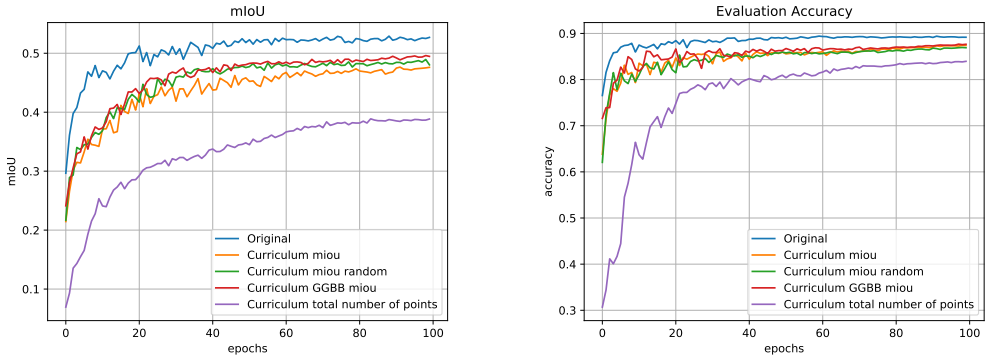
6.2 CURRICULUM RESULTS

In this section the results of the Curriculum Learning methods application, presented in Section 5.1, are reported.

6.2.1 Batch organization

The first set of proposed trials concern the organization of batches in a meaningful way, as explained in 5.1.1. Table 6.1 reports the quantitative results obtained from the proposed methods, while Figure 6.1 shows the temporal development of the accuracy and mIoU metrics.

These results show that the random choice of batches remains the best way to select the samples. In fact, adopting the curriculum approach decreases performance both in the mIoU and accuracy. In addition, the selection based on the number of points is even worse than selecting samples according to the mIoU.



(a) mIoU.

(b) accuracy.

Figure 6.1: Some results of the witty batch organization experiments.

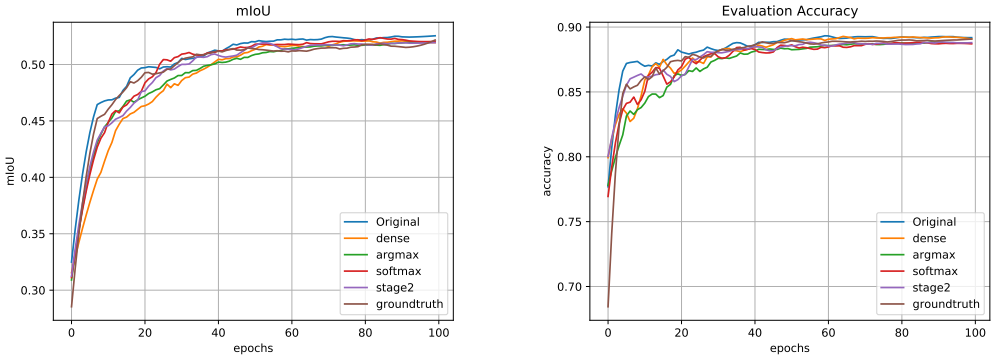
6.2.2 Coarse-to-Fine approaches

This section presents another set of trials, explained in Section 5.1.2, where the samples are selected at random, while the network is trained in a curricular way. The results obtained here are reported in Table 6.2 and Figure 6.3a.
The difference with the standard training scheme is minimal, an improvement can be seen when using these methods with a larger batch size (see Section 6.5.1). The pretraining of the network is useful for performing a coarse separation of the involved classes in order to have a better separation at a second stage (*C2F stage 2*) or an initial label to couple with

Trial name	mIoU	accuracy	epoch
Original	**52.91**	**89.25**	89
total points	38.84	83.95	82
CVPR random	49.16	86.23	86
mIoU	47.65	85.92	89
mIoU random	48.94	84.87	99
mIoU GGBB	**49.61**	**87.43**	96
total points GGBB	40.83	83.05	87

Table 6.1: Quantitative results of the curriculum witty batch organization.

data (*C2F argmax* and *C2F softmax*) while feeding the network in the second stage. This result can be clearly shown in Figure 6.3.



(a) mIoU.

(b) accuracy.

Figure 6.2: Some results of coarse to fine curriculum training schemes. The curves have been smoothed with a Savitzky-Golay filter [36] with window size $w = 9$ and polynomial degree $p = 2$.

6.3 Contrastive results

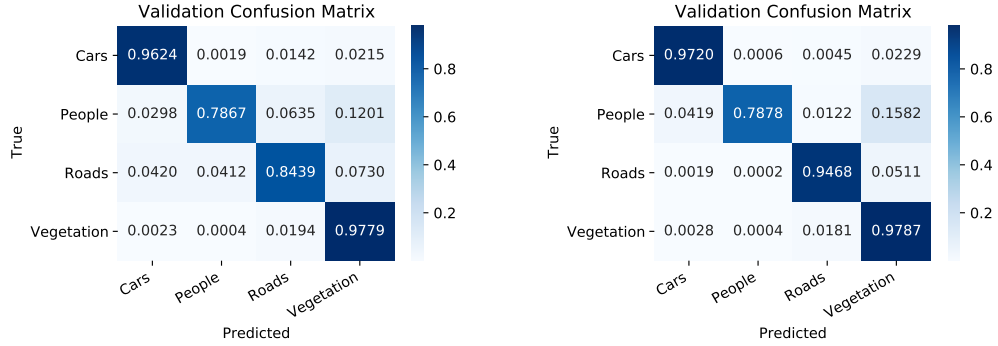
In this section the results of the Contrastive Learning methods application, presented in Section 5.2, are reported.

6.3.1 Alternative loss functions

The results obtained with different loss functions with respect to the standard cross-entropy (introduced in Section 5.2) are grouped in this paragraph. The implementation of the focal loss is here done by choosing $\alpha_t = 4$ and $\gamma = 2$, as suggested in the original paper [21]. Instead, the presented version of the triplet loss is built by choosing $\lambda = 0.1$ and $\mu = 1$.

Trial name	accuracy	mIoU	epoch
Original	89.25	**52.91**	89
C2F dense layer	**89.62**	52.65	77
C2F stage 2	89.07	52.44	68
C2F argmax	88.93	52.33	97
C2F softmax	88.94	52.71	86
C2F groundtruth	89.22	52.41	40
C2F stage 1	96.43	85.80	45

Table 6.2: Quantitative results of coarse-to-fine curriculum training schemes. Note that the last row (*C2F stage 1*) presents huge values of accuracy and mIoU since it refers to macro-classes only.



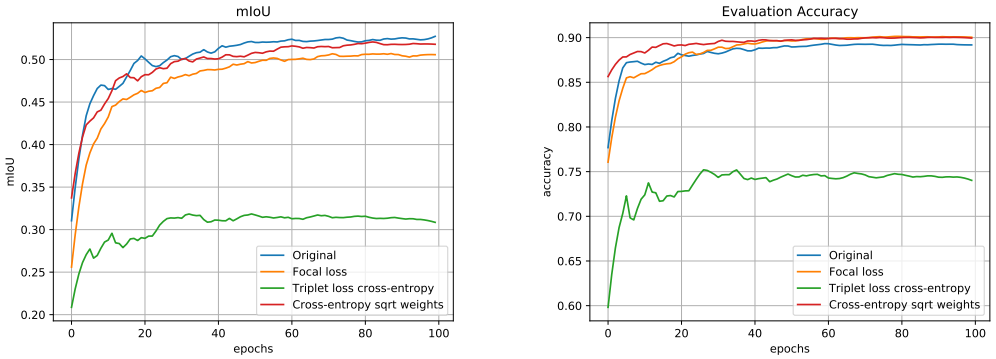
(a) Original.

(b) Coarse-to-fine pretraining.

Figure 6.3: Confusion matrix on the macro-classes of the original setting compared with the one obtained from the first stage coarse-to-fine training.

From Table 6.3 and Figure 6.4b it is possible to notice that there is not an improvement in the mIoU. Several tentatives have been performed by changing the parameter values, but they have very similar results, hence they are not reported here.

