UNIVERSITY OF PADOVA

Department of Information Engineering

Master of Science in ICT for Internet and Multimedia – *Cybersystems*

Master's Thesis

Mixing Deep Networks and Entangled Forests

for the Semantic Segmentation of 3D Indoor Scenes

Supervisor
Dr. Stefano Ghidoni

Co-supervisor
Matteo Terreran, MSc

Candidate
Filippo Rigotto

Academic year 2018 - 2019

A very big thank you to all the people who made this possible.

We have seen that computer programming is an art, because it applies accumulated knowledge to the world, because it requires skill and ingenuity, and especially because it produces objects of beauty. A programmer who subconsciously views himself as an artist will enjoy what he does and will do it better.

Prof. Donald E. Knuth

It's nice to be elegant, but it's more important to be effective.

Prof. Michele Zorzi

Abstract

Object recognition and semantic segmentation are major topics in the computer vision field: while the former aims to find objects in a scene, with the coordinates of their bounding boxes being the typical output, the latter focuses on a more specific pixel-wise classification.

This kind of elaboration has been performed firstly on 2D images often made of three channels (RGB), with great performance and results. In recent years, thanks to several advances in hardware, in particular lower costs and a huge boost in processing power, accurate and reliable 3D data from sensors has been more steadily available.

This work focuses on semantic segmentation over 3D data such as *point clouds* representing indoor scenes, like offices, conference rooms and working spaces: after researching the current state of the art, deep neural networks based on the *PointNet* concept are evaluated, and one, the *Superpoint Graph* architecture, is selected for blending and mixing its features with the ones used by the *3D Entangled Forests* algorithm, a modification of the random forests learning architecture.

The exchange of features between the two architectures occurs in both directions, to have an insight on their relevance during the preparation and learning phases.

Contents

Chapter 1

Introduction

Recent years have seen the rise of Computer Vision (CV) as a key field of research. The advent of Augmented Reality (AR), Virtual Reality (VR) and Artificial Intelligence (AI) and the general availability of faster and more powerful PCs and Graphics Processing Units (GPUs) alongside cheap but accurate 3D sensors. The most significant is Microsoft's Kinect, a camera that allows for the acquisition of RGB images along with depth given by lasers and structured light systems. This device has pushed forward the boundaries in this field, initially focused on 2D images, and now 3D scene processing and understanding has become quite a relevant topic in both research and industry.

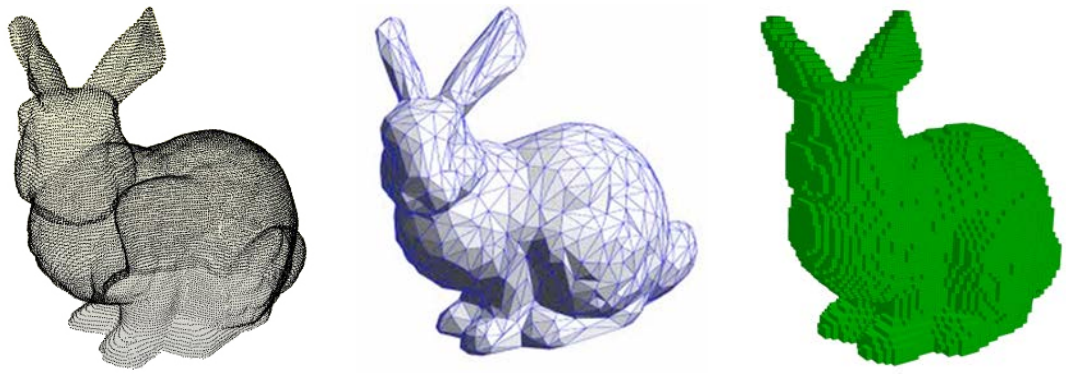
While activities like objects and shapes recognition and scene understanding are simple tasks for human beings, it is much difficult for a robot or for an AI to make autonomous decisions based on vision results. This has a particular relevance in indoor environments filled of objects of different shapes and volumes. In order for a robot to know its surrounding environment it has to be able to segment the input and to extract the semantic meaning of each section.

Semantic segmentation is a key sub-problem of scene understanding: it involves the decomposition of images into meaningful regions and structures. This is achieved by assigning a *label* from a predefined set to each and every pixel in the image, a task known as *pixel-wise classification*. Classification is a hard task because objects may be occluded by other things or their perception may be distorted by shadows or light reflection due to the materials they are made of. Because of this, objects are never well defined and lot of samples of them are needed to boost the accuracy of the algorithms. A sample image resulting from the semantic segmentation of an outdoor scenario, a populated street, can be found in Figure 1.1.

Figure 1.1: Semantic segmentation of a 2D image depicting an outdoor environment. Labels are represented by different colors superposed to the original image.
(source: sthalles.github.io/deep_segmentation_network)

1.1 Working with 3D data

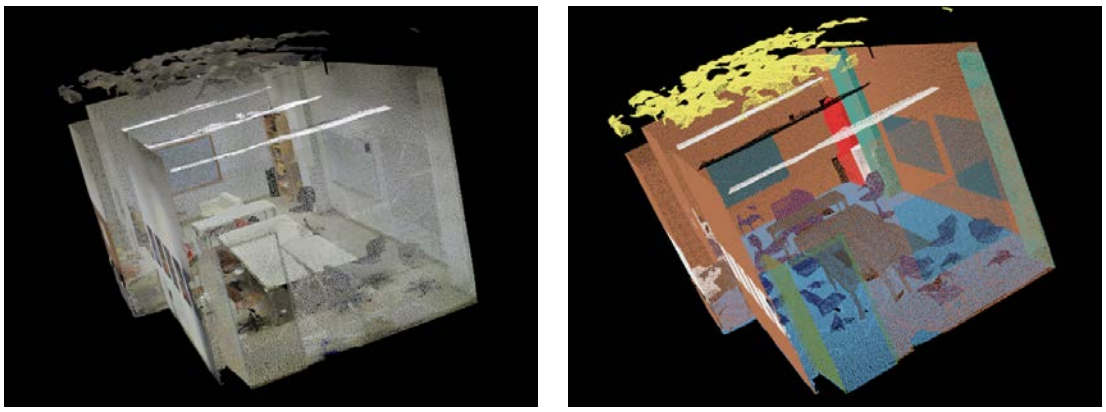
Considering our three-dimensional world, images become scenes and pixels become points. Thus a scene is made of multiple points, forming a *point cloud*. Examples of point clouds are depicted in Figure 1.2a and Figure 1.3a. Points are identified by their three coordinates with respect to a fixed *reference frame*, and in the vision field they are also typically characterized by additional properties such as the class label or the color expressed by a tuple of values according to a specific color space; for example, an RGB triplet, HSV values or a single intensity value for monochromatism. As clouds are in an Euclidean space a distance metric may be defined among points. Moreover, points in the cloud do not have an intrinsic order, contrary to image pixels that are structured in a 2D plane, or simply a 2D matrix, and for which a precedence order may be defined (in both row and column directions). Points also present invariance under certain transformations: for example, rotating the whole cloud together should not modify neither classification nor segmentation properties.



(a) Point cloud (b) Mesh version (c) Voxelized version

Figure 1.2: Several representations of the Stanford bunny.
(sources: pointcloudwarehouse.com, cse.osu.edu (CSE3541), ResearchGate)

Point clouds are a versatile tool to represent objects and surfaces but they are not the only structure for working with 3D data: a visual example, the historic Stanford bunny[1], is presented in several versions in Figure 1.2. *Meshes* are a more complete 3D representation that include faces defined between points. Another popular alternative is obtained through *voxelization*, a process that takes a 3D object and constructs a regular 3D shape surrounding it, usually made by simple polygons like cubes. Practically, voxelization is a uniform mapping of each points' coordinates (x, y, z) into discrete voxel coordinates (i, j, k). The mapping depends on the resolution of the voxel grid in the space, that can either be fixed or chosen so objects occupy a subvolume of predefined volume. Using a fixed resolution maintains the objects' relative scale, while using the second approach avoids the loss of shape that occurs when voxels are too small. The *projection* method consists of representing a 3D object with multiple 2D images ("views") taken from different orientations and stored alongside with *depth*, the distance between the camera and the 3D point location, for every pixel. Data is said to be in 2.5D, or RGB-D, in this case. The original object can be reconstructed from views by means of geometrical properties of the space, as performed by Structure from Motion (SfM) algorithms [1], [2].



(a) Original scene (b) Segmented scene

Figure 1.3: Semantic segmentation of a 3D indoor scene.

The task of semantic segmentation remains unchanged in this domain: the 3D scene is decomposed into meaningful regions and a label is assigned to each point of the cloud. An example of a processed indoor office is presented in Figure 1.3: left, the point cloud of the room with original colors evidenced; right, the same point cloud where colors represent different labels, that is, points colored in azure are part of the same *floor* class, brown is for points of the *wall* class, blue is for *chairs*, etc.

[1]More at the page http://graphics.stanford.edu/data/3Dscanrep/#bunny

1.2 Machine learning

Automated learning, or Machine Learning (ML), is the ability of programming computers so that they can "learn": in summary, it can be thought as the process of converting *experience* into *knowledge*. [3] While the given input represents experience, the output is a program that can generalize and perform some operations, the most common is the classification task. As a practical example of that, imagine an algorithm that can analyze e-mails and tell whether a specific one is spam or not, or another algorithm that predicts what single object is in an image (within a set of predefined choices). These algorithms are called *learners* or, for this particular task, *classifiers*. Classifiers initially starts with no knowledge and they learn by iterating multiple times over input data, for which they also know the right result labels (*supervised* learning). To have a good classifier the goal during learning is to minimize the classification error, quantified by a *loss function* that is an indicator of the difference between predicted and true labels, while keeping high the capability to generalize to new data, that is, to prevent *overfitting* on training data. Overfitting occurs when learning becomes too tailored to input data and the algorithm has bad performance on new data.

Several different kinds of Machine Learning (ML) algorithm exists: many tools may be used as automatic learners, and in this thesis two will be extensively used: deep networks and random (entangled) forests.

1.2.1 Deep neural networks

Artificial Neural Networks (NNs) are the most prominent trend in computer vision in the last decade, even if their foundations date way back in time. Networks of the simplest kind are made of a collection of interconnected *neurons* (or perceptrons, as pioneered by Rosenblatt in 1958. [4]) stacked in layers, and they are known as Multi-layer Perceptrons (MLPs). An example is in Figure 1.4. Each connection has a weight that is initially random and changes during learning through the *backpropagation* algorithm: after data has passed through the network, a gradient descent optimization approach exploits the chain rule of derivatives to compute the gradient of the loss function, which is then propagated back through layers to update nodes' weights. This phase is also denoted as *training* (or *fitting*) a NN. A trained network with its learned set of weights may be used to predict labels for new data, a phase called *inference*. During inference weights are frozen and backpropagation is not enforced, because the model is already fitted. A network is considered *deep* if it is made of more than three layers. MLPs are not the only type of network, and some architecture examples are presented in Chapter 4. Moreover, different network topologies are required for working with different 3D representations, such as the ones presented previously in this chapter.

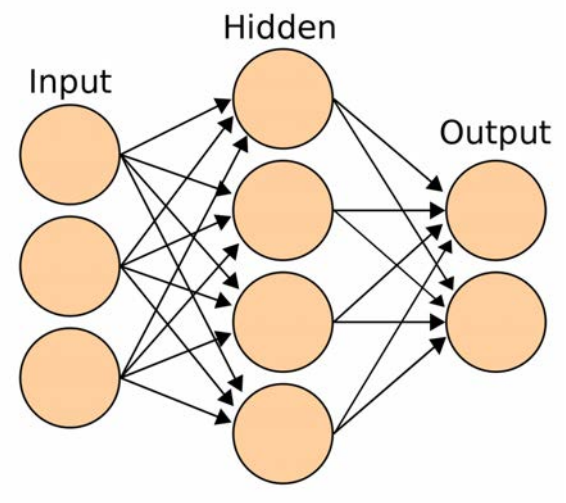


Figure 1.4: Scheme of a Multi-layer Perceptron (MLP), from [5, §11.2].

1.2.2 Random Forests

Random Forest (RF) classifiers are an *ensemble* learning method that work by constructing multiple *decision trees* on training data. Decision trees are a simple representation to classify sample data, but trees grown very deep tend to overfit and forests constitute a way to weight between trees trained on different parts of the dataset. The forest is trained using a modified *bagging* algorithm which consists on repeatedly fitting trees to a random sample of the training set. Predictions are then made by averaging single trees' predictions, this keeps low the variance of the model, or by taking the most suggested class (*majority* rule). A visual example is in Figure 1.5. RF classifiers will be further explored in Chapter 5.

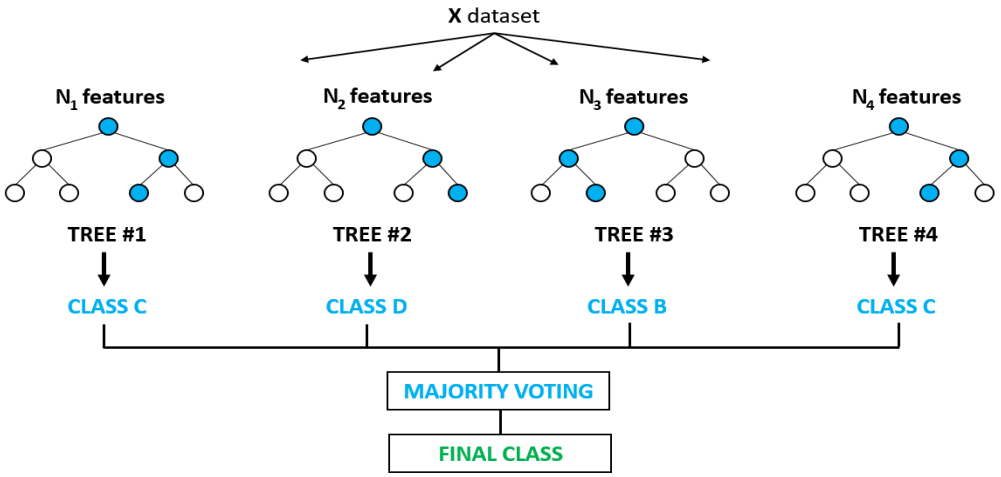


Figure 1.5: Scheme of a Random Forest (RF) classifier.
(source: globalsoftwaresupport.com/random-forest-classifier-bagging-machine-learning)

1.3 Thesis objectives and structure

This thesis poses as objectives to study and evaluate the current state of the art in semantic segmentation applied to indoor scenes, in the form of plain 3D point clouds, and to evaluate and compare both deep networks and Entangled Forest (EF) classifiers. Then, this thesis contributes by studying the feasibility of exchanging features between one of the selected deep networks and Entangled Forests.

This work has been developed during an internship at the Intelligent and Autonomous Systems Lab[2] at the University of Padova under the supervision of Dr. Stefano Ghidoni, assistant professor, and Matteo Terreran, Ph.D. student.

The thesis is structured as follows. In Chapter 2, the current state of the art and related works in semantic segmentation is reviewed. In Chapter 3 the Stanford dataset considered for the thesis is described and analyzed. In Chapter 4 three most prominent architectures based on deep learning are thoroughly examined and compared. In Chapter 5 the Entangled Forests algorithm is investigated. For both these two last chapters, configurations, parameters, issues encountered and proposed solutions are laid out. Chapter 7 discloses attempts and achieved results in exchanging features between the two realities. Finally, concluding remarks and possible future developments are presented in Chapter 8.

[2]IAS-Lab, Via Ognissanti, 72, 35131 Padova, http://robotics.dei.unipd.it

Chapter 2

State of the Art

The topic of semantic segmentation has been reviewed in several surveys in the literature. While most of them focus only on 2D images [6]–[9] some recent works conduct a more extensive review also including the 3D space and point clouds. Weinmann *et al.* [10] present a fully automated and versatile framework composed of neighborhood selection, feature extraction and choice, and classification; for each of them they consider and list a variety of common approaches and test applicability in terms of simplicity, efficiency and reproducibility. Garcia-Garcia *et al.* [11] collect and describe the most used datasets and challenges, and then go on reviewing existing deep learning-based methods and networks, highlighting contributions and relevance.

The first attempts at working on point clouds through deep learning techniques used a *volumetric* representation, that is, they employ voxelized input data. An example of this is the work of Maturana and Scherer [12]. They perform object detection through a 3D Convolutional Neural Network (CNN) that is an extension of the 2D version where convolution is performed with three-dimensional kernels over the voxelized input.

Because of the computational expensiveness of performing 3D convolutions, Su *et al.* [13] managed to build 3D shape classifiers from sets of 2D image renderings and then performed 2D convolutions. Their pipeline is then to obtain several image views of the scene, to apply a CNN on these and then to report and merge back results in the 3D space. They obtained better results because of the efficiency of 2D representations w.r.t. 3D voxelization, advances in image descriptors and massive databases such as ImageNet [14], all true at the time they wrote the paper. A similar approach of using multiple 2D views is presented by Boulch *et al.* [15].

At the same time manual feature extraction was still investigated by Hackel *et al.* [16]: they studied efficient ways of dealing with huge point clouds such as the ones coming from Light Detection and Ranging (LiDAR) scans, based on the central concept of points' neighborhood so to account for spatial correlation between classes. Multi-scale neighborhoods in 3D point clouds were investigated by Thomas *et al.* [17]:

based on spherical neighborhood and proportional subsampling, this allows feature computation with a consistent geometrical meaning, such that they can be used for example by RF classifiers. The latter work is based on other notable results in 3D descriptors, pooling and local surface features [18]–[21].

After having delved into CNN based on both volumetric and multi-view representations [22], a pioneering work by Qi *et al.* attempted to design a neural network that directly consumes point clouds: PointNet [23]. Such neural network was created arguing that traditional convolutional networks require highly regular input like image grids, 3D voxels or meshes; generally raw point clouds are not in a regular format. Directly working on points is more straightforward than recurring to voxelization. PointNet is a unified architecture that outputs class labels for the entire input (classification) or per-point segment labels (semantic segmentation). The network learns to summarize an input point cloud by a sparse set of key points.

A key limitation of PointNet in its first formulation is that it does not capture local structures, limiting the ability to recognize fine patterns. The solution proposed by the same authors, named PointNet++ [24], is to introduce a hierarchical network that applies the base PointNet recursively on an overlapping partition of the point cloud, thus learning local features at progressively increasing scales much like what does the SIFT algorithm [25] for 2D images. More insights on this network will be covered in Section 4.1.

Engelmann *et al.* [26] further recognize that the neighborhood context used by PointNet is not sufficient: spatial context is in general very important for semantic segmentation, and thus they introduce two mechanisms to account for that. While the first is similar to PointNet++ in incorporating neighborhood information by processing input data at different scales or by considering multiple adjacent regions, the second approach operates on the estimated point descriptors and consolidates them by exchanging information over a larger spatial neighborhood. Consolidation is a second stage in the network architecture and is done by employing MLPs and max-pooling or Gated Recurrent Units (GRUs).

Wang *et al.* [27] developed a new layer operation, namely EdgeConv, that incorporates local neighborhood information, can be stacked or recurrently applied to learn global shape properties and captures semantic characteristics. The need for such operation rises from previous considerations on PointNet: it operates on single points and extensions allow the network to consider points' neighborhood to exploit local features; anyway, treating points independently at a local scale neglects fundamental geometric relationships and this results in underrepresenting features. EdgeConv addresses this while maintaining permutation invariance, a foundation of PointNet. Long-distance context could also be exploited by means of a two-step hierarchical Recurrent Neural Network (RNN), as recently proved by Ye *et al.* [28]. The input point cloud is subdivided into partially overlapping blocks along the two horizontal directions, on which two sets of RNNs are consequently applied so to account for long

spatial dependencies. Experimental results by the authors show that this recurrent approach largely improves accuracy.

SEGCloud is a more complex framework presented by Tchapmi *et al.* [29], that is made of several parts all centered on the use of a CNN. The 3D CNN operates on voxelized regular 3D point clouds and outputs predictions at the voxel level, that is, all points in a voxel are assigned the same label. Fine-grained labeling (at the point level) is obtained through Trilinear Interpolation (TI), for the upsampling of class scores from voxel neighbors to points, and a final Conditional Random Field (CRF) that ensures spatial consistency and smoothness between labels. Additionally, according to [30], the iterative nature of operation of CRFs is the same of RNNs, and they can be freely exchanged.

The use of Self-Organizing Maps (SOMs) [31] to model spatial distribution of points in clouds is investigated by Li *et al.* [32]: hierarchical feature extraction is performed on individual points and SOM nodes, and the input point cloud is represented by a single feature vector. This approach is successfully used to perform part classification and segmentation on datasets made for object recognition such as ModelNet [33].

Landrieu and Simonovsky [34] developed a novel approach based on the Superpoint Graph (SPG) to tackle the challenge of dealing with point clouds made of million of points. Clouds are represented as a collection, a graph, of interconnected simple shapes, dubbed superpoints, on which classification and segmentation is performed using smaller PointNets for superpoint embedding and graph convolutions for semantic segmentation, through modified versions of Edge-Conditioned Convolutions (ECCs) and GRUs. Superpoint graphs encode contextual relationships between object parts in 3D point clouds, and can consider both fine details and long-range context information simultaneously. A deeper tractation of this network will be developed in Section 4.2. A similar approach based on supervoxel contexts and graph-based optimization applied to instance segmentation of trees in urban areas was presented shortly after by Xu *et al.* [35].

Convolutional networks applied to features extracted from input data by a MLP is the backbone of PointCNN [36]: the fully connected network is used to simultaneously weight and permute input features ideally keeping order invariance, while hierarchical convolutions perform the classification and segmentation tasks.

At the present time, an attempt at jointly considering both semantic and instance segmentation is carried on time by Pham *et al.* [37]. Pointwise networks predict the semantic classes of 3D points and embed them into high-dimensional vectors so that points of the same object instance are represented by similar embeddings. A multi-value CRF is used to incorporate semantic and instance labels in a joint optimization problem. The pointwise network part will be further expanded in Section 4.3.

Aside of the inquired neural networks, still today many works focus on non-deep algorithms to perform semantic segmentation on 3D point clouds. Their major strength is not to rely on GPUs, so they are more suitable for mobile robotics

applications where computation power and battery life have non-neglibigle limits. A notable mention is RF classifiers [38]. Their use for the task of semantic segmentation is sparse in the literature: Montillo *et al.* [39] worked on 3D tomography and first proposed entangling the binary tests applied to each tree node of the forest, such that the test result depend both on results of tests applied earlier in the same tree and on the input voxel to be classified.

Wolf *et al.* previously worked on RF classifiers [40] applied to 3D indoor scenes. Their framework involved over-segmenting the input point cloud, calculating discriminative feature vectors for each segment and conditional label probabilities using a RF, to then initialize a dense CRF that account for the likelihood of different class labels appearing close to each other. The same authors subsequently applied the entanglement concept to 3D point clouds introducing the 3D Entangled Forest (3DEF) [41], an extension of standard RF with the use of 3D entangled features, which capture frequently appearing combinations of classes and learn their 3D geometric configuration in the scene. Relations are learned between different classes and also within the same class, to have a smooth classification output without resorting to CRFs. Antonello *et al.* [42] studied an offline method to directly apply 3DEF classifiers to whole reconstructions, along with two approaches based on a multi-view frame fusion algorithm to improve semantic segmentation of single frames and to build a semantic map. 3DEF classifiers will be further developed in Chapter 5.

Chapter 3

The Stanford dataset

Several datasets that focus on semantic segmentation exist: some focus on 2D data (KITTI [43], Cityscapes [44], ScanNet [45]), some add depth information (NYU-D [46], Sun RGB-D [47], SceneNN [48]), others also comprise 3D point clouds (Stanford [49], Sun3D [50], Matterport3D [51]). Note that these lists are not exhaustive. Among all these datasets, the "Stanford" 2D-3D-S[1], is a dataset of large-scale indoor spaces collected by Armeni *et al.* for their work on joint 2D-3D scene understanding [49].

As the name suggests, this dataset contains both 2D RGB images and 3D point clouds with RGB data, plus depth and semantic annotations for 13 different classes, obtained using a Matterport[2] scanner. The 2D/2.5D part is made of 70000 RGB images, for which the scanner also provided the corresponding depths, surface normals and camera information. Semantic annotations and global XYZ images (in both regular and 360° equirectangular[3] formats) are elaborated in a post-acquisition phase. The 3D part includes point clouds along with RGB values and the semantic label for each point. The Stanford Large-scale 3D Indoor Spaces Dataset (S3DIS) dataset is a subset of the full 2D-3D-S dataset that contains only the 3D part, and it was used by the same authors for a previous work on 3D semantic parsing of indoor spaces [52].

The dataset is made of mainly educational and office rooms grouped into 6 areas originated from 3 different buildings, with a covered area of over 6000m^3 and over 215 million total points. For each room a colored point cloud and 3D semantic annotations for objects are provided. Detected objects are both structural elements (ceiling, floor, wall, beam, column, window and door) and office furniture (table, chair, sofa, board and bookcase). Rooms are heterogeneous and they contain different subsets of the objects, e.g. there is generally no sofa in the offices, but there are some

[1] http://buildingparser.stanford.edu/dataset.html

[2] https://matterport.com

[3] A projection of the 3D space into a 2D image, typically a panorama derived from 360° cameras.

in the open space, and equally the bookcase is likely to be in conference rooms but not in storage rooms.

As a last important note, this is an unbalanced dataset: it becomes clear from Table 3.2 where it can be noticed how bigger elements in rooms, such as walls, are made of much more points than for example sofas or boards. This is a common problem also in the literature and is a challenge for automatic learning.

Tables 3.1 and 3.2, reprinted from the website, show the composition of the dataset: for each area, it is reported the number and type of rooms, and the number of points per class. Figure 3.1 shows sample close-ups of 3D clouds and semantics data for each area.

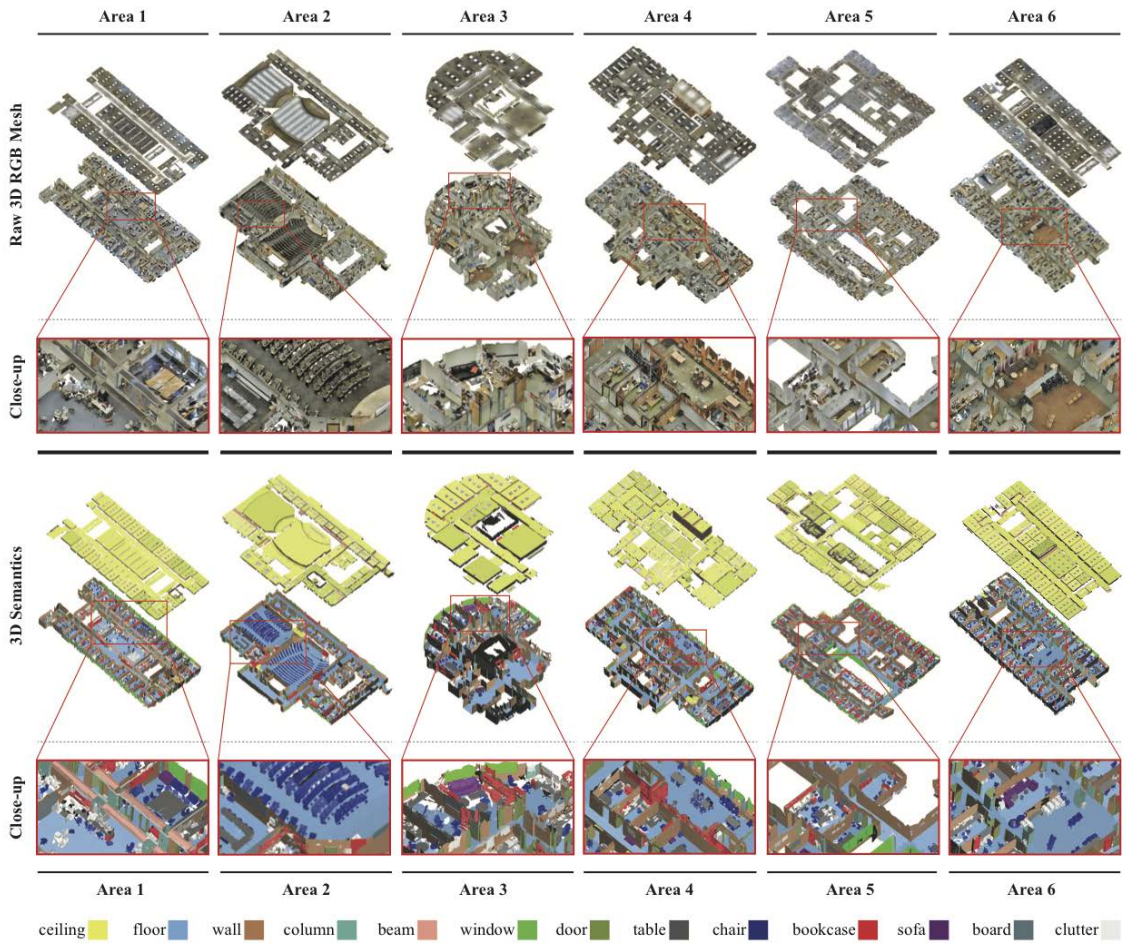


Figure 3.1: 3D part of the dataset (S3DIS). Above, 3D point clouds with RGB data. Below, semantic annotations represented by different colors for each class.
(source: http://buildingparser.stanford.edu/dataset.html)

Table 3.1: Composition of S3DIS dataset. Number of rooms per area.

	Area (m^2)	Vol. (m^3)	Office	Conf. Room	Auditorium	Lobby	Lounge	Hallway	Copy Room	Pantry	Open Space	Storage	WC	Total
Area 1	965	2850	31	2	–	–	–	8	1	1	–	–	1	45
Area 2	1100	3065	14	1	2	–	–	12	–	–	–	9	2	39
Area 3	450	1215	10	1	–	–	2	6	–	–	–	2	2	24
Area 4	870	2780	22	3	–	2	–	14	–	–	–	4	2	49
Area 5	1700	5370	42	3	–	1	–	15	–	1	–	4	2	55
Area 6	935	2670	37	1	–	–	1	6	1	1	1	–	–	53
Total	6020	17360	156	11	2	3	3	61	2	3	1	19	9	270

Table 3.2: Composition of S3DIS dataset. Number of 3D points per class, for each area.

	Ceiling	Floor	Wall	Beam	Column	Door	Window	Table	Chair	Sofa	Bookcase	Board	Total
Area 1	56	45	235	62	58	87	30	70	156	7	91	28	925
Area 2	82	51	284	62	58	94	9	47	546	7	49	18	1307
Area 3	38	24	160	14	13	38	9	31	68	10	42	13	460
Area 4	74	51	281	4	39	108	41	80	160	15	99	11	963
Area 5	77	69	344	4	75	128	53	155	259	12	218	43	1437
Area 6	64	50	248	69	55	94	32	78	180	10	91	30	1001
Total	391	290	1552	165	260	549	174	461	1369	61	590	143	6093

Chapter 4

Deep learning on 3D data

In this chapter, the focus moves towards deep learning on 3D data: a selection of deep networks from the state of the art (Chapter 2) that works on 3D point clouds are now thoroughly exposed and investigated.

4.1 PointNet++

As already mentioned in Chapter 2, PointNet is a deep network architecture that works directly on 3D point clouds and outputs class labels for the entire input (classification) or per-point segment labels (semantic segmentation).

In its basic settings the network considers only points' coordinates (x, y, z) and it learns to summarize an input point cloud by a sparse set of key points: in particular, it learns a set of optimization functions (or criteria) that select interesting or informative points in the cloud, and encodes the reason for their selection.

Nonetheless, a point cloud is just a set of points and thus it is natively invariant to point permutations: this property is preserved using max-pooling. Pooling is a simple symmetric function empirically proven to be the best candidate by the authors, that also tested other alternatives such as to sort input into a canonical order or treat the input as a sequence to train a RNN. Falling back to sorting procedures does not work because in higher dimensional spaces there is no ordering that is stable w.r.t. point perturbations, see [53], and RNNs add unwanted and unmanageable complexity to the architecture, while not being sure that the network will become invariant to input order after having been trained with randomly permuted sequences.

The network is robust to small perturbations or corruptions through point insertion and deletion, and to affine transformations on the input set. In fact, a data-dependent spatial transformation that attempts to canonicalize data before PointNet process them is shown to improve the results.

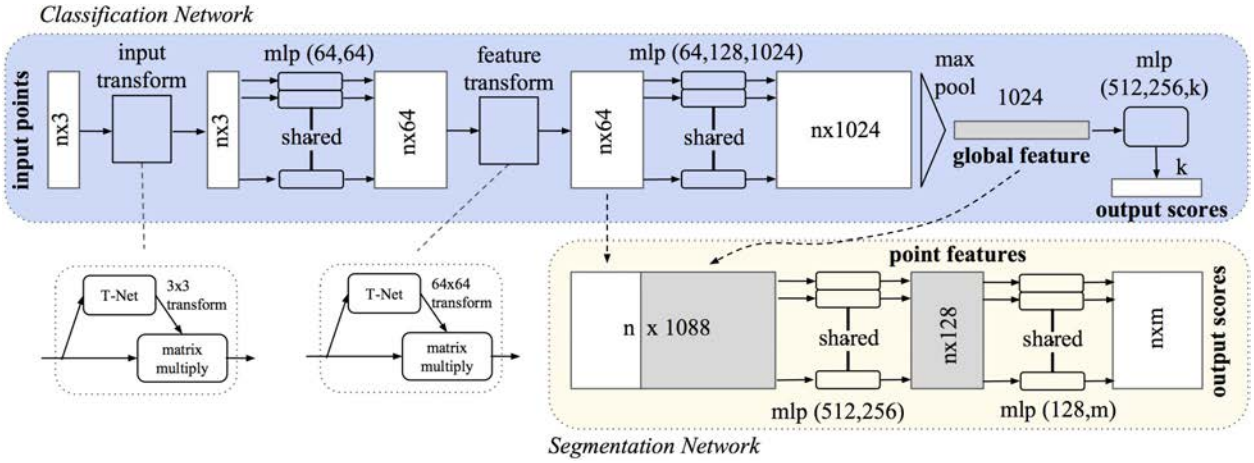


Figure 4.1: PointNet architecture [23]. n is the number of input points, k and m are the number of categories for classification and segmentation, respectively. In brackets, layer sizes.
(source: http://stanford.edu/~rqi/pointnet)

The network is presented in Figure 4.1: the initial stages are the same for both classification and segmentation operations. The architecture is made of a stack of MLP layers, interspersed with with max-pooling layers and smaller transformation networks (*T-net* in the figure).

T-net predicts an affine transformation that is directly applied to its input via matrix multiplication, to perform point alignment to a canonical space. T-nets are applied both to input space and feature space. However, the network that operates on the feature space has much higher dimension than the spatial transform matrix, which greatly increases the difficulty of optimization. The solution is to add a regularization term to the training loss: the transformation matrix is constrained to be close to an orthogonal matrix, so it will not loose information. The regularization term stabilizes optimization and enhances performance. Batch normalization [54] and ReLU are used for all layers, dropout [55] is used for the last MLP in the classification part.

The final fully connected layers aggregate learnt optimal values (i) into a global descriptor for the entire shape or (ii) for predicting per-point labels. In the first case, the network outputs k scores for all the k candidate classes, while in the latter the model outputs $n \times m$ scores for each of the n points and each of the m semantic categories.