Instead, *focal loss* and *cross-entropy loss with square root weights* bring an improvement in terms of accuracy. On the other hand, the improvement is modest, as the point accuracy represents a measure of correctness, weaker than the mIoU for semantic segmentation. In fact, classes with the major number of points are recognized in most cases, while classes with few points are misclassified with high probability, but holding only a few points does not affect much the final result of accuracy. On the other hand, their misclassification affects the mIoU in a greater percentage and, in fact, it is lower than in the original case. Hence, these losses overfit the data and do not outperform the original one.



(a) mIoU.

(b) accuracy.

Figure 6.4: Some results of the experiments with different standard loss functions and weighting schemes.

Trial name	mIoU	accuracy	epoch
Original	**52.91**	89.25	89
Focal loss	51.05	**90.08**	98
Triplet loss cross entropy	32.06	75.49	34
Dice loss	11.43	60.20	45
Cross-entropy square root weights	52.27	**90.14**	96

Table 6.3: Quantitative results of the experiments with different standard loss functions and weighting schemes.

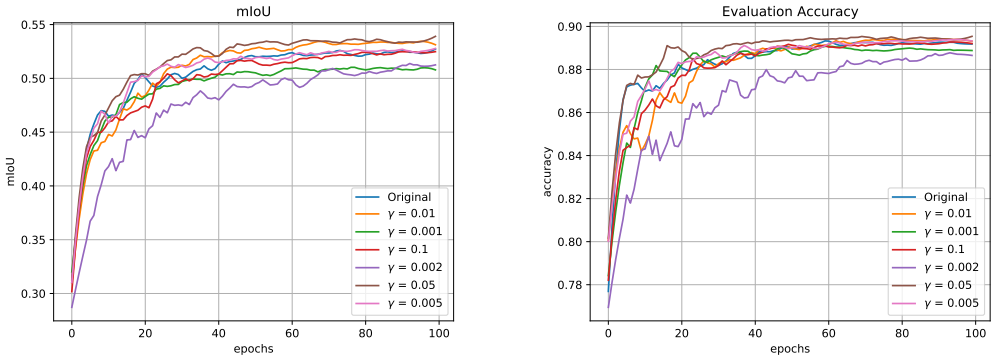
6.3.2 Hierarchical loss

In this section, a discussion of the results of the hierarchical loss, introduced in Section 5.2.1.5, is presented as a standalone. In fact, this loss function has proved to be very effective both in terms of accuracy and mIoU and to outperform the state of the art whenever an accurate choice of the γ parameter (see Equation 5.5) is performed.
Figure 6.5 and Table 6.4 show that if the macro-classes are weighted $\gamma = 0.05$ times the micro-classes, the best configuration is achieved and there is a substantial gain of 1.31% in mIoU with respect to the standard approach. The accuracy is also outperformed.

The confusion matrix of Figure 6.6 shows that following this approach, the micro-classees belonging to different macro-classes have a lower probability with respect to the original setting of being confused with micro-classes that belong to different macro-classes.

6.3.3 Feature level

The loss function that includes the euclidean distance measurement on the prototypes (Equation 5.7) has been tested only with parameter $\gamma = 0.1$, obtaining an overall accu-



(a) mIoU.

(b) accuracy.

Figure 6.5: Some results of the experiments with hierarchical loss and different values of parameter γ. The curves have been smoothed with a Savitzky-Golay filter [36] with window size $w = 9$ and polynomial degree $p = 2$.

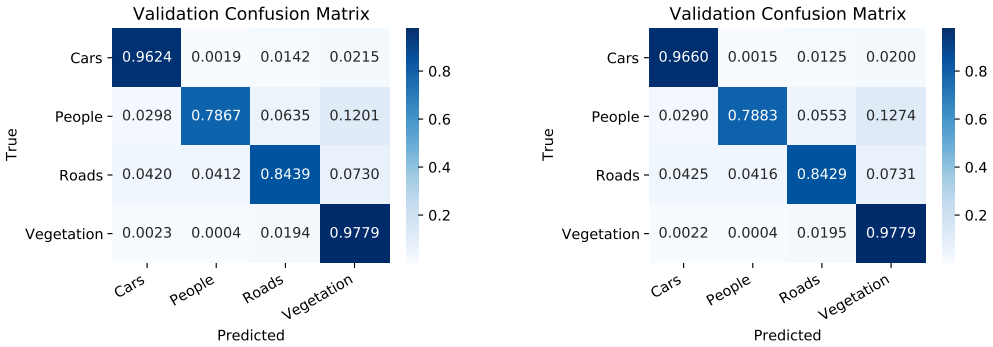
Trial name	accuracy	mIoU	epoch
Original	89.25	52.91	89
$\gamma = 0.010$	**89.43**	**53.73**	95
$\gamma = 0.001$	89.15	51.46	72
$\gamma = 0.100$	89.05	52.91	73
$\gamma = 0.002$	88.97	52.23	62
$\gamma = 0.050$	**89.54**	**54.22**	78
$\gamma = 0.005$	**89.39**	52.90	98

Table 6.4: Quantitative results of the trials with hierarchical loss for different values of parameter γ.

racy value of 88.18%. Only partial results have been obtained here and deeper analyses have to be conducted. In fact, only 40 epochs have been performed due to resource constraints and time limits.
However, the mIoU and accuracy curves (Figure 6.7) show that 40 epochs are not sufficient and the model is still improving its performance. In fact, the best accuracy and mIoU values for the model have been obtained at the last epoch (i.e., epoch 40). Moreover, the curves present a regular increasing behaviour, showing that the prototypical loss, acting at feature level, adds a lot of regularization with respect to the standard one.

6.4 FAIRNESS RESULTS

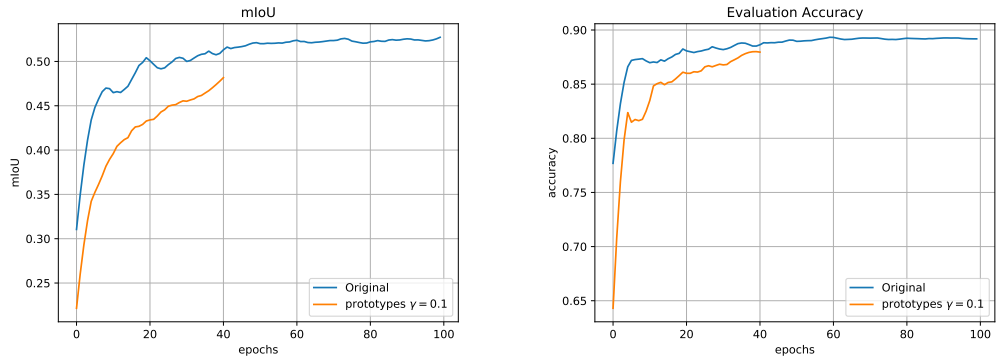
The results of the trials where a fairness term is added to the standard cross-entropy are reported in this section. In particular, several values for γ parameter (Equation 5.9) are tried. Table 6.6 and figure 6.8 report these results. Notice that by increasing the weight



(a) Original.

(b) Hierarchical Loss.

Figure 6.6: Confusion matrix on the macro-classes of the original setting compared with the one obtained from the training with hierarchical loss with parameter $\gamma = 0.05$.



(a) mIoU.

(b) accuracy.

Figure 6.7: Results of the experiments with prototypical loss with $\gamma = 0.1$. The curves have been smoothed with a Savitzky-Golay filter [36] with window size $w = 9$ and polynomial degree $p = 2$.

of the γ parameter, the results improve, reaching values even better than the original state-of- the-art configuration. However, the convergence is a little slower.