Practically, the output of the classification network is a vector that constitutes a global signature of the input set, but point segmentation requires a combination of both local geometry and global knowledge (semantics), achieved concatenating the global vector with per-point features and extracting new features based on this combination.

A key limitation of PointNet is that it does not capture local structures induced by the space points live in, limiting the ability to recognize fine patterns or to generalize to complex scene: PointNet++ [24] is the solution proposed by the same authors.

The input set of points is partitioned into overlapping local regions according to the distance metric of the underlying 3D space. Local features capturing fine geometric structures are extracted from small neighborhoods and then further grouped into larger units, again processed to obtain higher level features. The procedure is repeated until features of the whole cloud are obtained: PointNet++ leverages neighborhoods at multiple scales to achieve robustness and capture details; random input dropout during training helps adaptively learning weights and combine multi-scale features.

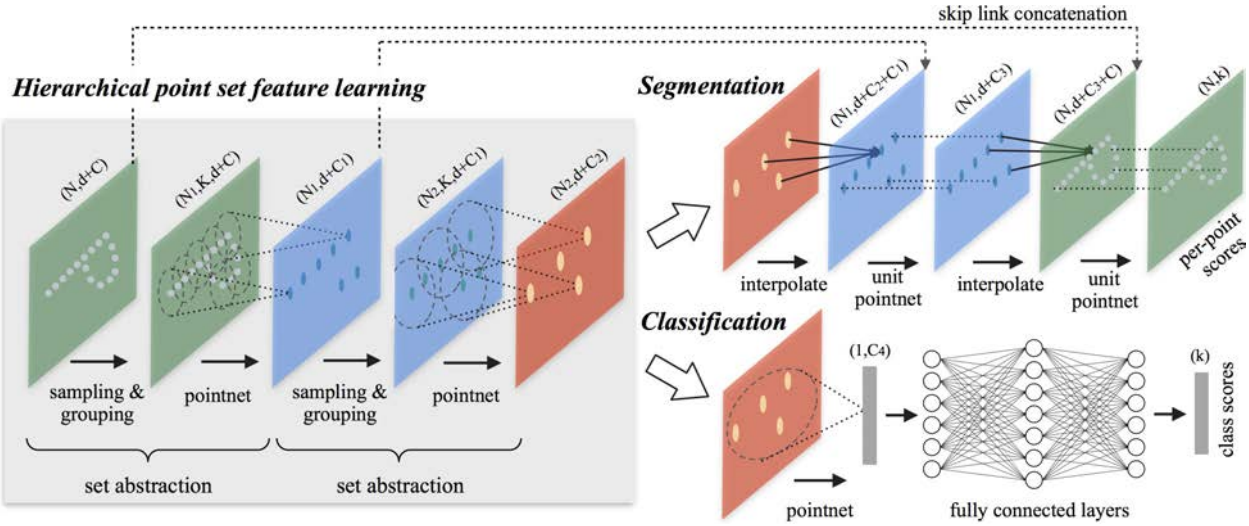


Figure 4.2: PointNet++ hierarchical architecture [24].
Points in 2D Euclidean space are used as an example.
(source: http://stanford.edu/~rqi/pointnet2)

PointNet++ structure is depicted in Figure 4.2: it is composed of many *set abstraction* levels that process a set of points and produce a subset ready for next level. Each set abstraction is made of three layers: the *sampling* layer selects a set of points as centroids of local regions, the *grouping* layer constructs local region sets based on centroids neighbors, and PointNet encodes region patterns into feature vectors. Sampling is performed using the iterative farthest point sampling (FPS) algorithm, that has better coverage than simple random choices. Grouping is performed using the *ball query* algorithm that finds all points within a given radius from the centroids or alternatively using the K-nearest neighbors search that finds a fixed number of neighbors.

Each partition is then defined as a neighborhood ball in the 3D space, parameterized by the centroid location and the scale, a common structure needed to share weights in PointNet layers. The authors acknowledge that the input point set may have variable density at different areas, a commonly verified assumption, and thus taking a small neighborhood may result in having too few points that prevent PointNet

to capture robust patterns: in this case a larger scale should be used to look to a sufficient number of points.

In PointNet++ each abstraction level extracts *multiple* scales of local patterns and combine them according to local densities: PointNet layers are density-adaptive. Two types of layer are proposed by the authors, also represented in Figure 4.3:

- *Multi-scale Grouping (MSG):* apply grouping layers and PointNets at different scales and concatenate all the features in a single feature vector. Random input data dropout optimizes learning by different sparsity and varying uniformity. It is computationally expensive since PointNets run for every centroid, which are many at lower levels.
- *Multi-resolution Grouping (MRG):* the feature vector of a region at some level is a concatenation of two vectors, one summarizes features at each subregion of the lower level, the other is obtained by directly processing all raw points in the region using a single PointNet. The importance of each of the two subvectors depends on the density of the local region, so they must be weighted differently for each case. It is computationally more efficient since feature extraction in large scale at the lowest levels is avoided.

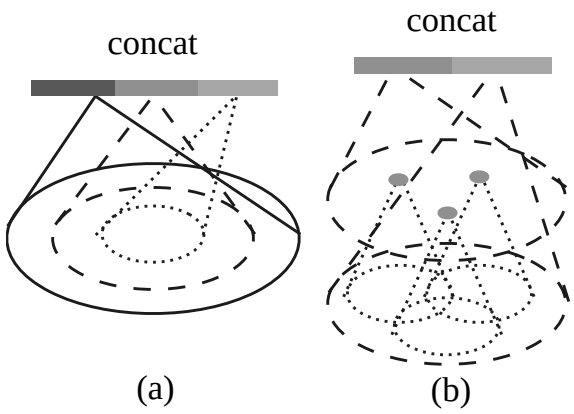


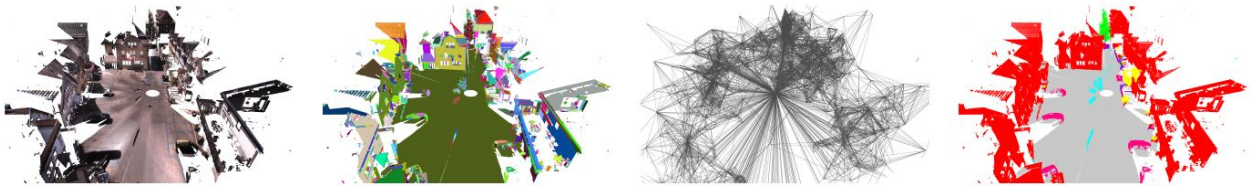
Figure 4.3: Density-adaptive grouping layers in PointNet++ [24].
(a) Multi-scale grouping (MSG); (b) Multi-resolution grouping (MRG).

In each set abstraction layer, the input point set is subsampled. But in the semantic segmentation task labels are needed for each point: distance-based interpolation across levels is employed to propagate labels to all the original input data points.

4.2 Superpoint Graphs

Superpoint Graphs are a novel point cloud representation introduced by Landrieu and Simonovsky in 2018 [34] and resumed in [56]. They propose to represent large 3D point clouds as an interconnection of simple shapes named *superpoints*. This structure can be captured by an attributed directed graph called the *superpoint graph*

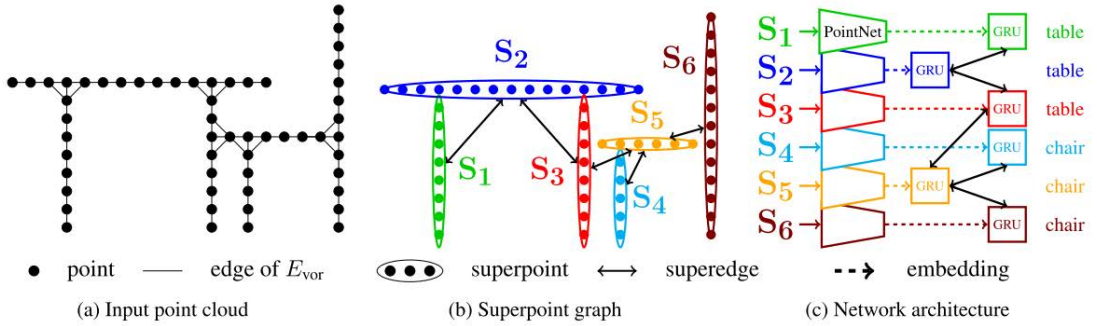
(SPG). The representation is able to consider entire object parts as a whole, and to describe in detail relationships between adjacent objects.



(a) RGB point cloud (b) Geometric partition (c) Superpoint graph (d) Semantic segmentation

Figure 4.4: Pipeline for the Superpoint Graphs architecture [34].

The authors' starting point are LiDAR scans: they are made of hundreds of millions of points, and the problem is intractable using direct deep learning approaches like PointNet. The superpoint representation allows to split the problem into three distinct subproblems, also visually presented in Figure 4.4 and in Figure 4.5.



(a) Input point cloud (b) Superpoint graph (c) Network architecture

Figure 4.5: The SPG framework on a toy example with a table and a chair [34].

First, the point cloud is partitioned into geometrically simple but meaningful shapes: the objective is not to retrieve individual objects but rather to break down objects into simple parts, clusters. Clusters are geometrically simple and it can be assumed that they are semantically homogeneous, that is, they do not cover objects of different classes. This step is unsupervised as are clustering procedures in general, and makes no use of labels. Moreover, the segmentation adapts to local geometry: clusters can be both large simple shapes like walls and smaller components such as parts of a chair. For each point of the cloud a set of geometric features is computed. In the implementation, they characterize the shape of the points' 10-nearest neighborhood: *linearity, planarity, scattering, verticality*, as defined in [57], [58] and *elevation*, the z coordinate of the point, normalized over the whole cloud. The partition is constituted by connected components, the superpoints, that solve the *generalized minimal partition problem*: it is an optimization problem of a non-convex and non-continuous functional, and thus the problem cannot be solved exactly for a large point cloud, however the ℓ_0-cut pursuit algorithm [59], [60] developed by one of the authors can provide an approximate solution in few iterations.

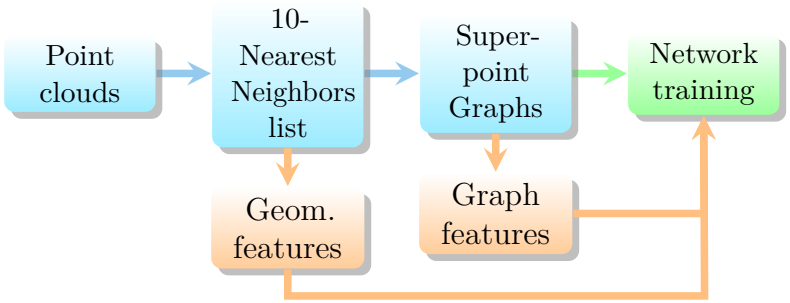


Figure 4.6: Preprocessing pipeline for the SPG architecture.

The constant connected components define the superpoints from which the superpoint graph (SPG) can be easily derived: it is an oriented attributed graph whose nodes are the set of superpoints and edges, referred as *superedges*, represent the adjacency between them. Moreover, superedges are annotated with 13 features that comprise mean, standard deviation and centroid offsets, along with length, surface, volume and point count ratios, that can also be computed from the eigenvalues of the covariance matrix of points' positions in each superpoint, sorted by decreasing value.

Every superpoint is downsampled to a few hundred of points, on which a small PointNet can easily operate to obtain a compact descriptor, a fixed-size vector named *embedding*. Descriptors are computed in isolation: contextual information is injected in a later stage.

The SPG is by orders of magnitude smaller than any graph built on the original point cloud: as a last step, deep learning algorithms can be used to classify each superpoint based on its embedding. The most suited class of architectures for this task is graph convolutions, specifically ECC [61], developed by one of the authors: ECC can dynamically generate filtering weights using a MLP that build up a weight matrix for each edge. The general idea is that superpoints refine their embeddings by using information passed along superedges: in the implementation, each superpoint maintains its state in a GRU, initialized with the embedding from PointNet. GRUs are modified to ignore the external context if the class state is almost certain and to learn to focus on specific feature channels. They operate in a way similar to CRFs for 2D image segmentation, but the domain is richer and less constrained. GRUs provide the final labels for the semantic segmentation task.

For more detailed information, mathematical insights and references about each step in the framework and in the architecture, please refer to the original paper [34].

4.3 JSIS3D

Pham *et al.* [37] try to jointly address the problems of semantic and instance segmentation of 3D point clouds: as already mentioned in the introduction, while semantic segmentation aims to classify each point using a set of predefined classes,

instance segmentation is a further step that also discriminates between different objects: for example, two chairs will have the same semantic label (the one that identifies *chair*, of course) but two different instance labels.

In their work, the authors acknowledged that most of the times object categories and object instances are mutually dependent and shape and appearance features extracted on an instance may help to also identify the object category, and they developed a deep Multi-task Pointwise Network (MT-PNet) for predicting semantic classes and embedding points into high-dimensional vectors such that points of the same object instance end up having similar representations and thus can be clustered, and also proposed a multi-value CRF to incorporate and jointly optimize both semantic and instance labels in the field model into a unified framework.

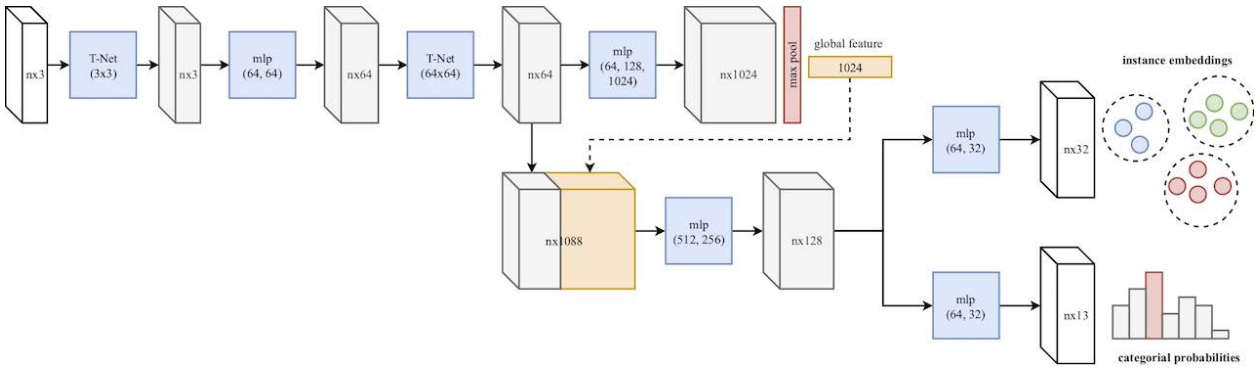


Figure 4.7: The MT-PNet network structure [37].

The network, named MT-PNet, is depicted in Figure 4.7. It accepts as input overlapping 3D windows scanned from the entire cloud and produces labels for all vertices within the windows. The network structure highly resembles PointNet architecture but in the end it diverges in two branches that predict semantic labels and generate pointwise instance embeddings for 3D points. The loss used for the network is the sum of the prediction loss, that is, the usual cross-entropy, and of the embedding loss, which is defined as $L_e = \alpha \cdot L_{pull} + \beta \cdot L_{push} + \gamma \cdot L_{reg}$ where intuitively L_{pull} attracts embeddings towards their centroids, L_{push} keeps centroids away from each others, and L_{reg} is a regularization term. Constants are empirically set.

At this stage, points in the cloud are represented by their 3D location, their normal vector, color and the resulting embedding from MT-PNet. Semantic and instance labels are then represented as random variables taking values from their respective sets of possible choices. While the set of semantic labels is known, the set of instance labels needs to be determined. The multi-value CRF works on a graph defined over the point cloud and its set of labels, and simultaneously optimizes both labels' random variables according to an *energy* function that includes many potentials linked to both physical and semantic constraints. This latter part is optional and only serves to smooth the network's results: for full mathematical details, please refer to the original paper [37].

Chapter 5

Entangled forests

As briefly mentioned in the introduction, Chapter 1, Random Forest (RF) classifiers are made of a collection of Decision Trees (DTs) (recall Figure 1.5). A DT is a structure made of nodes that represent a test on a data attribute. For each node the test's outcome generates two branches that lead to two other nodes. The structure then evolves from the root node down to the final *leaves* as a binary tree.

5.1 RF Training and inference

To separate differently labeled data, tests have to be performed. Every node in the trees is associated with a *split function*: these functions are learned during the forest's training phase. Trees start with only the root node and they grow by subsequently splitting data until one of the following conditions occur:

- the class label distribution on data that came to a node only contain points with the same label;
- a maximum tree depth level is reached;
- the size of the subset of the data that landed on a specific node is below a given threshold;
- the difference from the best achievable split is negligible.

Training stops when all child nodes have become leaves and a further split is not possible anymore. Trees in the forest can be trained in parallel. Leaves store a probability distribution given by the subset of training data that came to them through decisions.

The right split function for each node is selected from a randomly drawn pool of functions from a predefined set: the final candidate is the function that maximized the quality of the split, according to an evaluation metric. The most common metric

is the entropy of the class distribution: the best split is the one that has the lowest entropy among the ones produced by the candidate functions.

As for neural networks, a trained RF classifier may be used to predict classes for new data points: they go through the now-defined trees' structure and the final class probabilities are given by averaging each tree's output distribution.

5.2 Strengths and weaknesses

RF achieve high accuracy and generally do not require too much data preprocessing. The forest structure is highly parametrical and flexible: while after training the trees' structure is fixed, some parameters have to be set beforehand, like the number of trees in the forest and the maximum allowed depth for each tree. A tradeoff is needed: computation time scales linearly with the number of trees (RFs are much harder and time-consuming to construct than DTs), but performance also saturates.

Single trees tend to overfit the data, and RF try to solve this problem. While averaging or combining results in the forest mitigates the problem, two further actions may be evaluated: setting the maximum tree's depth to prevent going too much deeper, and employing the *bagging* strategy, that is, to train each tree on a smaller randomly-drawn subset (a *bag*) of the whole training data.

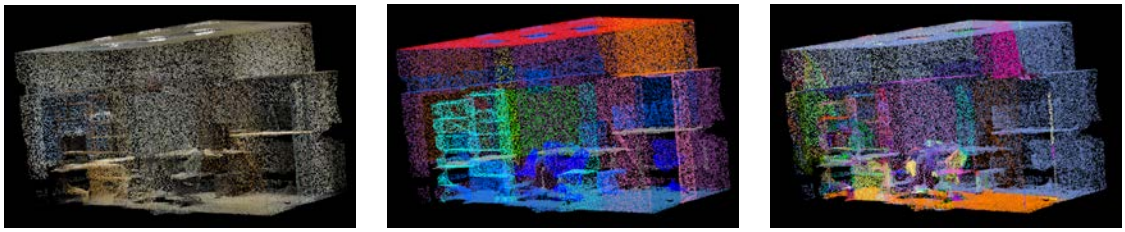
Another drawback of RF is that they process input points independently, such that each data point is related to a feature vector. Nonetheless, contextual information that derives from considering also the point's neighborhood may be relevant to help points' classification for the semantic segmentation task. Wolf *et al.* [41] proposed a modification of the standard architecture of RF to incorporate spatial information: they introduced *entangled* features in addition to standard *unary* features associated to single points, and named the new architecture 3D Entangled Forest (3DEF).

5.3 3D Entangled Forests

Figure 5.1: Preprocessing pipeline for the 3DEF classifier.

The algorithm accepts point clouds as input, which in the case of S3DIS represent single rooms. Voxel Cloud Connectivity Segmentation (VCCS) [62] is responsible to operate an *unsupervised* oversegmentation of the cloud into regions of perceptually similar 3D points, named *supervoxels*. By construction supervoxels obey to the spatial geometry of the scene and do not violate object edges. Next, a region growing algorithm is applied to recursively merge pairs of adjacent supervoxels into larger segments if their relative distance is below a fixed threshold. The distance function

considers a weighted combination of color in CIELAB space, surface normal and point-to-plane distances between clusters. This procedure leads to a set of segments, or clusters, input for the classifier. An example is in Figure 5.2. Feature vectors are computed for each segment, and on that basis the forest is trained. The trained forest is then refined with a rebalance of trees' leaves, a step that allows to achieve better performance.



(a) RGB scene (b) True labels (c) Clusters

Figure 5.2: Example of clustering for a conference room: (a) the original scene; (b) true labels from the dataset; (c) output of VCCS segmentation and subsequent merging procedure. Each label and cluster is represented with a different color.

5.3.1 Unary features

Point clouds and segments are aligned with the ground (floor) plane, an information that is both passed as parameter (the floor label) and acquired in the previous clustering step (the floor points' location). Then for each segment a 18-dimensional *unary* feature vector is computed, using the set of points that belongs to each segment. Considered features include non-complex and fast-to-compute appearance and geometric properties such as:

- mean and standard deviation of the points' color, for each channel in the CIELAB color space;
- mean and standard deviation of the angle between the surface normal and the ground plane;
- height above ground for the highest and lowest point in the segment;
- width, height and depth of the bounding box surrounding the segment;
- horizontal and vertical elongations from ratios between the previous dimensions;
- "thickness" of the segment;
- occupied area in the horizontal and vertical plane.

During training, for each feature the only parameter to be sampled is a threshold. During inference, if a segment's value for the feature is larger than this threshold, the feature evaluates to true.

5.3.2 Entangled features

Entangled features come from information extracted from closer nodes during the forest's training: when learning deeper levels of each tree, storing class distributions even for intermediate nodes helps splitting data and further growing the trees in the structure. The crucial point is to have a method to find across different scenes other data points whose contextual information help prediction for current data point. Each 3D entangled feature learns a set of geometric constraints relative to currently classified segment. Given two segments s_i and s_j, considered constraints include:

- the point-to-plane distance measured along the surface normal of s_j, from the centroid of s_j to the closest point of s_i, to be in a range between a minimum and a maximum;
- the oriented enclosed angle between the surface normals of s_i and s_j projected into both horizontal and vertical planes. For indoor scenes, these are very descriptive, because objects and structures are often aligned to some of the canonical room axis;
- the Euclidean distance between the two segments, to be lower than 1, to limit the influence of distant segments.

These constraints reduce the whole set of available segments to a small candidate subset, on which the feature learns the parameters for a binary test, to extract contextual information. Binary tests are different for each feature:

- *Existing segment:* the feature evaluates to true if at least one of the candidate segments passes the binary test, that is, it fulfills the geometrical constraints;
- *TopN:* based on the previous with the addition of intermediate label predictions for each candidate segments. The feature also learns a candidate label l and a positive integer N: if l is among the top N class labels of the distribution for at least one of the candidate segments, the feature evaluates to true;
- *Inverse TopN:* same binary test as above, but the angle difference is computed against the ground plane instead of s_i, and the point-to-point distance is measured inversely from the centroid of s_i to the closest point of s_j, to account for classes with non-regular surface normal (classes not composed by mainly planar surfaces);
- *Node descendant:* considers the path a segment s_i took through the tree during inference. The feature evaluates to true if the current tree node that s_i has reached is a descendant of tree node n, an additionally learned parameter;
- *Common ancestor:* the dual equivalent of the previous, additionally taking the classification path of the target s_j into account. The feature evaluates to true if the first common ancestor node of both s_i and s_j is reached after at most m steps, a learned parameter.

Chapter 6

Experiments on S3DIS

After some introductory details, the networks previously introduced in Chapter 4 and the 3DEF algorithm presented in Chapter 5 are separately tested and their performance on the Stanford dataset's 3D point clouds is compared.

6.1 Training and test split

Specific areas in the S3DIS dataset contain similar rooms, both in terms of content and building structure. As also noted in Chapter 3, the dataset has imbalanced semantic classes: bigger elements in rooms are made of more points than smaller objects. For a better learning, it is necessary to split data in training and test subsets in such a way that no similar areas and rooms appear together.

The authors in [52] propose three different splits, and then they perform cross-validation on them to avoid overfitting and to have a more balanced sampling. The first of these three proposals is used throughout all the following tests: Area 5 is kept as test set, while the remaining ones constitute the training set.

6.2 Evaluation metrics

The segmentation task is generally a M-class problem ($M > 1$) and segmentation algorithms may be evaluated using a $M \times M$ *confusion matrix*, that collects and divides the number of classified samples: a sample of such a matrix, restricted to a simpler 2-class problem, is depicted in Table 6.1. Having on columns actual "true" classes and on rows predicted labels from the algorithm, the diagonal of the matrix contains the number of all correctly matched samples, and the remaining constitutes the number of all the wrongly classified samples. In Table 6.1, a True Positive (TP) is a sample whose class is `A` and is correctly labeled as such, while a True Negative (TN) is a sample of `B` correctly classified: they both reside on the diagonal; False

Positives (FP) and False Negatives (FN) are, respectively, samples of `A` classified as `B` and vice-versa, and they are outside the diagonal.

Table 6.1: Sample confusion matrix, with example numbers. True labels are on columns and predicted labels are on rows. Definitions are w.r.t. the `A` class.

	`A`	`B`
`A`	TP = 100	FN = 10
`B`	FP = 5	TN = 50

Accuracy is defined as the ratio of true predictions to the total evaluated samples:

$$A = \frac{TP + TN}{TP + TN + FP + FN}$$

In a multi-class problem, accuracy can be computed separately for each class or as a single value over the total number of samples; in this case it is called *global* or *overall accuracy* and it is obtained from the diagonal of the matrix. Moreover, *Mean accuracy* is the average of per-class accuracies:

$$A_i = \frac{TP_i}{TP_i + FP_i} \qquad \text{MA} = \frac{1}{M}\sum_i A_i \qquad \text{OA} = \frac{1}{N}\sum_i TP_i$$

where N is the total number of considered samples and $i \in [1..M]$ is the class indexer. Note that the average in MA is *not* weighted according to the class distribution.

Precision P and *recall* R metrics are also equally meaningful in a multi-class problem: they are defined as the ratio of true positives (TP) to the total predicted positives (TP+FP), and the ratio of true positives to all predictions classified in the same class (TP+FN). The *F1-score* is the harmonic average between the two. Mathematically,

$$P = \frac{TP}{TP + FP} \qquad R = \frac{TP}{TP + FN} \qquad F1 = 2 \times \frac{P \times R}{P + R}$$

These metrics can be computed separately for each class.

The *Intersection-over-Union* (IoU) is another measure of the accuracy of predicted samples $\hat{Y}$ compared to the ground truth, that is, the true labels Y. The name derives from considering the number of common points in between the two sets divided by the number of points present in the union of the sets:

$$\text{IoU}(Y, \hat{Y}) = \frac{Y \cap \hat{Y}}{Y \cup \hat{Y}}$$

It can be rewritten using the above definitions, generating per-class and mean IoUs:

$$\text{MIoU}_i = \frac{TP_i}{TP_i + FN_i + FP_i} \qquad \text{MIoU} = \frac{1}{M}\text{MIoU}_i$$

All the presented metrics are widely used to compare algorithms and models in the literature for classification-like tasks, and will also be used here to evaluate and compare previously descripted architectures.

6.3 PointNet++

While the official implementation[1] of PointNet++ is developed using the TensorFlow framework [63], for system compatibility reasons this thesis uses the implementation brought by Erik Wijmans[2] which uses the PyTorch library [64].

The network accepts batches of 4096 points: in the training phase, when all points in the training subset have passed through the network it is called the end of an *epoch*, and metrics are computed. Occasionally, metrics are also measured against the test subset. During learning, increasing accuracy and diminishing loss values on the training set are typical, and the same behavior is expected on the test set: accuracy will improve but after a certain epoch the gap between training and test loss values gradually increase. This is an indication of overfitting and overall performance will get worse. In this case, the training phase lasted for 250 epochs, but from Figure 6.2 it can be seen from increasing loss values how overfitting occurs after more or less 30 epochs, and Figure 6.1 also shows that accuracy does not improve anymore: learning after this point is thus useless.

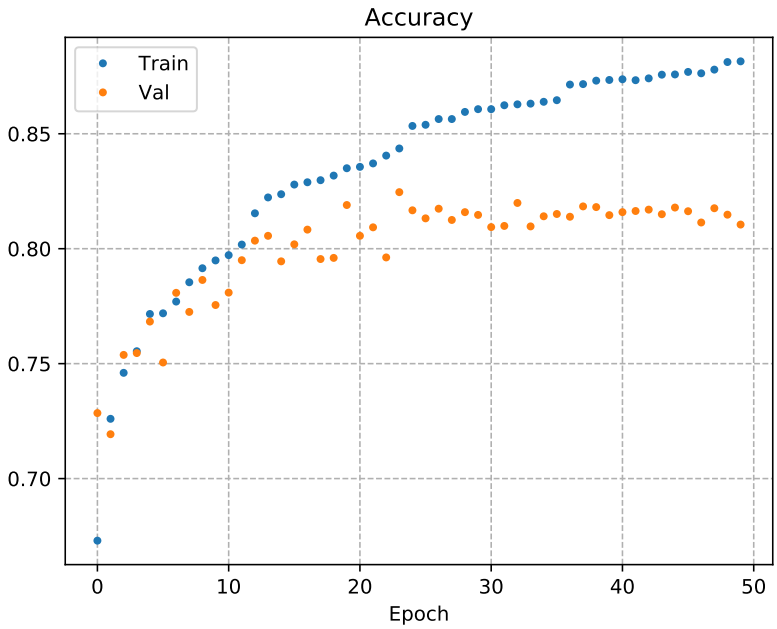


Figure 6.1: Accuracy values during training and validation phases.

[1] https://github.com/charlesq34/pointnet2

[2] https://github.com/erikwijmans/Pointnet2_PyTorch

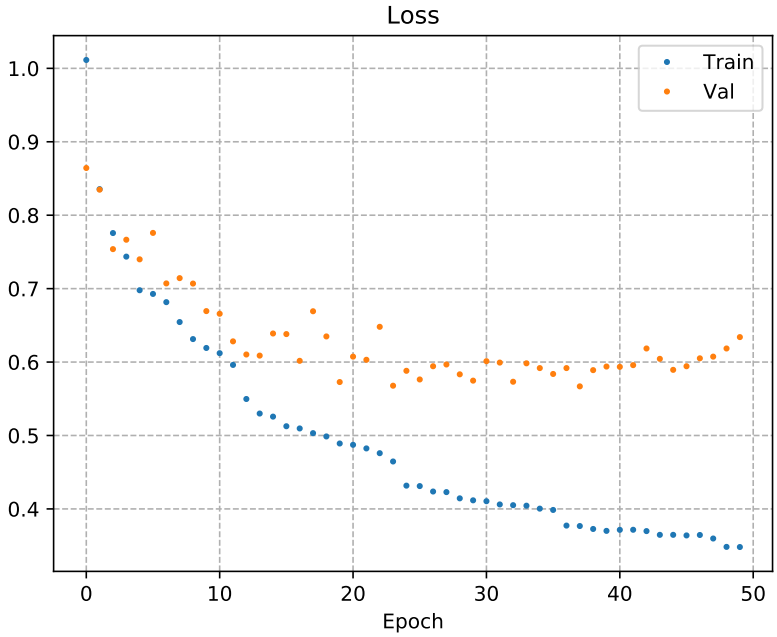


Figure 6.2: Loss values during training and validation phases.

6.4 Superpoint Graphs

The implementation of this network is available on GitHub[3] and it also uses the PyTorch framework.

Tests on this network focused on searching the best pair of *batch size*, the number of points that are simultaneously fed into the network, and single PointNets input point number (recall Figure 4.5). This can be challenging: increasing both the batch size and the point number leads to a huge GPU memory usage, but provides a low performance increment. Keeping both value low should result in the network not learning as broadly as it could: the batch size is also related to weight propagation between layers, and a greater size imply information exchange between more cells.

Figure 6.3 shows training and validation accuracy for the best tried configurations: the legend is structured as `s3dis-[batch size]-[point number]`. Other metrics for the same configurations are provided in Table 6.2. The baseline, provided by the authors of the network and also used in their paper, is the pair 2–128. For most tests validation was sparsely performed, because the aim was to rapidly increase the point number so as to reach values similar to what the full PointNet runs on: configuration 4–2048 (not depicted in the figure) quickly saturates the 24 GB of RAM installed on a Titan RTX. For the same reason, training of 8–1024 broke after every few epochs: manually restarting the process leads for the network to always see the same batches of data, and this leads to overfitting; training stopped after

[3] https://github.com/loicland/superpoint_graph

160 epochs due to errors in the procedure. The same figure also shows how there is no real improvement in validation accuracy for all tests after 100 epochs. Other configurations behave pretty similar and in the end 4–1024 was selected as the best choice, but 8–512 could equally be used as they are pretty close in accuracy. Plots for the selected configuration can be seen in Figure 6.4.