The main result that is achieved with this method, on the other hand, is actually a *fairness* among classes. Figure 6.9 shows the mIoU per class compared to the original baseline approach. It is clearly visible that the problem of imbalance among classes (dealt in Chapter 4) persists, but it is reasonably compensated for by our introduced term $\mathcal{F}$.
This result is confirmed by the standard deviation in the mIoU measurement, as it shows

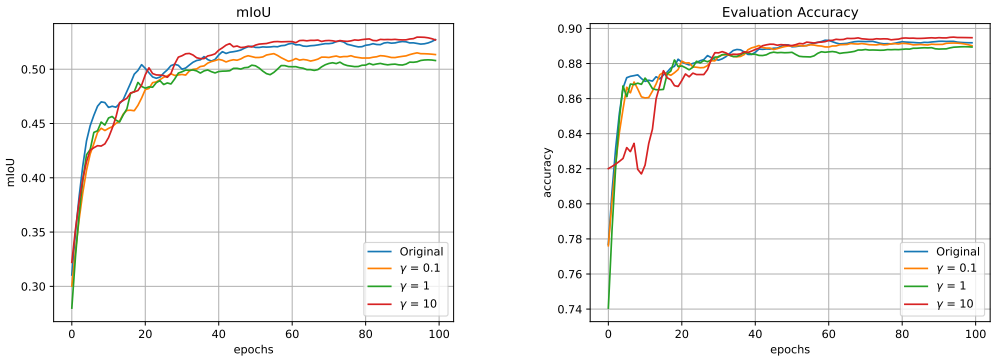
Trial name	accuracy	mIoU	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign	epoch
Original	**89.3**	**52.9**	92.9	12.7	28.4	70.0	35.6	52.3	69.0	0.0	91.2	37.1	76.0	1.5	87.6	43.7	85.0	60.2	73.7	51.6	36.9	89
prototypes	88.2	48.5	91.5	5.9	25.0	36.1	34.4	41.1	66.0	0.0	89.6	35.4	74.5	0.3	86.0	41.6	83.7	54.2	74.4	48.4	33.1	40

Table 6.5: Quantitative results of the contrastive loss at the feature level.

a reasonable decrease from $\sigma = 29.0$ to $\sigma = 28.2$. In addition, we computed also the MSE and the entropy H, as:

$$H = \sum_{i=1}^{M} p_i \cdot \log_2 \frac{1}{p_i} \tag{6.1}$$

where M is the number of classes and p_i is the mIoU value obtained for that class. All these measurements, which provide another proof of the effectiveness of our methods and are reported in Table 6.6.



(a) mIoU.

(b) accuracy.

Figure 6.8: Some results of the experiments with cross-entropy loss with fairness and different values of parameter γ. The curves have been smoothed with a Savitzky-Golay filter [36] with window size $w = 9$ and polynomial degree $p = 2$.

6.5 FURTHER EXPERIMENTS

Further experiments have been led by changing the batch sizes b, b_V to the original dimensions and by testing the promising methods jointly. Instead, the methods that gave poor results with $b = b_V = 4$ have been neglected. The results of these tests are presented in the following sections.

Trial name	accuracy	mIoU	epoch	σ	MSE	entropy
Original	89.25	52.91	89	29.0	3644.2	6.46
$\gamma = 0.1$	89.05	51.17	95	28.7	3591.5	6.55
$\gamma = 1$	89.31	51.81	55	**28.2**	3601.4	**6.61**
$\gamma = 10$	**89.57**	**53.22**	79	**28.2**	**3714.7**	6.43

Table 6.6: Quantitative results of cross-entropy loss with fairness for different values of γ parameter.

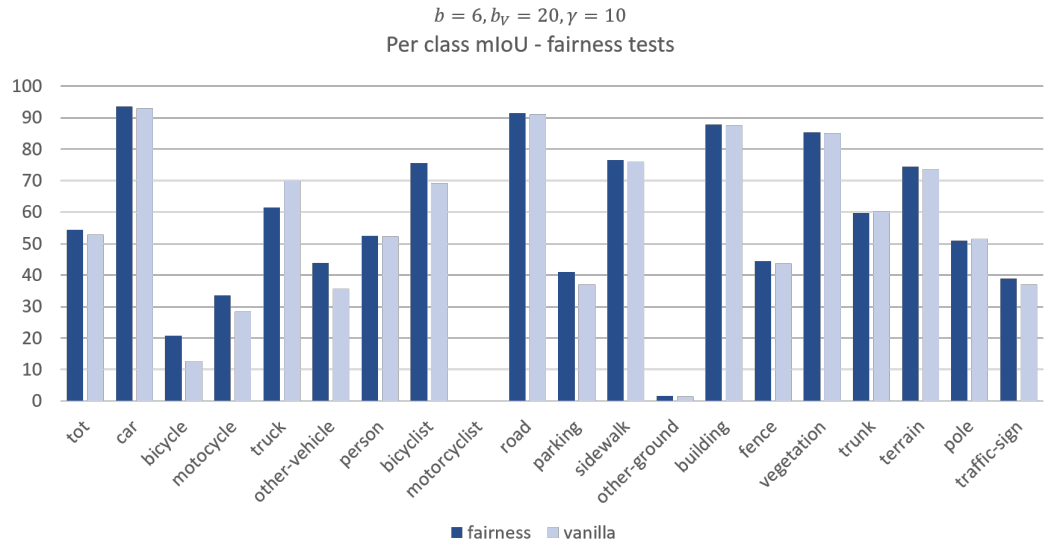
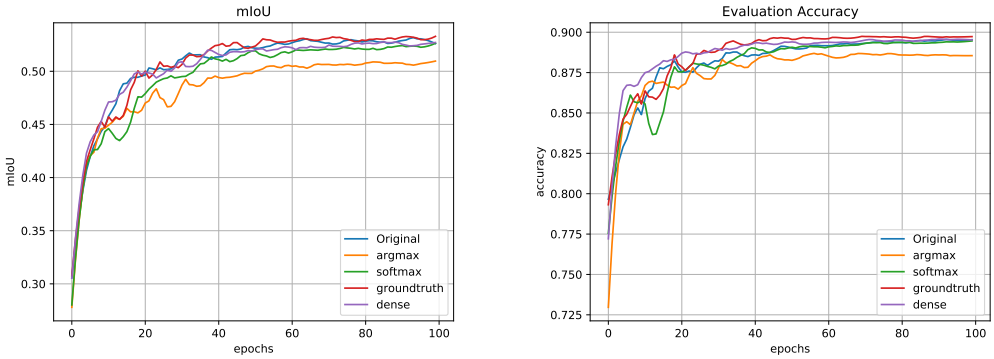


Figure 6.9: The mIoU per class for the *fairness* with respect to the standard cross-entropy. Parameters are set to $b = 6$, $b_V = 20$, $\gamma = 10$. Notice that there is a rise in the classes with low values, e.g., *bicycle*, *parking*.

6.5.1 Tests with original batch size

Experiments have been conducted with the original batch size $b = 6$ and the original validation batch size $b_V = 20$, to compare our approaches with the state of the art. Table A.1 reports the tests led before, while Table A.3 the ones with the increased batch dimension. The per class mIoU is reported here to underline the results achieved with respect to each class.
We can observe that the behavior of these results is similar, and the performances are slightly improved. Note that our results with the standard configuration achieve little worse performance with respect to what is declared in the original paper [15].
The main result here is achieved with coarse-to-fine that gains in performance while increasing the batch dimension. Figure 6.10 and Table A.3 report these results.



(a) mIoU.

(b) accuracy.

Figure 6.10: Some results of coarse-to-fine curriculum training schemes with batch size $b = 6$ and validation batch size $b_V = 20$. The curves have been smoothed with a Savitzky-Golay filter [36] with window size $w = 9$ and polynomial degree $p = 2$.

Trial name	accuracy	mIoU	epoch
Original	89.59	53.36	93
fairness $\gamma = 0.1$	89.68	53.00	68
fairness $\gamma = 1$	89.73	53.37	63
fairness $\gamma = 10$	89.71	**54.40**	94
hierarchy $\gamma = 0.01$	89.79	**54.20**	37
hierarchy $\gamma = 0.001$	89.44	52.71	63
hierarchy $\gamma = 0.05$	89.60	53.79	89
hierarchy $\gamma = 0.005$	89.38	52.66	92
C2F dense layer	89.63	53.00	96
C2F stage 2	89.78	**53.81**	56
C2F argmax	88.81	51.07	42
C2F softmax	89.54	52.86	99
C2F groundtruth	89.85	53.60	87

Table 6.7: Results of tests with $b = 6$ and $b_V = 20$.

6.5.2 Joint experiments

The final set of experiments has been conducted by testing some of the methods giving the best performances, jointly. In particular, the fairness index and the hierarchical loss have been mixed, and experiments have been led to training the network in stages. The results are reported in Table A.2.