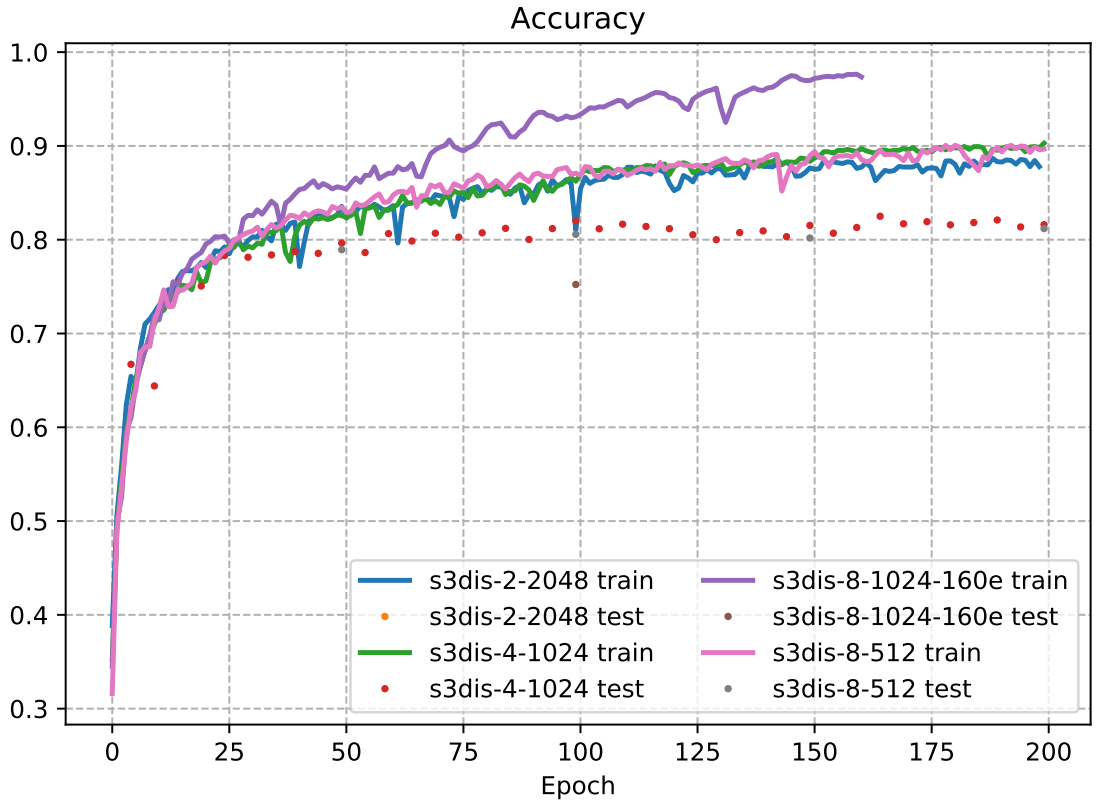
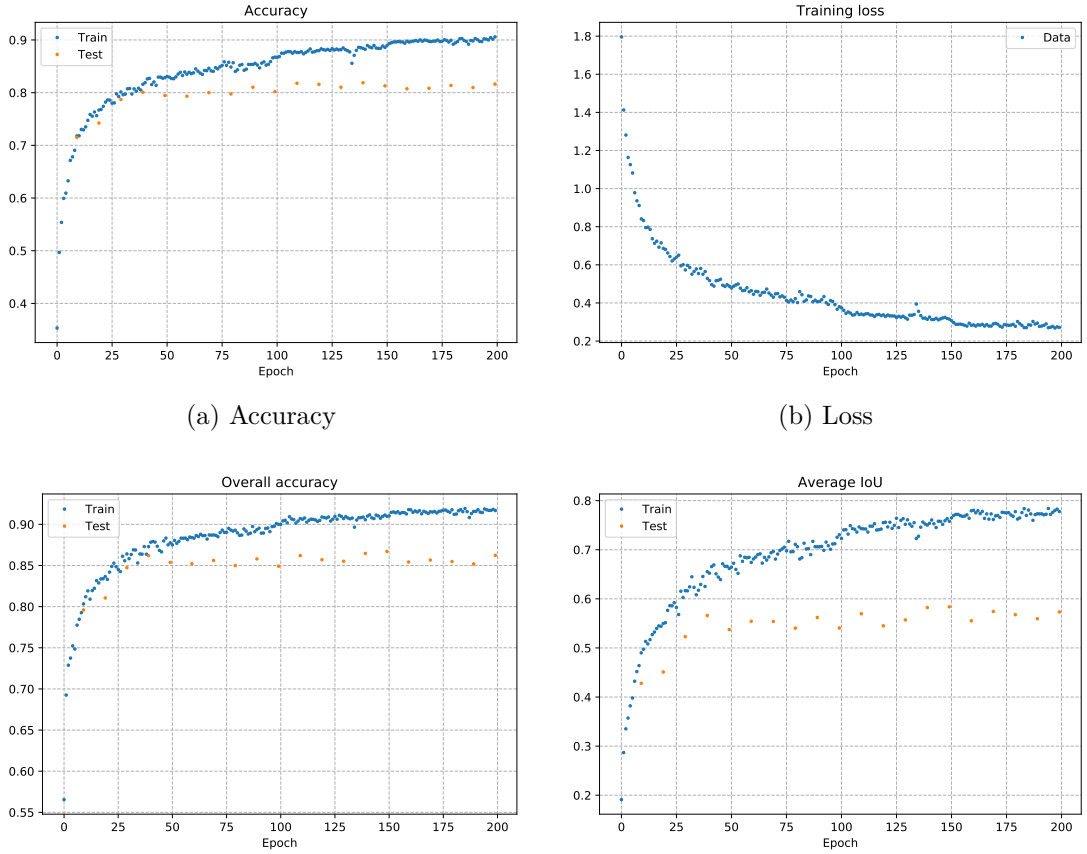


Figure 6.3: Accuracy values during training and validation phases, for several configurations.

Table 6.2: Values for training and validation phases, for several configurations. All metrics but loss are expressed as percentage value. Metrics were computed after 200 training epochs, except for * where values are for the 100th epoch.

	2 – 128		8 – 512		4 – 1024		2 – 2048		8 – 1024*	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Accuracy	88.06	81.28	89.66	81.18	88.88	81.06	87.79	81.21	93.10	75.22
Loss	0.329	—	0.287	—	0.320	—	0.322	—	0.188	—
Avg. IoU	74.95	53.80	77.77	57.40	76.00	56.49	74.42	60.56	79.68	50.27
Over. Acc.	90.77	85.92	91.68	85.56	91.17	86.14	90.53	86.38	92.32	82.68



(a) Accuracy

(b) Loss

(c) Overall accuracy

(d) Average IoU

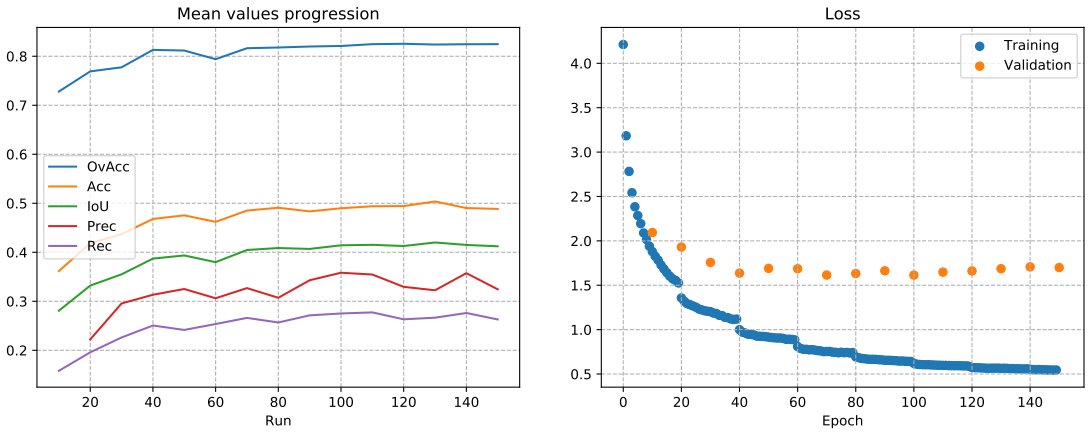
Figure 6.4: Plots for the selected configuration: batch size 4 and 1024 points as PointNets' input.

6.5 JSIS3D

The implementation is open-source[4]: the network is developed using PyTorch, and the CRF part is in C++. This latter part was not publicly available at the time of this thesis, so tests focused only on the deep network.

The network can be trained using a GPU, but performance evaluation, that is, inference on the test set, requires a CPU due to the use of Mean Shift algorithm [65] and is a long process. Tests involved running training for 10 epochs, saving a snapshot of the state of the network for inference, and going on for another 10 epochs as in a loop to seek the best epoch for optimal learning. Loss, overall accuracy and average metrics values for each test are presented in Figure 6.5: learning is nearly stable after epoch 40 and it does not show an overfitting indication.

[4] https://github.com/pqhieu/jsis3d



(a) Overall acc. and average metrics values

(b) Loss values during training

Figure 6.5: Plots for training phase.

6.6 Entangled forests

The 3DEF algorithm works on point clouds saved as PCD files, one for each room in the case of S3DIS, and the clustering program internally finds the room's floor plane to align the cloud: it expects to have floor points' in a nearly horizontal plane. However the original point clouds structure is such that the point of view is from above, that is, the floor of the room is a nearly vertical plane in the 3D space. A preprocessing step was thus necessary to rotate the cloud.

Assuming a global reference frame O oriented with the person's point of view, a rotation by -180° along the O_x axis and a subsequent shift upwards in the O_z axis are needed to quantitatively align the floor of the rooms with a horizontal plane before feeding the clouds to the clustering program. This is a logic assumption if operating for example with robot-mounted cameras that will see the floor in the bottom part of the field of view. This is achieved using a coordinate transformation matrix $\boldsymbol{T}$ to rotate each point $\boldsymbol{p}$ of coordinates $[x, y, z]^\top$ into $\boldsymbol{p}_N$ of coordinates $[x_N, y_N, z_N]^\top$ of an angle α (in radians):

$$\begin{bmatrix} x_N \\ y_N \\ z_N \end{bmatrix} = \boldsymbol{p}_N = \boldsymbol{T}\boldsymbol{p} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

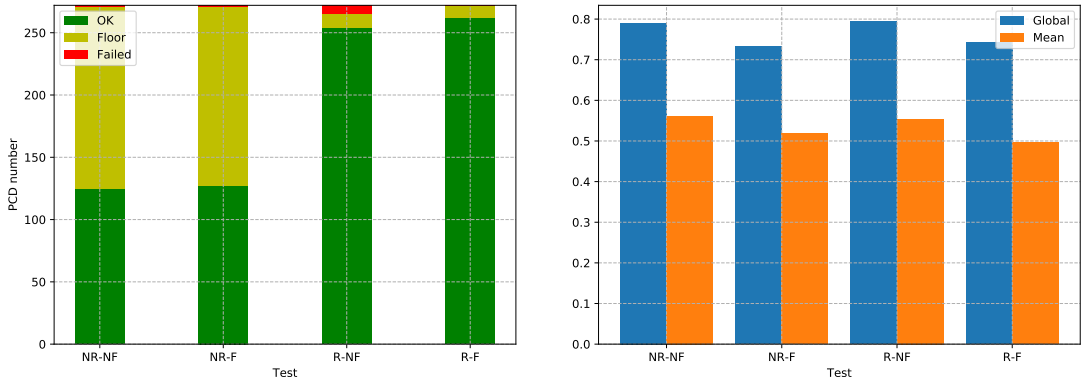
As a convention, counterclockwise rotations have $\alpha > 0$.

A proof that the rotation is needed can be found in Figure 6.6a comparing `NR-*` and `R-*` tests: of the 259 PCDs (rooms) that make the dataset, if no rotation is applied more than half of them, colored in yellow in the histogram bars, are not processed because for the algorithm the floor is missing, while instead its points are aligned exactly in the vertical plane and thus are not detected. Figure 6.6b also shows that rotation also slightly improves both overall accuracy and mean accuracy among all

the classes, but the improvement is not substantial. While the rotation solves this issue, still some PCDs cannot be processed due to errors in the segmentation part and clusters cannot be built.

3DEF algorithm allows to use some filters during the preprocessing step. A second test is whether bilateral and outlier filtering should be used. Bilateral filtering necessitates an *organized* PCD, that is, the assumption is that it exists a point of view in the space from where points in the cloud can be seen as aligned to a 2D grid. This is clearly not the case for this dataset, where clouds are simply a list of points and have no structure. Outlier filters work but accuracy values is lower in any test they are used, as it can be seen comparing `*-F` with respect to `*-NF` tests in Figure 6.6b.

The above considerations drove to select to rotate PCDs and not applying any filtering, that is, `R-NF` is selected as the winning configuration.



(a) Histogram of floor detections in PCDs.

(b) Overall and mean accuracy values.

Figure 6.6: Statistics on the preprocessing steps of 3DEF algorithm, for the choices of whether to rotate PCDs (abbreviated as `R` or `NR`) and to apply filtering (`F` or `NF`).

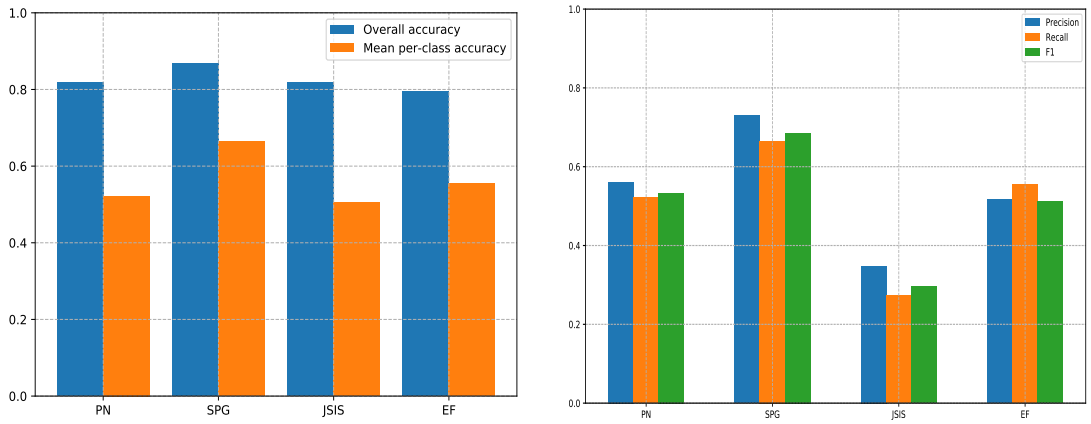
6.7 Cumulative evaluation

Here we present the summing results of the preliminary tests: PointNet++, Superpoint Graphs in the 4–1024 configuration and JSIS3D have been retrained up to the optimal number of epochs, and their evaluation is compared with 3DEF's performance.

Figure 6.7 and Table 6.3 show that SPG has better overall accuracy and mean per-class accuracy compared to the other deep networks and to 3DEF. The difference can be noted also looking at the unfolded per-class accuracies as presented in Figure 6.8 and Table 6.4: we can clearly distinguish "major" classes like *wall*, *floor* and *ceiling*, that are well recognized by all the algorithms, but less represented objects are the ones that contribute most to the improvement SPG shows w.r.t. the competition: look for example at *column*, that is never well classified except by SPG, or to all the subsequent classes (in the figure) where SPG excels, except *board* and *beam*.

Classes like *column* or *beam* are almost never well classified but overall accuracies are nonetheless high because, as mentioned in Chapter 3, there are less points to be classified in such classes compared to sets of walls' or similar classes points.

It is interesting to note that, even if 3DEF is based on a different algorithm, namely random forests, it achieves similar overall accuracy values w.r.t. deep learning networks. Looking at the mean accuracy, 3DEF is better than two out of three deep networks: it is thus comparable to the state of the art, while not requiring a GPU or high computational power; it can be a good choice for mobile robotics applications, where limits are strict due to power constraints.



(a) Overall accuracy

(b) Precision, recall and F1

Figure 6.7: Overall accuracy and global metrics for each architecture on S3DIS.

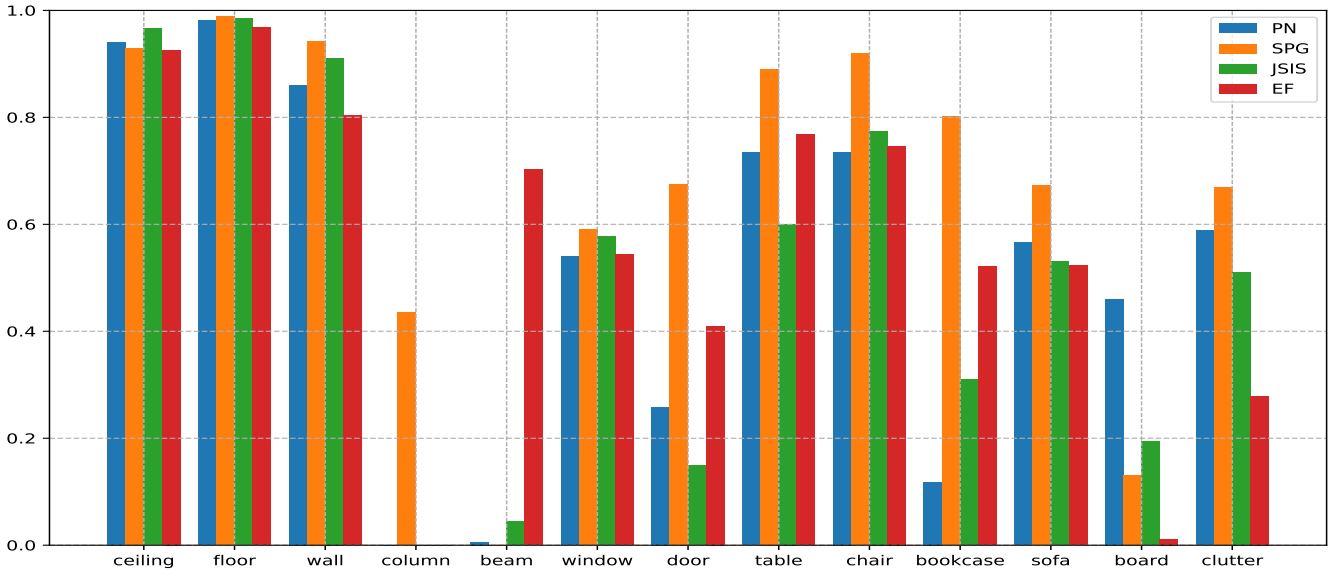


Figure 6.8: Accuracy for each class and tested architecture on S3DIS.

Table 6.3: Evaluation metrics values for each architecture.
IoU was not implemented in 3DEF at the time of these tests.

Arch.	Epochs	Ov. Acc.	Avg. Acc.	Avg. Precision	Avg. Recall	Avg. F1	Avg. IoU
PN++	50	0.81836	0.52223	0.56117	0.52223	0.53315	0.42833
SPG	100	0.86670	0.66523	0.73081	0.66523	0.68552	0.58100
JSIS3D	70	0.81799	0.50408	0.34824	0.27385	0.29637	0.41937
3DEF	—	0.79416	0.55406	0.51587	0.55406	0.51153	—

Table 6.4: Accuracy for each class and tested architecture.

Arch.	ceiling	floor	wall	column	beam	window	door	table	chair	bookcase	sofa	board	clutter
PN++	0.9411	0.9810	0.8603	0.0	0.0063	0.5395	0.2587	0.7341	0.7349	0.1178	0.5662	0.4602	0.5885
SPG	0.9294	0.9884	0.9430	0.4351	0.0	0.5901	0.6757	0.8908	0.9198	0.8019	0.6723	0.1315	0.6695
JSIS3D	0.9658	0.9847	0.9107	0.0	0.0438	0.5775	0.1494	0.6009	0.7748	0.3101	0.5300	0.1940	0.5109
3DEF	0.9258	0.9684	0.8039	0.0	0.7020	0.5437	0.4102	0.7681	0.7463	0.5208	0.5228	0.0114	0.2787

Chapter 7

Exchanging features between architectures

Among the three analyzed deep networks, SPG is the one that most resembles 3DEF, in terms of preprocessing steps: the former works by building small graphs on the cloud, the latter builds clusters. Both algorithms extract features on these structures, on which the network and the forest is trained, respectively. To try to improve the performance of both algorithms, we experimented in mixing, adding and removing these features in both directions.

7.1 Experiments on 3DEF

We wanted to have an insight on the most relevant and useful unary features for training the forests. Several tests have been conducted:

1. Linearity, planarity, scattering and verticality, part of SPG's geometric features as discussed in Section 4.2, have been added to the original set of 19 unary features already part of 3DEF. The aim was to see if they could add any new information or if they are more efficient and useful than existing features.
2. Forests have been trained on both geometric and graph features as computed by the SPG partition script. The aim was to see if forests can be trained on graphs preprocessed by SPG, and how their performance compares to those trained on 3DEF's clustering output.

7.1.1 SPG features added to 3DEF clustering part

A different forest has been trained on the rotated PCDs of the S3DIS dataset, preprocessed by the 3DEF clustering script:

- With the original 19 unary features (the baseline for comparisons);
- With only the 4 features inherited from SPG, plus color;

- With both the original 19 and the 4 inherited unary features.

For each test entangled features might be enabled or not.

Figure 7.1 visually provides insights that are numerically presented also in Table 7.1. The original configurations with 19 unary features (tests `19` and `19+E`) are compared to using color plus the 4 inherited features from SPG, with and without entangled features (tests `4` and `4+E`): entangled features play an enormous role in enhancing performance in the last two tests, meaning that this set of unary features alone is not able to successfully summarize input data. Using all the original and inherited features (tests `23` and `23+E`) leads to slightly lower accuracy and metrics values, compared to the baseline: this evidences how adding these four extra features is not providing new ways for the algorithm to learn from data. This behavior could be an indication that the analyzed features are not adequate or meaningful w.r.t. the original set of features, or that 3DEF has been pushed forward to the limit where it is no more possible to learn new meaningful things on the specific dataset; this can be confirmed by the fact that entangled features provide less performance enhancement w.r.t. the previous tests.

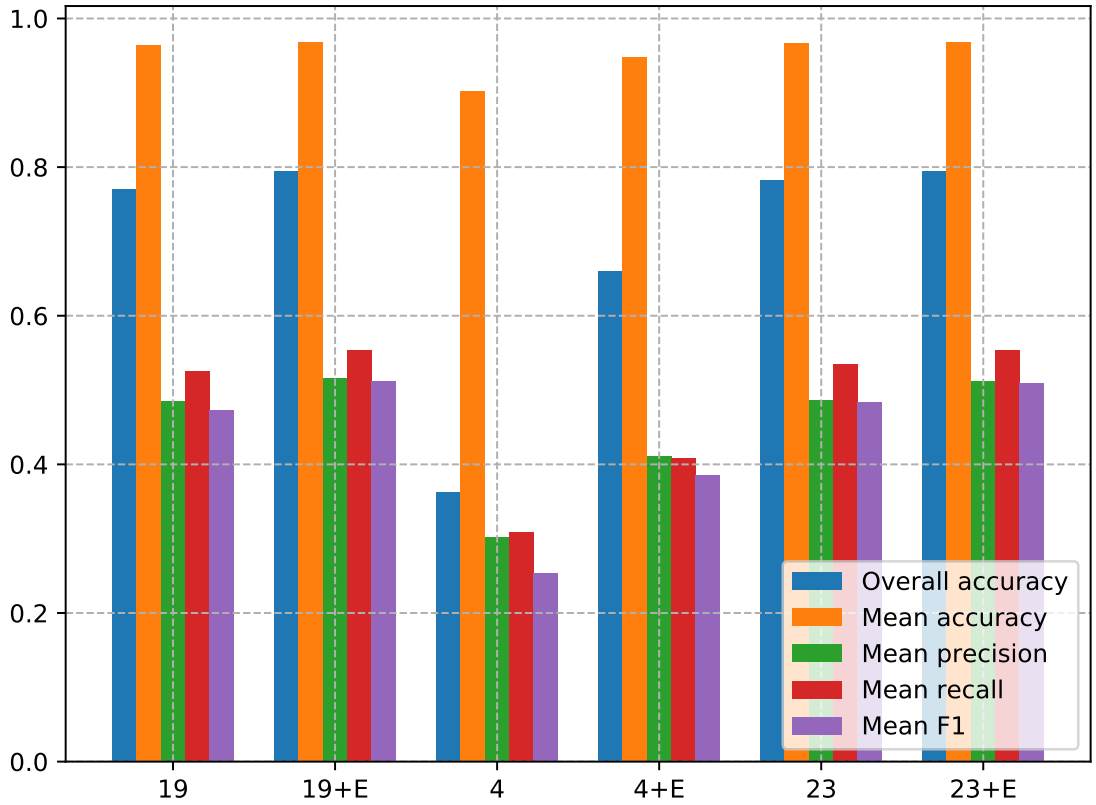


Figure 7.1: Accuracy and metrics for each test.
Metrics are obtained by averaging each class' single values.

Table 7.1: Evaluation metrics values for each test.

Test	Ov. Acc.	Avg. Acc.	Avg. Precision	Avg. Recall	Avg. F1
19	0.77009	0.96463	0.48548	0.52493	0.47309
19+E	0.79416	0.96833	0.51587	0.55406	0.51153
4	0.36211	0.90186	0.30166	0.30883	0.25391
4+E	0.66024	0.94773	0.41070	0.40794	0.38556
23	0.78215	0.96649	0.48667	0.53425	0.48333
23+E	0.79407	0.96832	0.51209	0.55329	0.50992

Figure 7.2 represents accuracy values for each class present in the database, comparing all the tests. Even if results may seem pretty high for all classes, bad performance of test `4` may be explained by noting that the forest missed points in the more represented classes like *ceiling*, *floor* and *wall*, and the global accuracy value suffered from this, while the average of the accuracy values stayed high because most values were around 90% and the average is *not* weighted on the frequency of points for each class. Again, entangled forest mitigated this issue, as it can be seen by the huge gap between the two tests' accuracy values of the first three classes in the figure. Full numerical results are in the upper part of Table 7.7.

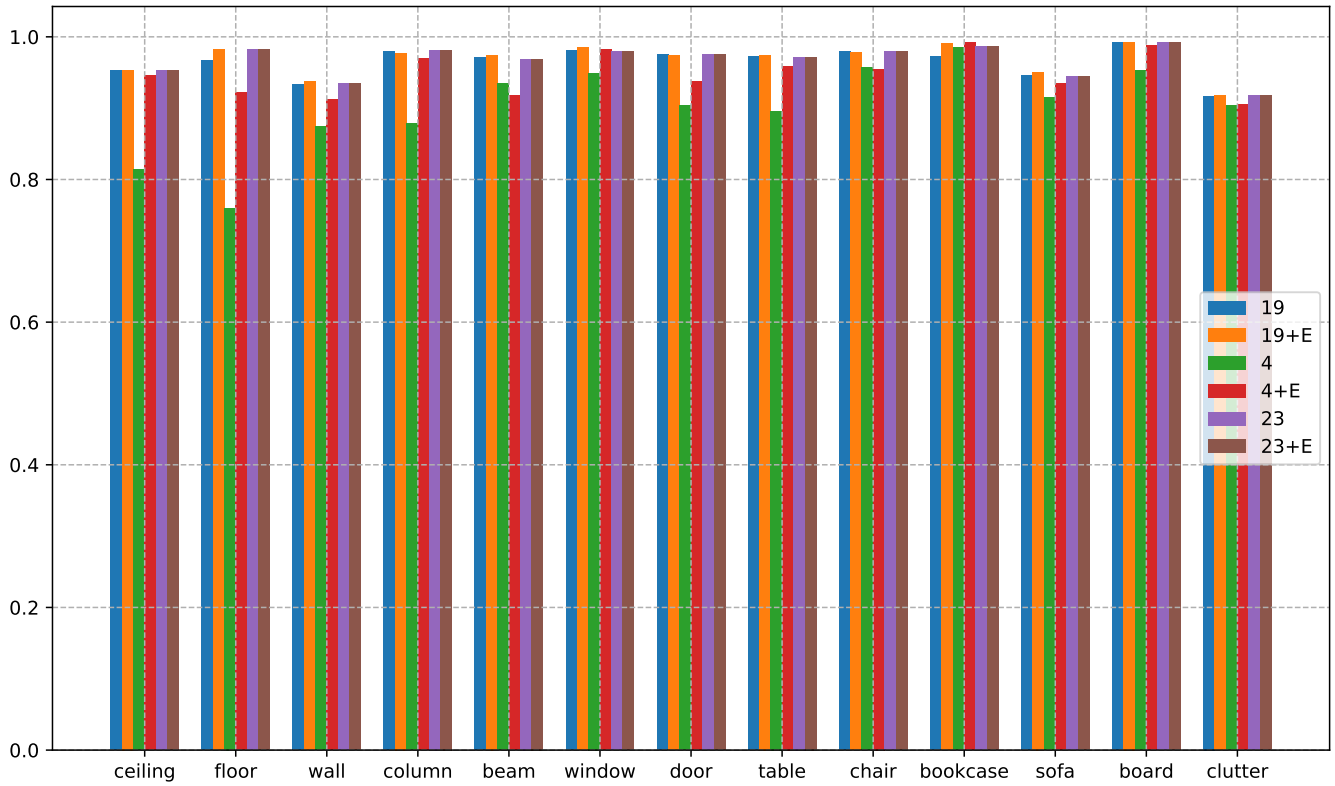
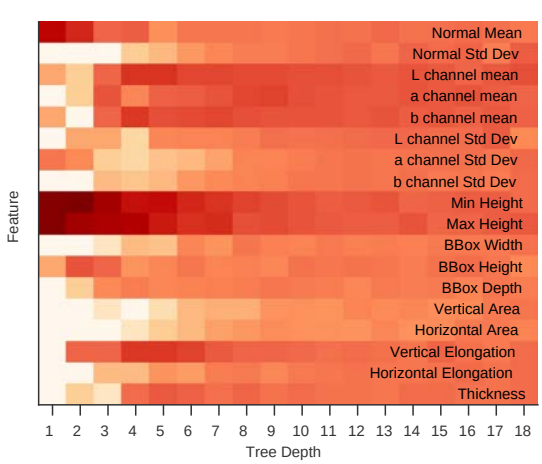
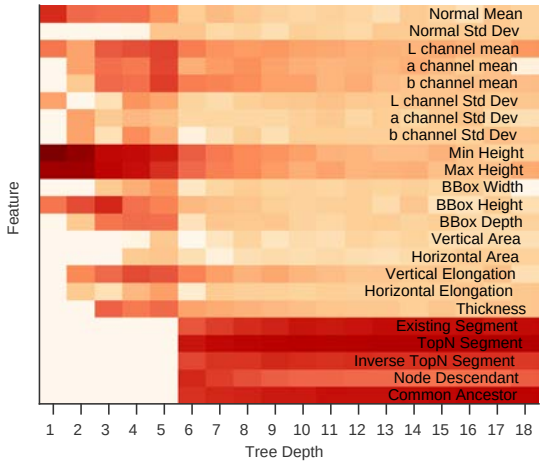


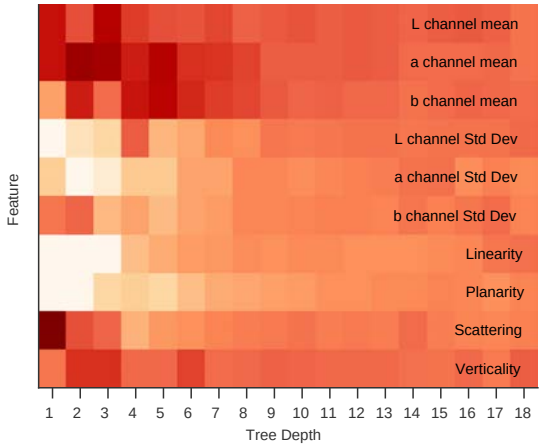
Figure 7.2: Accuracy for each class and for each test.



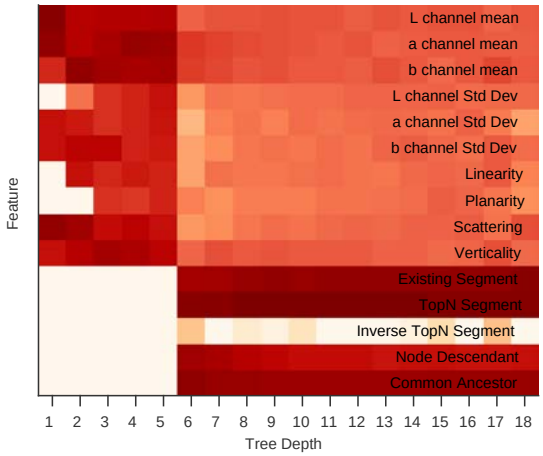
(a) Original conf., 19 unary features



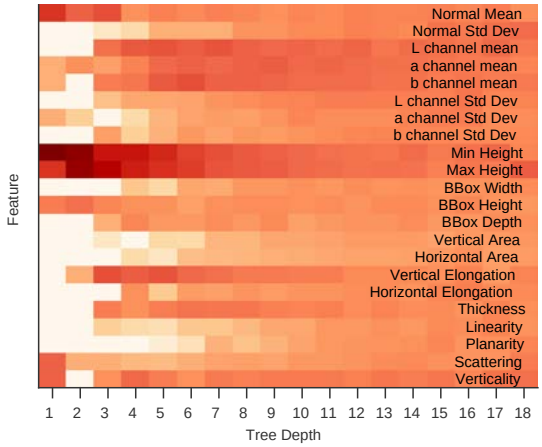
(b) Same as (a), plus entangled features



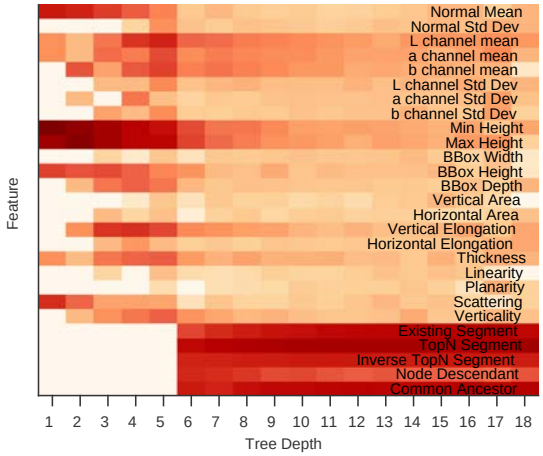
(c) Color and 4 inherited unary features



(d) Same as (c), plus ent. features



(e) All 19 plus 4 unary features



(f) Same as (e), plus ent. features

Figure 7.3: Importance of features during trees' trainings, for all tests. The darker the color, the more the feature was used for splits while growing trees.

A tool of 3DEF provides a way to analyze the trained forest to have insights on the relevance of features during subsequent steps, that is, related to trees' depth. Figure 7.3 visually present this information: a darker color in the plot means the feature was used a lot to perform splits between sets of points in the training phase. For tests `23` and `23+E` it can be noted how features that come from SPG are among the less used, apart from verticality: 3DEF does not find these features to be so relevant to made splits based upon them. Definitely, when entangled features come into play, they clearly dominate the scene; in tests where they are not enabled we can see how scattering and heights are among the most relevant for decisions. Some features related to color are needed to leverage this information during training.

7.1.2 Training forests on SPG features

For these tests, the structures created by the SPG partition script, that is, archives for both geometric and superpoint graphs' features, are converted in a format that is readable by 3DEF's forest training program. Then, for each point cloud, that is, for each room in the S3DIS dataset, a unique cluster ID is associated to each of the 10-neighborhoods point sets, and to every component of each superpoint graph, respectively (recall Figure 4.6): in this way, graphs are converted to clusters by simply considering only their nodes.

Clusters generated by using the 10-neighborhoods sets overlap, and each point ends up belonging to more than one cluster: to avoid ambiguity in decisions, the distance between the point and the centroid of the neighborhood is used, that is, the point is assigned to the nearest cluster, the one that minimizes the distance. But at some time in the computation each point ends up being a centroid for its neighborhood: clusters degenerate to comprise only a single point (the centroid itself), and this made clustering nearly useless. Nonetheless, a forest can be trained on this setup, it just takes a lot more time to process all the points. Instead, components of the SPG did not overlap and the conversion process has been successful.

3DEF preprocessing was skipped and forests were directly trained on the specific set of features, and evaluation is performed against the relative converted clusters.

Figure 7.4 allows for metrics comparisons between the same baseline of previous tests, a forest trained with 19 unary (and no entangled) features on 3DEF's clusters, compared to those trained with SPG's geometric and graph features on clusters derived from superpoint graph's components. Poor overall accuracy is explained using with the same previous discussion: the forest is weak in classifying the most representative labels, as it can be seen looking at low accuracy values for *ceiling*, *floor* and *wall* on Figure 7.5. Moreover, for the same leitmotiv the unweighted average of accuracy values for each of the classes, orange in the figure, is high as in the previous case (compare to Figure 7.1) because all the values are pretty high, and most of them are above 80%.