Notice that all these experiments bring an improvement with respect to the original training scheme. However, some combinations are not very effective, while other combinations

improve even the results on the singular components. The best combination is obtained while mixing the three methods in such a way that the loss function results:

$$\mathcal{L} = \mathcal{L}_{micro} + \gamma_1 \cdot \mathcal{L}_{macro} + \gamma_2 \cdot (1 - \mathcal{F}) \tag{6.2}$$

where $\gamma_1 = 0.05$, $\gamma_2 = 10$ and the model is trained for 50 epochs on the coarse labels and for 100 on the fine labels. This configuration shows an remarkable improvement of 1.5% mIoU with respect to the standard model trained with our hardware.
On the other hand, the number of configurations tested in this work is limited and further improvements can be possibly achieved with finer tuning of the parameters involved.

Trial name	accuracy	mIoU
Original	89.25	52.91
fairness $\gamma = 10$ + C2F	87.96	53.00
hierarchy $\gamma = 0.01$ + fairness $\gamma = 10$	89.86	53.87
hierarchy $\gamma = 0.01$ + C2F	89.60	52.95
hierarchy $\gamma = 0.05$ + fairness $\gamma = 10$	89.56	53.99
hierarchy $\gamma = 0.05$ + C2F	89.53	54.12
hierarchy $\gamma = 0.05$ + fairness $\gamma = 10$ + C2F	89.69	**54.34**

Table 6.8: Results of joint tests with $b = 4$, $b_V = 4$.

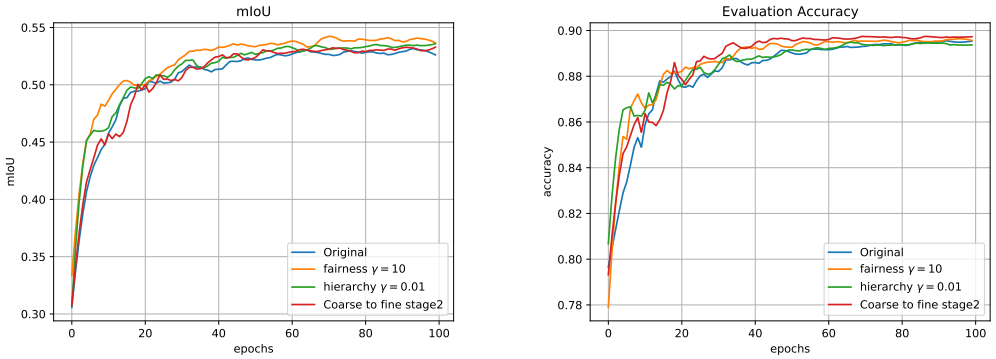
6.6 Final Discussion

Overall, the results underline an improvement in the performance of the network. The experiments have proved to outperform the state of the art not only in terms of total mIoU and accuracy, but also as regards the speed of convergence to the optimal value. In fact, if we restrict our measurements to a few epochs only, we can notice that satisfactory results are achieved by our methods.

The hierarchical learning applied to the loss function has shown to be efficient in achieving good class separation at a macro level (Figure 6.6) as well as at a micro level. Similarly, the curricular approach that provides training in stages has been proven to improve performance by separating classes into subsequent steps. Finally, fairness results, as expected, in a more evenly distributed distribution of the learnt classes.

6.6.1 Qualitative Results

In this section, the qualitative results of some of the methods tested are presented. In particular, Figure 6.12 shows some example point clouds labeled according to our methods (hierarchical loss with $\gamma = 0.05$, fairness loss with $\gamma = 10$, Coarse-to-Fine), compared to the baseline approach and the ground truth.



(a) mIoU.

(b) accuracy.

Figure 6.11: Results of the best experiments with original batch size. The curves have been smoothed with a Savitzky-Golay filter [36] with window size $w = 9$ and polynomial degree $p = 2$.

The specification in terms of number of points and overall performance for each of the selected point clouds is described in Table 6.9. Note that our methods obtain better values for the MSE and a lower percentage of wrongly classified points in each of them, with respect to vanilla training. Note that in the computation of such values, the *unlabeled* class has not been taken into consideration. The most common misclassifications can be seen in the table and are highlighted in the figure.

Figure 6.13 and Table 6.10 are instead referred to by the labelling according to the macro-classes. In that, we can see an accordance with the confusion matrix of Figure 6.3 as regards the coarse-to-fine approach and Figure 6.6 for the hierarchical loss.

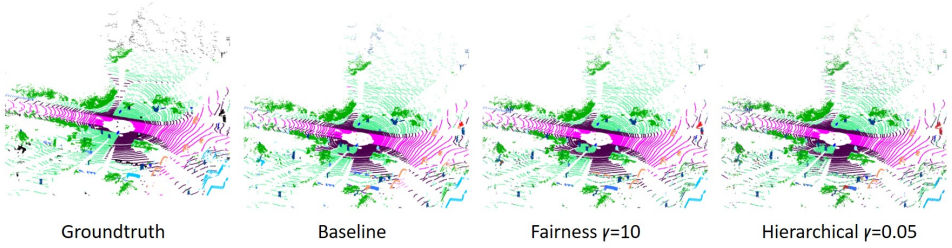
The qualitative results point out the effectiveness of our methods. Specifically, we can notice some particulars marked by red circles in the examples (e.g., the *building* in Figure 6.12c). Note that the results obtained with the fairness model are noisier than the other methods, but they are able to recognize classes that are less frequent, like class *people*, represented by the blue points in Figure 6.13b. The hierarchical approach, from a qualitative point of view, obtains results similar to the baseline, but improved.

		unlabeled	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign	MSE	% of wrong points
08x000000	**gt**	6567	1503	20	0	0	0	160	0	0	12024	84	15677	0	2120	518	23523	2830	22066	311	706		
	vanilla	0	2548	0	0	0	172	0	0	0	19069	2861	7026	0	791	3596	32834	1621	19428	368	0	44765	25.09%
	fairness	0	1997	0	21	0	208	0	0	1	12571	55	18727	40	1618	925	24856	4964	20498	1005	623	11291	6.66%
	hierarchical	0	1845	0	2	1	394	62	0	0	13104	104	20098	31	1275	614	25088	4823	18846	1047	775	14933	8.80%
08x000090	**gt**	11015	1427	50	0	0	0	64	1038	0	12422	81	10822	1	3386	198	13501	1722	25880	389	458		
	vanilla	0	3068	0	6	0	0	0	883	0	11760	835	9391	495	3989	3377	17732	2526	27596	737	59	16537	10.75%
	fairness	0	2661	0	0	0	1	15	1612	0	16690	281	9815	18	2516	2440	14910	1758	28740	958	39	15805	10.27%
	hierarchical	0	2282	1	0	0	30	4	1237	0	14725	48	9670	0	2641	3060	14427	2115	31120	1015	79	15853	10.30%
08x001010	**gt**	2660	1746	20	116	0	977	0	0	0	16999	5141	7530	0	3639	18	33835	1004	16327	296	6		
	vanilla	0	2548	0	0	0	172	0	0	0	19069	2861	7026	0	791	3596	32834	1621	19428	368	0	17820	10.01%
	fairness	0	2572	0	0	8	698	0	0	0	20888	1961	7469	26	1544	880	33749	1054	19077	388	0	14346	8.06%
	hierarchical	0	2263	0	0	8	585	0	0	0	17177	2868	9400	30	3074	1525	32160	1763	18976	463	22	12742	7.16%

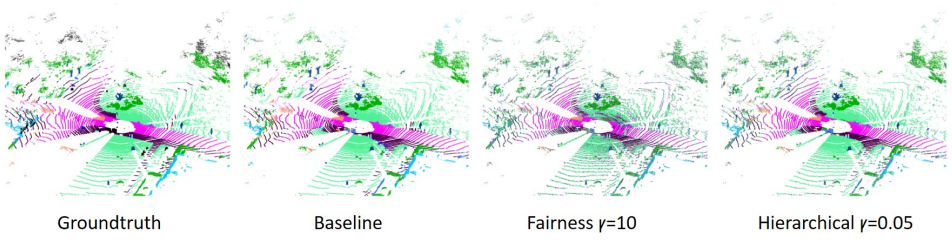
Table 6.9: Labelling specifications of point clouds 08x000000, 08x000090 and 08x001010. The table reports the true labels and labels per each method.