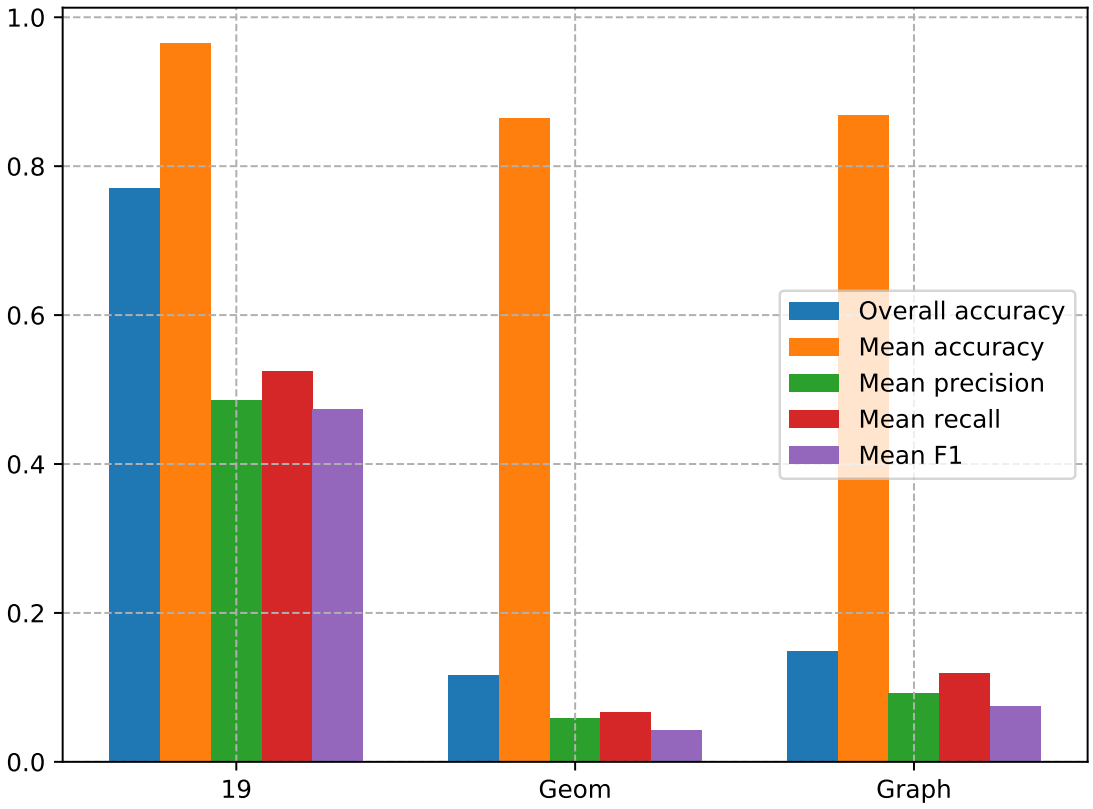


Figure 7.4: Accuracy and metrics for each test.
Metrics are obtained by averaging each class' single values.
Test 19, the same as before, is kept for comparisons.

Table 7.2: Evaluation metrics values for each test.

Test	Ov. Acc.	Avg. Acc.	Avg. Precision	Avg. Recall	Avg. F1
19	0.77009	0.96463	0.48548	0.52493	0.47309
Geom	0.11596	0.86399	0.05839	0.06689	0.04244
Graph	0.14790	0.86891	0.09196	0.11891	0.07478

Numerical results are presented in Table 7.2 and in the lower part of Table 7.7. What is not numerically reported is that forests trained on graphs' nodes features are trained way more fast than their counterpart built over geometric features, because graphs have been already reduced from the whole cloud when their nodes' features are computed, while geometric features are computed on subsets of the whole cloud and their number is by orders of magnitude higher than the number of graph's components.

Figure 7.6 shows the importance of features during training: points' coordinates are among the most used for splits, along with scattering and verticality in `Geom` test.

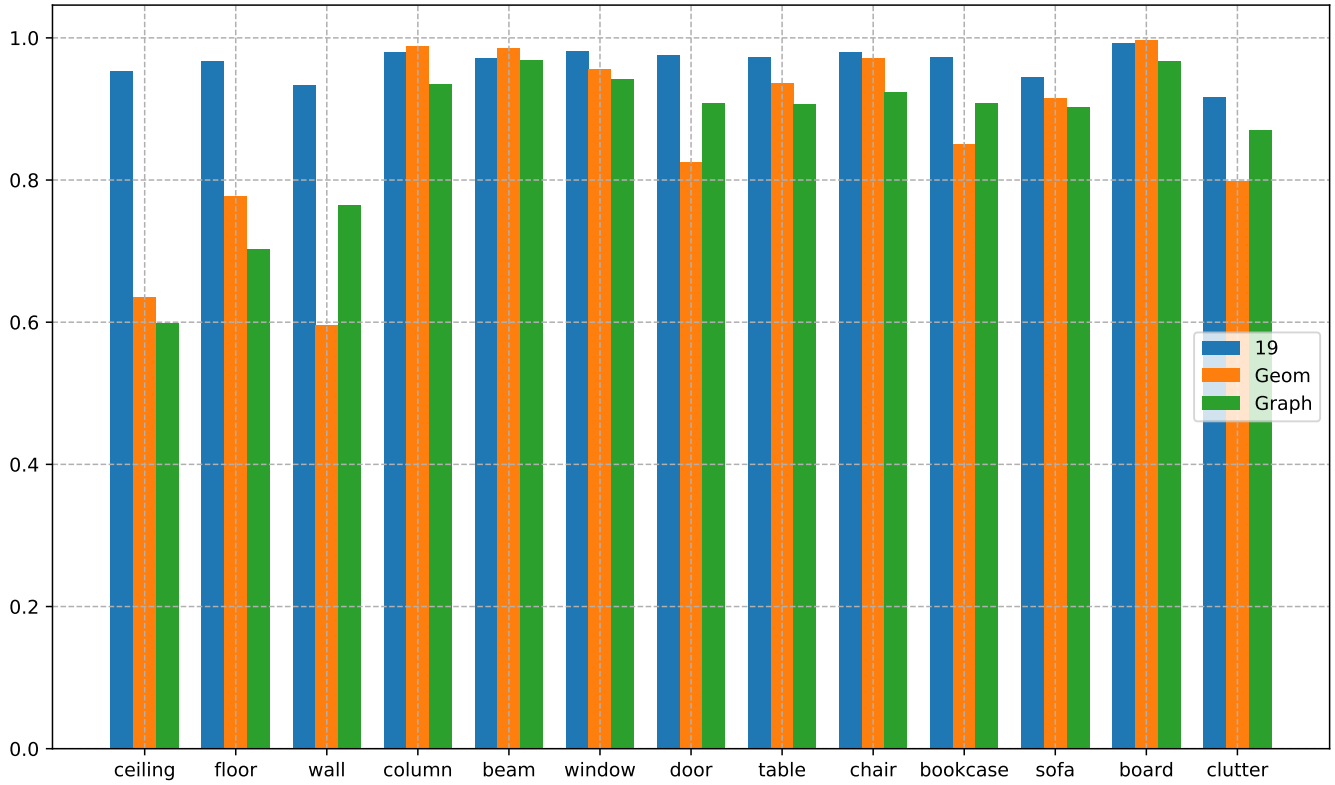
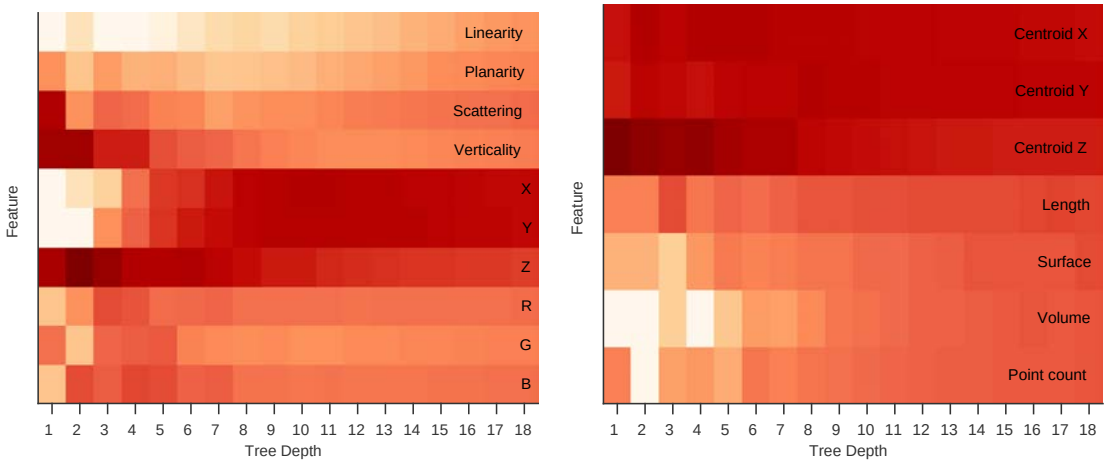


Figure 7.5: Accuracy for each class and for each test.
Test 19, the same as before, is kept for comparisons.



(a) Geom: features from 10-neigborhoods (b) Graph: SPG nodes' features

Figure 7.6: Importance of features during trees' trainings, for both tests.

7.2 Experiments on SPG

In this opposite scenario, unary features from 3DEF clustering script are brought into the SPG partition process. The next step in the pipeline after partition is to prepare archives that can be consumed by the network's learning phase: these archives are built based on the SPG graph components and by concatenation of points' coordinates X, Y, Z, color components R, G, B, features and normalized points' coordinates w.r.t. the whole cloud.

7.2.1 Porting features from 3DEF

It is easy to add features during the partition phase and the most straightforward to be transferred are the one that refers to the cluster's bounding box (BB): the concept is directly translated to virtually have the box to be wrapped around graphs' nodes instead of considering clusters.

Porting the whole block of BB-related unary features resulted in an error during the graph reduction phase as operated by the ℓ_0-cut pursuit algorithm mentioned in Section 4.2. As a result, features were gradually included in blocks and several tests were conducted; beside of the original linearity, planarity, scattering and verticality already part of the original algorithm (`original` test), these were separately added:

- BB's dimensions (width, depth and height, `whd` test);
- BB's area in the vertical and horizontal directions (`area` test);
- BB's minimum and maximum height (`height` test);
- BB's ratios between dimensions (elongation in the two planes and "thickness");

These tests evidenced how the problem in the optimization derived from considering features based on the ratios between dimensions, this is the reason why they were excluded in the last test that was made, that comprise all the features used in previous tests, namely SPG's original four plus the three dimensions, two areas and two heights (`all` test).

Tests' evaluations are compared to the original configuration as a baseline. Note that this *differs* from results presented in Chapter 6 for SPG because our own version of the source code has been updated in the meantime, and a bugfix was made available by the original author[1] that in fact lowered performance for all these tests: as the baseline is obtained using the same code of the tests, comparisons are reliable.

Training evolutions are presented in Figure 7.7. In general validation accuracy is almost always in the range $[0.45,\ 0.6]$, even if learning is much more difficult when considering area features; simpler features like those made by single values or that involve only differences seem not to worsen accuracy and yield the best results.

[1] See issue 127 on the GitHub repository for more information on the bug.

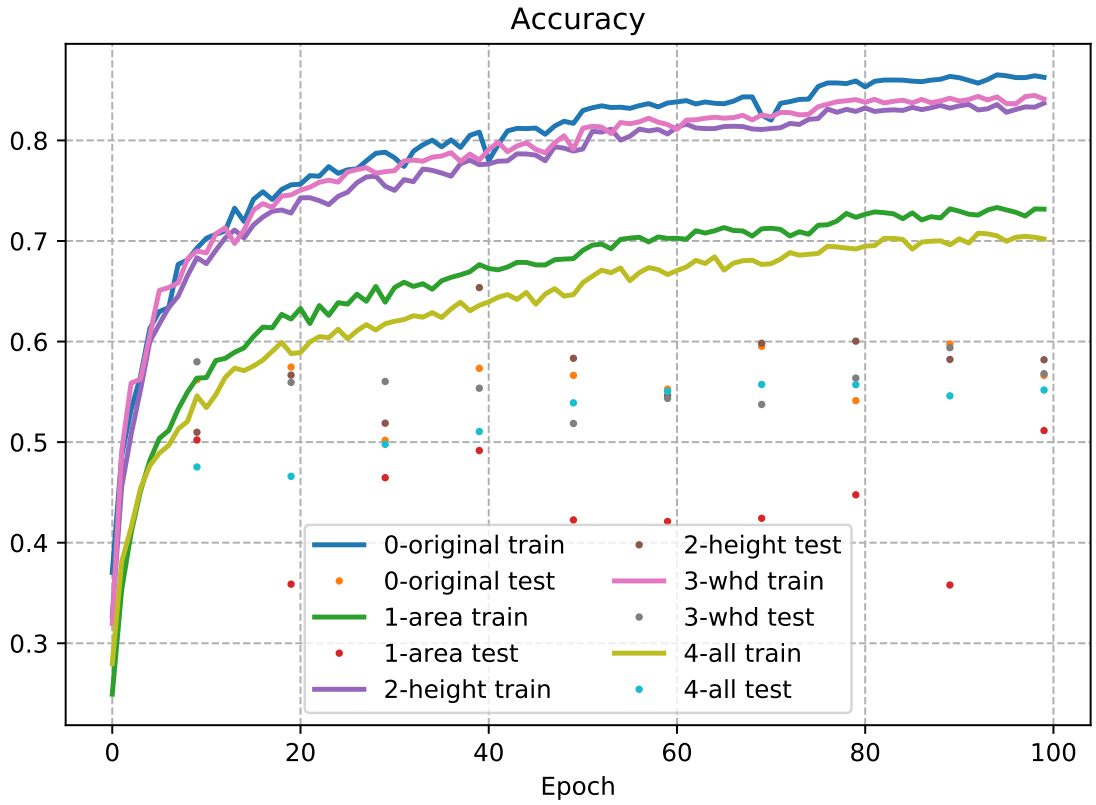


Figure 7.7: Accuracy values during training and validation phases, for all the different tests. The legend shows what features have been added to the standard 4: the first number is merely a counter to keep the progression order.

Adding minimum and maximum heights enhances performance by a little over the original configuration: this is also confirmed by the overall accuracy and other metrics in Figure 7.8. Numerical results are presented in Table 7.3 for metrics and in the upper part of Table 7.8 for accuracy values for each class.

Table 7.3: Evaluation metrics values for each test.

Test	Ov. Acc.	Avg. Acc.	Avg. Prec.	Avg. Recall	Avg. F1	Avg. IoU
original	0.75715	0.43998	0.56496	0.43998	0.43427	0.34374
area	0.69883	0.38281	0.53019	0.38281	0.36615	0.28711
height	0.75943	0.44840	0.59033	0.44840	0.45081	0.35928
whd	0.72243	0.36627	0.60306	0.36627	0.36805	0.29261
all	0.72415	0.39747	0.59064	0.39747	0.42053	0.32236

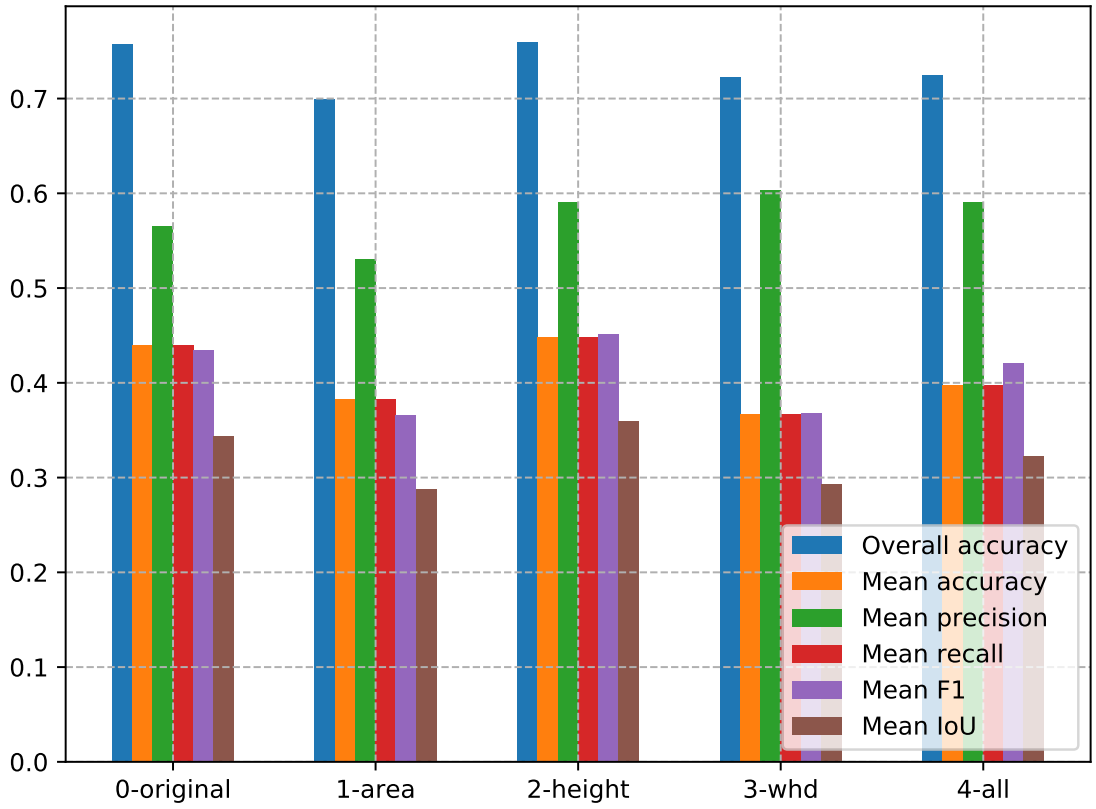


Figure 7.8: Accuracy and metrics values for each test.
Mean values are obtained by averaging single per-class values.

7.2.2 Changing color space

As the 3DEF algorithm uses the LAB color space and means and standard deviations for every channel are part of the unary features, a test has been conducted swapping RGB values with their LAB counterpart in the archives. `RGB` test is thus equal to `original` of Subsection 7.2.1. However, Figure 7.9 and Table 7.4 show that there is no real improvement in changing the color space: performance is slightly worse due to "minor" classes being less correctly classified, as it can be seen numerically comparing their accuracy in the middle part of Table 7.8. This evidence can be noted also by looking at validation accuracy during training, as per Figure 7.10.

Table 7.4: Evaluation metrics values for each test.