		unlabeled	cars	people	roads	vegetations	MSE	% of wrong points
08x000000	**gt**	6567	1523	160	49851	30008		
	vanilla	0	2720	0	48384	39210	12026	7.00%
	fairness	0	2226	1	51891	33991	6885	4.06%
	hierarchical	0	2242	62	52183	33622	6763	3.99%
08x000090	**gt**	11015	1477	1102	49206	19654		
	vanilla	0	3074	883	50077	28420	11453	7.44%
	fairness	0	2662	1627	55544	22621	11015	7.16%
	hierarchical	0	2313	1241	55563	23337	11015	7.16%
08x001010	**gt**	2660	2859	0	45997	38798		
	vanilla	0	2720	0	48384	39210	2938	1.65%
	fairness	0	3278	0	49421	37615	5026	2.82%
	hierarchical	0	2856	0	48451	39007	2666	1.50%

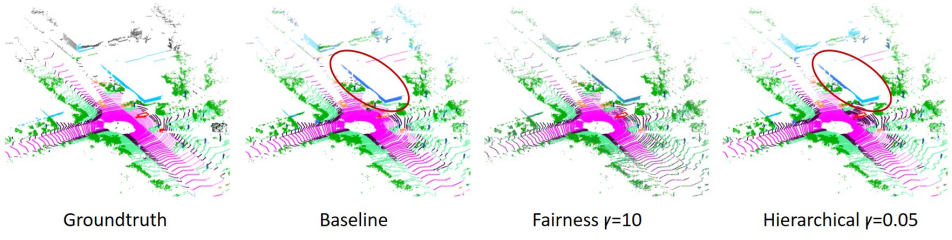
Table 6.10: Macro classes labelling specifications of point clouds 08x000000, 08x000090, 08x001010.



(a) 08x000000.

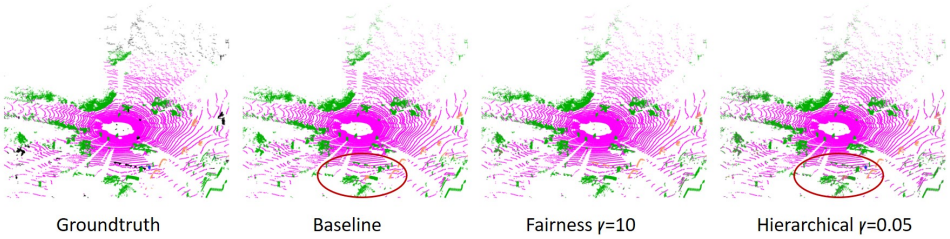


(b) 08x000090.

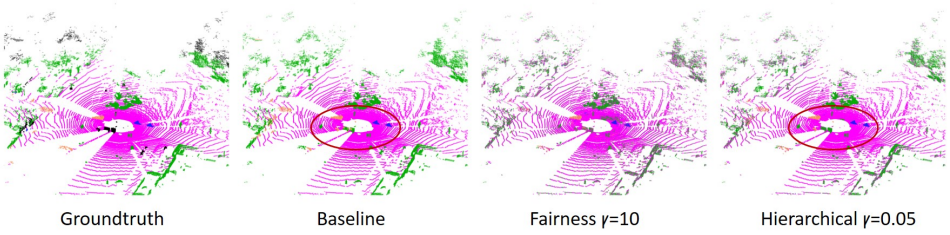


(c) 08x001010.

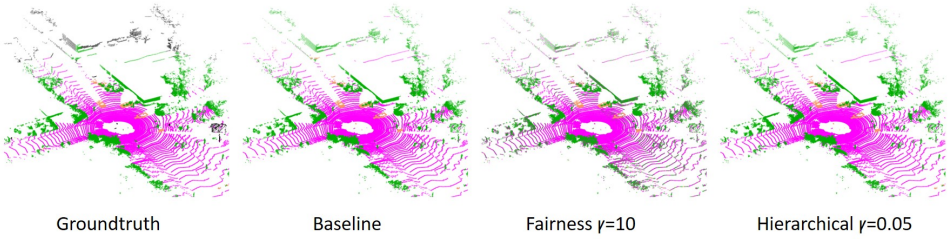
Figure 6.12: Qualitative results of the best experiments compared to the baseline and grountruth with examples 08x000090, 08x001010, 08x000000.



(a) 08x000000.



(b) 08x000090.



(c) 08x001010.

Figure 6.13: Qualitative results of the best experiments compared to the baseline and grountruth with examples 08x000090, 08x001010, 08x000000.

7 Conclusions

In this chapter the overall work is summarized while final conclusions and remarks are drawn. Finally, a perspective towards future works is presented.

* * *

The task of LiDAR semantic segmentation is an actual problem that must be addressed in the development of future applications such as self-driving cars. The current GPU capabilities are a limited resource for providing this mean, and this must be considered while approaching the problem. In this thesis the task has been developed by focusing on learning strategies, in order to leverage state-of-the-art architectures while improving learning performance. The goal is accomplished by using methods that aim at separating different classes and gradually learning concepts by progressively refining the learned knowledge.

Preliminary analyses led on the SemanticKITTI [2] dataset have pointed out the overall imbalance of classes and a suggestion of a witty subdivision of them into macro categories. Such subdivision has been obtained through spectral clustering algorithm and is based on the mIoU metric of a general supervised model. These results have been exploited to tackle LiDAR semantic segmentation task by means of a well-known point-based architecture, RandLA-Net [15].

The first idea applied was to adopt a curriculum learning approach based on this macro-classes subdivision. Therefore, methods based on a hierarchical loss function have been investigated, as well as different training schemes that involve a pretraining of the network on the macro-classes to refine the knowledge of the whole set. These methods have proven to be very effective as they provide an improved overall distinction between categories with respect to the standard one.

In addition, the class imbalance has led us to include homogeneity terms in the loss to have a more uniform accuracies with respect to the various classes. In this, the fairness index has played its role, showing an improvement on the least frequent classes while lowering the standard deviation measure in the mIoU.

7.1 FUTURE WORKS

The perspective of improvement is very broad, especially considering working in the feature space. For future work, the purpose is to develop the prototypical representation by trying different application methods. For example, prototypes are now built starting from the very first epoch, where the weights are not stable yet. The idea is to start their construction later, or progressively weight more the new incoming features in the process of prototype construction.

Progressive training can be accomplished even when using other methods. For example, in the hierarchical loss, the parameters can decrease as the learning rate increases, and similarly, the ground truth labels or the predictions can be included progressively with the training time. Clearly, these solutions can be integrated and tested jointly.

Overall, this thesis's contribution can be considered as a starting point for wider work. In fact, the methods applied here can even be tested on diverse architectures and using different datasets for LiDAR semantic segmentation, given that our methods are not dataset or architecture dependent. In particular, the idea is to use synthetic data in the training process, which requires adopting new techniques and processes to ease their integration with real ones. Our methods are finally suitable even for tasks different from LiDAR semantic segmentation, and this is another perspective that must be kept into consideration.

Appendix

In this chapter additional data concerning the analyses led on the dataset are reported.

* * *

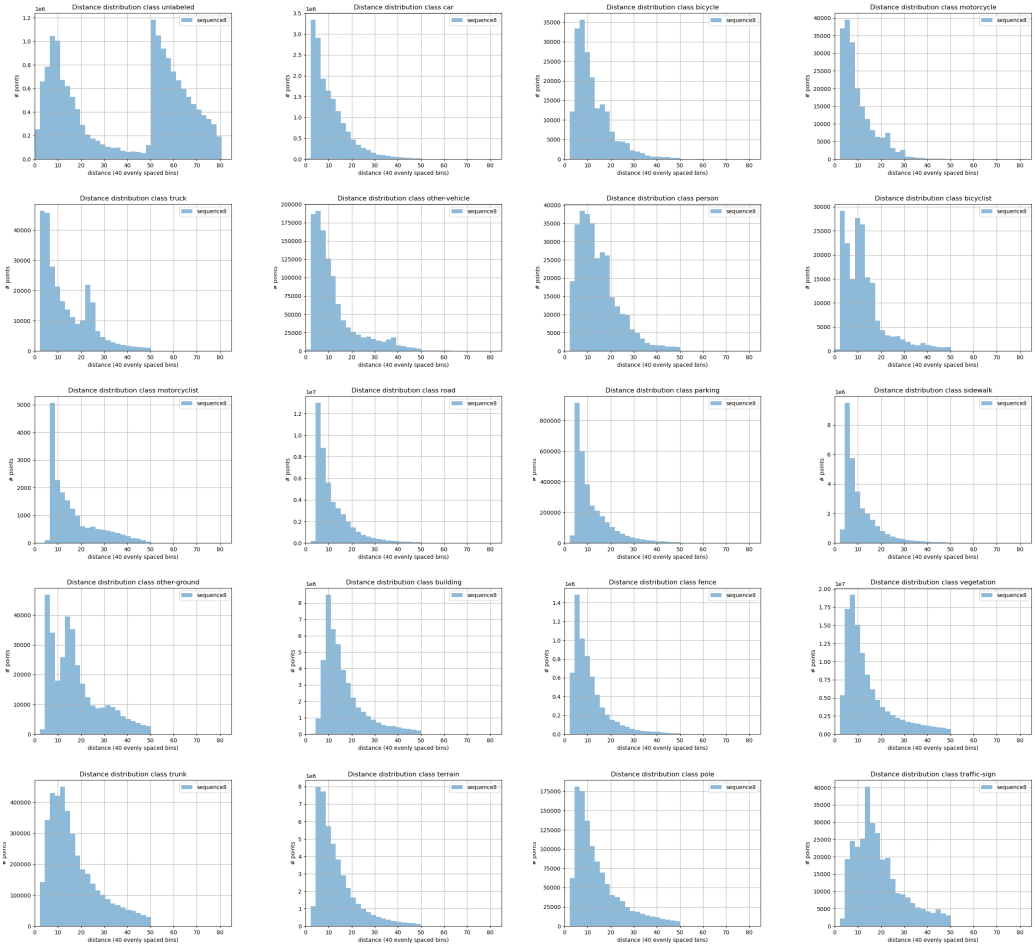


Figure A.1: Distance distribution per each class in test sequence.

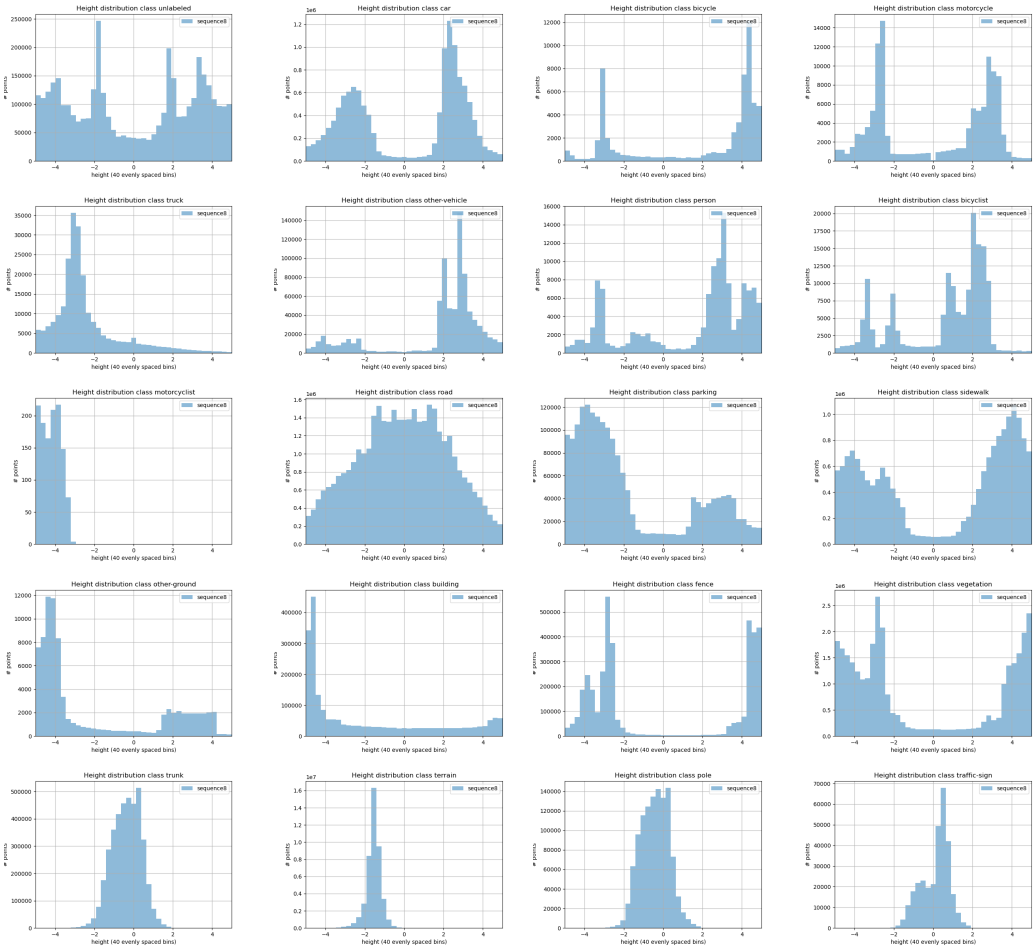


Figure A.2: Height distribution per each class in test sequence.

Figures A.1 and A.2 report the distances and heights per class in the test set. No further details are provided here since the plots have been discussed before in Chapter 4. Figure A.3 represents an extended version of Figure 4.4 where the predictions are subdivided per class. Finally, Tables A.1, A.3 and A.2 report the overall results obtained with our methods that gave significant performance in terms of mIoU and accuracy.

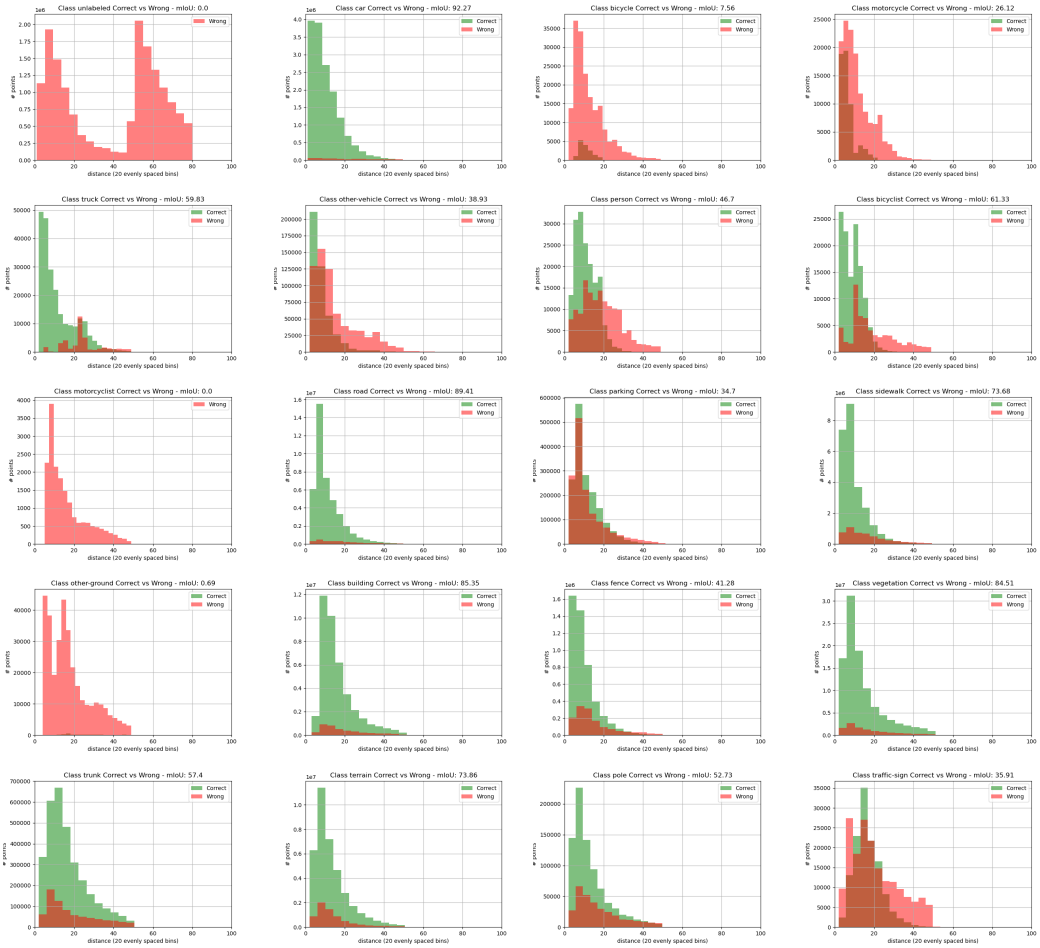


Figure A.3: Distance distribution of the correctly and wrongly classified points.

Trial name	mIoU	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign
Original	52.91	92.94	12.67	28.43	70.02	35.62	52.31	69.03	0.00	91.15	37.06	75.96	1.45	87.62	43.68	84.95	60.23	73.67	51.56	36.93
hierarchical $\gamma = 0.01$	53.73	93.20	10.67	33.16	73.76	41.64	50.70	70.03	0.00	91.35	39.46	**76.68**	1.53	87.63	41.38	84.70	59.64	**75.00**	**52.07**	38.35
hierarchical $\gamma = 0.001$	51.46	92.50	11.79	33.51	58.98	35.33	49.01	63.16	0.00	90.84	38.46	75.68	0.52	87.02	39.62	84.33	57.99	75.26	49.52	34.20
hierarchical $\gamma = 0.1$	52.91	92.99	13.26	29.35	75.41	37.66	49.41	63.86	0.00	91.12	40.49	76.17	0.14	87.81	42.22	84.78	59.15	74.99	50.97	35.42
hierarchical $\gamma = 0.002$	52.23	92.46	13.22	26.35	71.93	39.04	48.81	62.51	0.00	91.06	39.39	75.40	1.48	86.81	40.10	84.19	58.43	72.69	51.29	37.18
hierarchical $\gamma = 0.05$	**54.22**	**93.48**	14.67	**36.07**	72.95	**41.95**	49.46	64.21	0.00	91.33	39.52	76.02	1.93	**88.22**	**45.52**	**85.67**	**62.32**	73.99	51.72	**41.12**
hierarchical $\gamma = 0.005$	52.90	92.89	13.15	31.00	69.39	34.02	49.53	69.12	0.00	91.44	39.06	76.61	1.53	87.59	41.94	85.00	59.26	74.39	50.94	38.30

Trial name	mIoU	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign
Original	52.91	92.94	12.67	28.43	70.02	35.62	52.31	69.03	0.00	91.15	37.06	75.96	1.45	87.62	43.68	84.95	60.23	73.67	51.56	36.93
fairness $\gamma = 0.1$	51.17	92.41	11.81	26.78	61.63	31.65	48.91	61.19	0.00	91.48	37.82	76.00	0.26	87.15	41.23	84.57	58.90	73.54	50.03	36.92
fairness $\gamma = 1$	51.81	92.77	13.60	25.43	65.43	39.45	46.55	60.77	0.00	91.27	38.73	75.74	1.15	87.14	42.65	84.73	58.27	74.59	49.38	36.84
fairness $\gamma = 10$	**53.22**	92.97	14.79	29.40	72.09	34.44	**54.22**	64.60	0.00	91.31	40.56	76.39	0.38	87.60	43.80	85.03	60.93	73.46	51.41	37.75

Trial name	mIoU	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign
Original	**52.91**	92.94	12.67	28.43	70.02	35.62	52.31	69.03	0.00	91.15	37.06	75.96	1.45	87.62	43.68	84.95	60.23	73.67	51.56	36.93
C2F dense layer	52.65	93.31	4.87	29.78	**76.93**	41.49	48.54	67.88	0.00	**91.50**	37.65	76.26	1.47	87.89	42.79	84.58	59.48	73.82	44.90	37.17
C2F stage 2	52.44	92.52	11.23	33.35	61.19	39.29	47.31	66.22	0.00	90.75	37.67	75.25	2.97	87.68	42.78	84.47	59.47	73.46	50.61	40.07
C2F argmax	52.33	92.73	16.13	30.00	70.20	38.35	45.30	64.60	0.00	91.07	38.98	76.04	0.60	87.21	41.47	83.94	59.15	72.90	51.43	34.19
C2F softmax	52.71	92.91	14.84	34.89	60.23	38.89	50.91	**70.15**	0.00	91.48	**40.68**	76.31	0.93	86.77	39.36	83.77	60.92	72.98	50.41	35.04
C2F groundtruth	52.41	92.59	**17.09**	33.01	69.46	35.79	49.33	63.93	0.00	90.66	37.10	75.10	**3.59**	86.77	39.65	84.35	58.74	73.62	50.37	34.55

Table A.1: Overall quantitative results on per class mIoU using $b = 4$, $b_V = 4$.

Trial name	mIoU	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign
Original	52.91	92.94	12.67	28.43	70.02	35.62	52.31	69.03	0.00	91.15	37.06	75.96	1.45	87.62	43.68	84.95	60.23	73.67	51.56	36.93
fairness $\gamma = 10$ + C2F	53.00	93.3 0	11.31	29.96	76.27	40.53	49.24	55.34	0.00	91.49	39.06	76.75	1.59	87.63	45.36	85.64	60.30	73.90	52.59	36.80
hierarchy $\gamma = 0.01$ + fairness $\gamma = 10$	53.87	93.23	16.33	32.23	66.55	41.18	52.92	67.67	0.00	91.79	39.91	76.65	0.45	87.78	43.96	85.41	62.13	73.26	52.95	39.09
hierarchy $\gamma = 0.01$ + C2F	52.95	93.18	15.51	27.69	72.37	40.62	44.61	67.77	0.00	91.54	39.36	76.83	0.68	87.78	42.24	84.99	59.62	75.51	51.16	34.63
hierarchy $\gamma = 0.05$ + fairness $\gamma = 10$	53.99	92.83	17.72	30.83	73.59	38.74	49.96	69.87	0.00	91.64	42.39	76.92	3.29	87.33	44.12	85.15	58.52	73.89	51.15	37.91
hierarchy $\gamma = 0.01$ + fairness $\gamma = 10$ + C2F	54.12	93.12	15.42	34.01	63.76	40.66	53.83	66.63	0.00	91.54	40.78	77.11	1.34	89.16	48.93	85.47	61.58	73.45	52.48	39.06
hierarchy $\gamma = 0.05$ + fairness $\gamma = 10$ + C2F	**54.34**	93.52	14.30	32.01	73.34	40.46	51.26	66.84	0.00	91.61	41.73	76.18	1.27	88.92	47.89	84.24	62.85	73.92	52.39	39.77

Table A.2: Overall quantitative results on per class mIoU using $b = 4$, $b_V = 4$.

Trial name	mIoU	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign
Original	53.36	93.37	13.42	30.30	69.89	39.45	50.55	67.24	0.00	91.81	41.03	76.96	0.97	87.73	43.73	84.90	60.35	73.38	50.88	37.92
fairness $\gamma = 0.1$	53.00	93.20	14.51	28.79	64.69	47.43	46.9	62.73	0.00	91.39	39.29	76.75	1.02	87.25	43.41	85.71	62.17	74.41	50.56	36.81
fairness $\gamma = 1$	53.37	93.48	16.82	31.08	64.02	43.65	46.84	65.23	0.00	91.61	39.43	76.76	3.70	87.72	43.5	85.37	59.03	74.28	51.81	39.66
fairness $\gamma = 10$	**54.40**	93.47	20.83	33.60	61.46	43.94	52.45	75.68	0.00	91.50	41.02	76.62	1.66	87.73	44.36	85.36	59.70	74.40	50.96	38.95

Trial name	mIoU	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign
Original	53.36	93.37	13.42	30.30	69.89	39.45	50.55	67.24	0.00	91.81	41.03	76.96	0.97	87.73	43.73	84.90	60.35	73.38	50.88	37.92
hierarchical $\gamma = 0.01$	**54.20**	93.50	14.84	32.85	69.56	43.25	52.14	67.43	0.00	91.8 0	40.92	77.07	5.09	87.68	43.03	85.4	61.19	74.52	50.97	38.63
hierarchical $\gamma = 0.001$	52.71	93.43	13.49	28.14	57.91	48.35	49.40	64.41	0.00	91.49	40.11	76.73	0.53	88.25	42.09	84.15	60.52	73.78	51.46	37.34
hierarchical $\gamma = 0.05$	**53.79**	93.20	16.54	25.59	71.80	44.32	49.42	69.89	0.00	91.47	38.76	76.58	6.49	87.88	43.34	85.06	60.32	73.33	50.80	37.28
hierarchical $\gamma = 0.005$	52.66	93.42	17.34	30.14	54.68	43.11	49.53	64.95	0.00	91.68	41.37	77.2	1.41	86.86	41.19	84.46	60.69	74.17	50.29	37.96

Trial name	mIoU	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign
Original	53.36	93.37	13.42	30.3	69.89	39.45	50.55	67.24	0.00	91.81	41.03	76.96	0.97	87.73	43.73	84.9	60.35	73.38	50.88	37.92
C2F dense layer	53.00	93.30	11.31	29.96	76.27	40.53	49.24	55.34	0.00	91.49	39.06	76.75	1.59	87.63	45.36	85.64	60.30	73.90	52.59	36.80
C2F stage 2	**53.81**	93.78	17.20	34.36	62.16	43.85	50.35	68.32	0.00	91.30	41.83	76.75	1.13	87.95	45.15	85.28	59.41	73.75	50.82	39.01
C2F argmax	51.07	92.36	6.18	27.82	74.95	35.60	43.82	58.79	0.00	90.54	40.95	75.18	0.35	85.81	39.71	83.87	55.98	73.85	48.13	36.37
C2F softmax	52.86	93.21	11.45	23.53	73.16	41.66	47.80	62.83	0.00	91.53	39.20	76.78	1.54	87.75	44.38	85.08	59.83	74.42	51.20	38.94
C2F groundtruth	**53.60**	93.57	14.02	25.54	74.85	47.47	48.69	63.29	0.00	91.85	42.28	77.15	0.43	87.50	42.72	85.56	59.51	74.62	51.19	38.22

Table A.3: Overall quantitative results on per class mIoU using $b = 6$, $b_V = 20$.

Acknowledgements

Vorrei ringraziare tutte le persone che sono state al mio fianco durante questo percorso.
Prima tra tutti la mia famiglia, che mi ha sempre sostenuto, incoraggiato ad andare avanti e inseguire i miei sogni. Grazie a mamma e papà per avermi sempre supportato e spinto ad intraprendere questo percorso. Grazie anche a Roberto, alla nonna e al nonno, che si è sempre interessato ai miei studi e alle mie passioni. Un grazie speciale a Daniele, che ha sempre sopportato tutte le mie crisi e lamentele, standomi vicino e dandomi sempre il suo sostegno.

Poi vorrei ringraziare anche tutte le altre persone, colleghi, amici e professori che hanno creduto in me e mi hanno permesso di arrivare dove sono. Ringrazio Laura e Matteo per tutti i progetti insieme e il supporto reciproco. E grazie anche a Sebastiano, a Daniele e ad Arianna, che sono sempre stati sempre disponibili a darmi una mano. Ringrazio il prof. Zanella, che mi ha aperto la strada verso la ricerca, il prof. Badia per aver sempre creduto in me, anche quando io stessa non ci credevo, e gli altri professori del DEI per essere sempre stati con me gentili e disponibili.

E soprattutto, ringrazio chi mi ha seguito, incoraggiato e aiutato in questo lavoro.
Grazie infinite a Francesco, che nonostante non fosse suo compito, mi ha aiutato tantissimo, ed è sempre stato disponibile a risolvere i miei problemi. Un grazie di cuore ad Umberto per essere stato il mio punto di riferimento, per avermi guidata passo passo, sostenendomi sempre, consigliandomi e aiutandomi nelle mie scelte oltre che in questo lavoro. E infine, grazie al prof. Milani per avermi sempre motivato e per avermi aperto la strada a tutte queste magnifiche opportunità di crescita personale e professionale.

Bibliography

[1] Albert-László Barabási and Márton Pósfai. *Network science*. Cambridge University Press, Cambridge, 2016.

[2] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019.

[3] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 4148, New York, NY, USA, 2009. Association for Computing Machinery.

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[5] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Fast point r-cnn. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9775–9784, 2019.

[6] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[7] Michal Drozdzal, Eugene Vorontsov, Gabriel Chartrand, Samuel Kadoury, and Chris Pal. The importance of skip connections in biomedical image segmentation. In *Deep learning and data labeling for medical applications*, pages 179–187. Springer, 2016.

[8] M. Everingham, L. Gool, Christopher K. I. Williams, J. Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88:303–338, 2009.

[9] Hehe Fan and Yi Yang. Pointrnn: Point recurrent neural network for moving point cloud processing. *arXiv preprint arXiv:1910.08287*, 2019.

[10] Biao Gao, Yancheng Pan, Chengkun Li, Sibo Geng, and Huijing Zhao. Are we hungry for 3d lidar data for semantic segmentation? *ArXiv abs/2006.04307*, 3:20, 2020.

[11] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, and Jose Garcia-Rodriguez. A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857*, 2017.

[12] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012.

[13] Timo Hackel, Nikolay Savinov, Lubor Ladicky, Jan D Wegner, Konrad Schindler, and Marc Pollefeys. Semantic3d. net: A new large-scale point cloud classification benchmark. *arXiv preprint arXiv:1704.03847*, 2017.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds, 2020.

[16] Shruti Jadon. A survey of loss functions for semantic segmentation. In *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pages 1–7. IEEE, 2020.

[17] J. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46:604–632, 1999.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[19] Haifeng Li, Jian Peng, Chao Tao, Jie Chen, and Min Deng. What do we learn by semantic scene understanding for remote sensing imagery in cnn framework? *arXiv preprint arXiv:1705.07077*, 2017.

[20] Ying Li, Lingfei Ma, Zilong Zhong, Fei Liu, Michael A. Chapman, Dongpu Cao, and Jonathan Li. Deep learning for lidar point clouds in autonomous driving: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 32(8):3412–3432, 2021.

[21] Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP:1–1, 07 2018.

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[23] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[24] Alfonso Medela and Artzai Picon. Constellation loss: Improving the efficiency of deep metric learning loss functions for optimal embedding. *arXiv preprint arXiv:1905.10675*, 2019.

[25] Umberto Michieli, Edoardo Borsato, Luca Rossi, and Pietro Zanuttigh. Gmnet: Graph matching network for large scale part semantic segmentation in the wild. In *European Conference on Computer Vision*, pages 397–414. Springer, 2020.

[26] Umberto Michieli and Pietro Zanuttigh. Continual semantic segmentation via repulsion-attraction of sparse and disentangled latent representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1114–1124, 2021.

[27] Shervin Minaee, Yuri Y Boykov, Fatih Porikli, Antonio J Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[28] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, page 849856, Cambridge, MA, USA, 2001. MIT Press.

[29] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[30] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas 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.

[31] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

[32] Xutong Ren, Lingxi Xie, Chen Wei, Siyuan Qiao, Chi Su, Jiaying Liu, Qi Tian, Elliot K Fishman, and Alan L Yuille. Generalized coarse-to-fine visual recognition with progressive training. *arXiv preprint arXiv:1811.12047*, 2018.

[33] Adian Fatchur Rochim, A. Muis, and R. Sari. A discrimination index based on jain's fairness index to differentiate researchers with identical h-index values. *Journal of Data and Information Science*, 5:5 – 18, 2020.

[34] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[35] Xavier Roynard, Jean-Emmanuel Deschaud, and François Goulette. Paris-lille-3d: A point cloud dataset for urban scene segmentation and classification. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2108–21083, 2018.

[36] Abraham. Savitzky and M. J. E. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 36(8):1627–1639, 1964.

[37] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2015.

[38] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[39] Otilia Stretcu, Emmanouil Antonios Platanios, Tom M Mitchell, and Barnabás Póczos. Coarse-to-fine curriculum learning. *arXiv preprint arXiv:2106.04072*, 2021.

[40] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[41] Greg Turk. The stanford bunny, 2000.

[42] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, 2001.

[43] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.