Test	Ov. Acc.	Avg. Acc.	Avg. Precision	Avg. Recall	Avg. F1	Avg. IoU
RGB	0.75715	0.43998	0.56496	0.43998	0.43427	0.34374
LAB	0.73861	0.40119	0.51841	0.40119	0.38360	0.31177

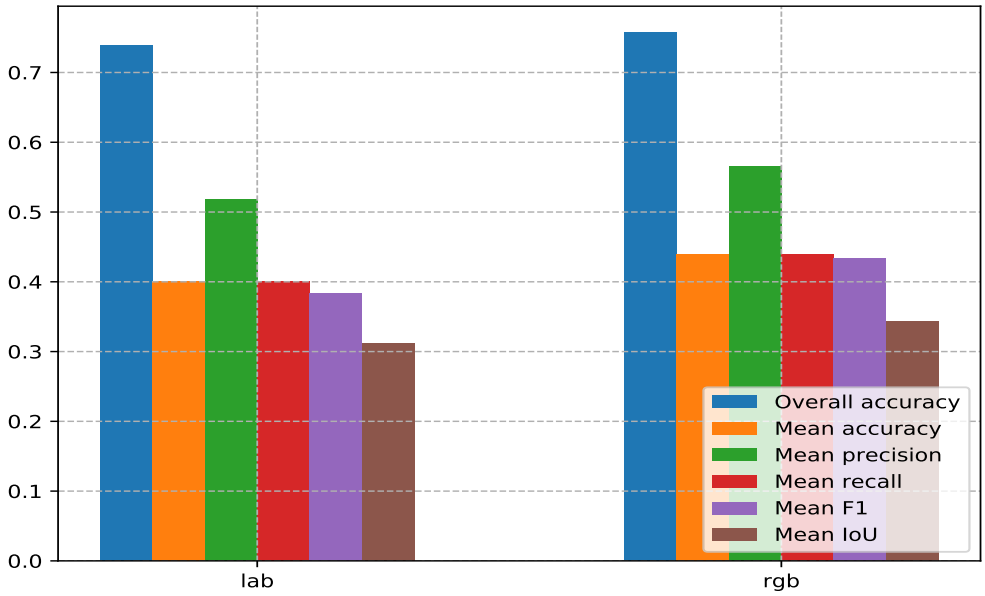


Figure 7.9: Accuracy and metrics values considering the LAB color space.

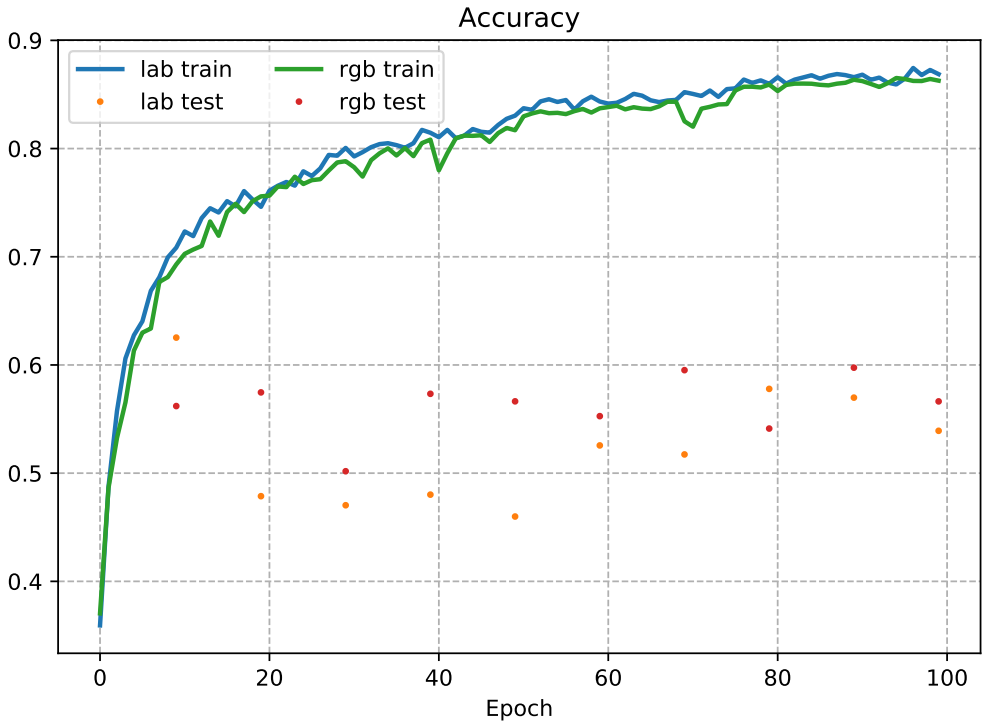


Figure 7.10: Accuracy values during training and validation phases, considering the LAB color space.

7.2.3 Enabling features in different steps

An interesting question is where features are more relevant: previously mentioned tests have been made by computing features in the early stages and then having them for all the pipeline (recall Figure 4.6). We tried a different approach and built a test configuration where additional features were computed on the 10-neighborhood sets and then kept for the graph reduction phase but discarded for learning by the network, and the opposite, that is, to compute features and to use them only for learning but not in the previous stage of computing the superpoint graph. A summary of the tests is displayed in Table 7.5. We chose to work with height features because, as evidenced in previous tests in Subsection 7.2.1, they may improve over the baseline.

Table 7.5: Use of features for every test.

Test	SPG computation	Network learning
Full	Original + height	Original + height
Only lrng	Original only	Original + height
Only SPG	Original + height	Original only

Figure 7.11 exposes mixed outcomes: comparing the original height test (`full` in the figure, which is the same as `height` in Subsection 7.2.1) with `onlylearning`, the test where superpoint graphs are computed only on the original 4 features and *not* height ones but learning occurs on the whole set of features, it can be noted how all metrics are lower: this can be seen as the fact that the algorithm benefits from having more features to aid in drawing the graph structure. A higher training accuracy for the `onlylearning` test may also be an indication of overfitting input data, as previously seen for certain preliminary tests. On the opposite, a network that learns only from the 4 original features *and* the graph constructed from a more solid foundation with more features may enhance performance, as it can be seen from the last epochs in the figure, where validation accuracy has started to rise over the opponents. Nonetheless, at the last epoch performance between the baseline and `onlyspg`, the test where superpoint graphs are computed on the full set of features but learning only occurs on the original 4 features, is comparable, as evidenced by Figure 7.12.

Numerical results are exposed in Table 7.6 and in the lower part of Table 7.8.

Table 7.6: Evaluation metrics values for each test.

Test	Ov. Acc.	Avg. Acc.	Avg. Prec.	Avg. Recall	Avg. F1	Avg. IoU
Full	0.75943	0.44840	0.59033	0.44840	0.45081	0.35928
Only lrng	0.72437	0.37617	0.60249	0.37617	0.38880	0.30403
Only SPG	0.75591	0.43670	0.65957	0.43670	0.44633	0.35339

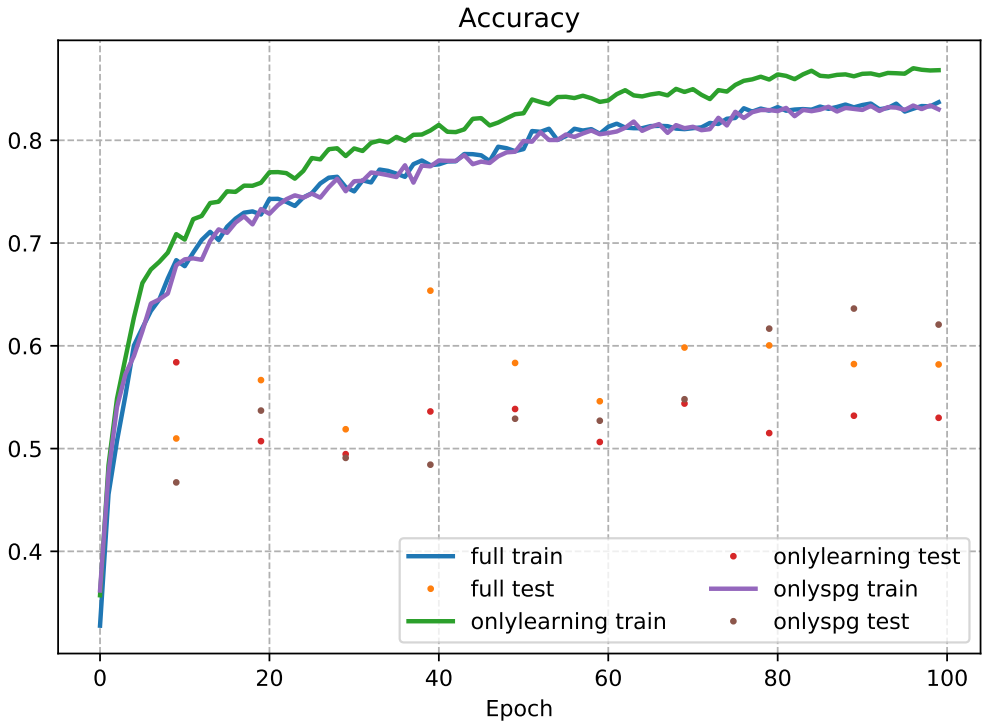


Figure 7.11: Accuracy values during training and validation phases, for both tests.

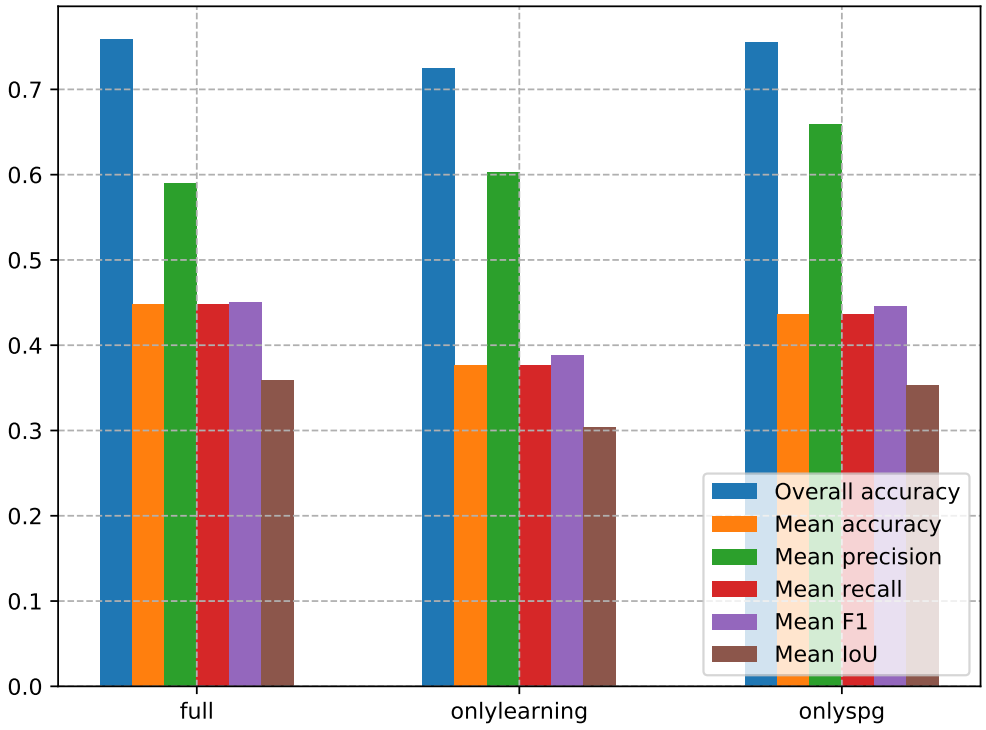


Figure 7.12: Accuracy and metrics values for both tests.

Table 7.7: Tests on 3DEF. Accuracy for each class and test.

Test	ceiling	floor	wall	column	beam	window	door	table	chair	bookcase	sofa	board	clutter
19	0.9525	0.9673	0.9327	0.9799	0.9714	0.9809	0.9757	0.9724	0.9796	0.9725	0.9453	0.9924	0.9169
19+E	0.9533	0.9820	0.9368	0.9774	0.9745	0.9853	0.9742	0.9745	0.9780	0.9910	0.9505	0.9928	0.9175
4	0.8142	0.7601	0.8740	0.8788	0.9342	0.9493	0.9033	0.8958	0.9572	0.9850	0.9152	0.9528	0.9035
4+E	0.9454	0.9220	0.9122	0.9697	0.9175	0.9824	0.9374	0.9591	0.9548	0.9920	0.9340	0.9883	0.9051
23	0.9526	0.9820	0.9345	0.9815	0.9688	0.9792	0.9747	0.9715	0.9789	0.9869	0.9440	0.9916	0.9173
23+E	0.9502	0.9823	0.9371	0.9802	0.9746	0.9834	0.9764	0.9747	0.9779	0.9904	0.9530	0.9934	0.9141
Geom	0.6349	0.7777	0.5959	0.9888	0.9853	0.9565	0.8256	0.9357	0.9716	0.8497	0.9149	0.9962	0.7985
Graph	0.5985	0.7027	0.7643	0.9347	0.9682	0.9415	0.9079	0.9062	0.9241	0.9074	0.9025	0.9675	0.8698

Table 7.8: Tests on SPG. Accuracy for each class and test. Symbols mark equal tests, repeated for convenience.

Test	ceiling	floor	wall	column	beam	window	door	table	chair	bookcase	sofa	board	clutter
original†	0.9232	0.9929	0.9524	0.0475	0.0	0.3060	0.2795	0.4418	0.9203	0.2077	0.0	0.0	0.6480
area	0.7905	0.9913	0.9238	0.0145	0.0	0.0970	0.1810	0.4727	0.7570	0.1245	0.0023	0.0007	0.6205
height*	0.9349	0.9879	0.9693	0.0706	0.0	0.2521	0.3963	0.6011	0.8812	0.1741	0.0	0.0032	0.5581
whd	0.9606	0.9930	0.9253	0.1654	0.01	0.0596	0.1194	0.3221	0.4187	0.0719	0.0049	0.0	0.7065
all	0.8350	0.9600	0.9532	0.0527	0.0	0.2038	0.2285	0.3951	0.5686	0.2487	0.0069	0.0942	0.6200
RGB†	0.9232	0.9929	0.9524	0.0475	0.0	0.3060	0.2795	0.4418	0.9203	0.2077	0.0	0.0	0.6480
LAB	0.9615	0.9850	0.9722	0.0037	0.0	0.0951	0.1254	0.5605	0.8739	0.0753	0.0	0.0	0.5624
Full*	0.9349	0.9879	0.9693	0.0706	0.0	0.2521	0.3963	0.6011	0.8812	0.1741	0.0	0.0032	0.5581
O. lrng	0.9418	0.9919	0.9510	0.1288	0.03	0.1019	0.2626	0.1526	0.5943	0.1280	0.0009	0.0138	0.5953
O. SPG	0.8394	0.9960	0.9642	0.0649	0.0	0.1368	0.2403	0.5415	0.8914	0.3774	0.0057	0.0004	0.6186

Chapter 8

Conclusions

3D semantic segmentation is truly a major research topic in the computer vision field: this is testified by the many works and active researchers on the subject, the literature is huge and it is always growing with new ideas and new approaches, not restricted to the ones that involves the use of deep learning.

PointNet pioneered the deep learning way by introducing an architecture that directly consumes 3D point clouds without recurring to voxelization or other representations, and many works are based on this concept, like the Superpoint Graph architecture. It extracts a reduced graph from the whole point cloud and then applies small PointNets to subgraphs, in order to gather compact representations to be fed to recurrent networks that maintain the state and make predictions on points.

In parallel, traditional methods have also been perfected, and 3DEF maintained state of the art performance on datasets like S3DIS as proved in Chapter 6. Moreover, the use of entangled features has been demonstrated to enhance performance on accuracy by allowing spatial context to be included as a relevant part in learning, thus capturing more meaningful insights on data.

Even if with mixed outcomes, the extensive tests we made on both architectures confirm that there is no algorithm that can overwhelm the others: both deep networks and the 3DEF algorithm have their pro and cons, and they perform better in different and heterogeneous scenarios and setups; it clearly depends on the equipment that is planned to be used for performing the task of segmentation, for example whether a GPU is available or not, or whether there is the need of online computations on a mobile robot or the elaboration is completely offline: these will set the prevalent method to be used.

For 3DEF, it was demonstrated that entangled features play an important role but they were not extensively used as this work principally focused on unary features. Adding the specific set of features from SPG did not enhance the performance: this is not a symptom the algorithm cannot get better, but rather that the set of feature simply does not add valuable information in the learning phase. It was also proven

that currently the partition method in SPG is not a valid substitute for 3DEF's original clustering algorithm, built upon the PCL library. For SPG, adding height features showed marginal improvement over the original method, while using other types of features currently implemented in 3DEF or changing color space did not prove to be useful experiments.

Tests also shows that there is room for improvement on both algorithms, but choices have to be weighted by also considering the relative percentage gain in accuracy, and the changes in training, learning and computation times that are proportional to the complexity of the final network or to the number and complexity of used features, as revealed but not reported while working on this thesis. It was planned but not implemented due to lack of time to perform some tests to bring the entangling concept to SPG by leveraging points' distances to build clusters of points inside the 10-neighbors of a point P, and to compute on them features such as mean and standard deviation of color's channels, along with centroids' coordinates and distances from P. In our vision, these features would have been added to the original set to mimic the entangling process proper of 3DEF; to investigate this approach is one of the possible path to further develop research in this field.

List of Figures

List of Tables

Acronyms

3DEF 3D Entangled Forest.

AI Artificial Intelligence.

AR Augmented Reality.

BB bounding box.

CNN Convolutional Neural Network.

CRF Conditional Random Field.

CV Computer Vision.

DT Decision Tree.

ECC Edge-Conditioned Convolution.

EF Entangled Forest.

GPU Graphics Processing Unit.

GRU Gated Recurrent Unit.

LiDAR Light Detection and Ranging.

ML Machine Learning.

MLP Multi-layer Perceptron.

MT-PNet Multi-task Pointwise Network.

NN Neural Network.

RF Random Forest.

RNN Recurrent Neural Network.

S3DIS Stanford Large-scale 3D Indoor Spaces Dataset.

SOM Self-Organizing Map.

SPG Superpoint Graph.

TI Trilinear Interpolation.

VCCS Voxel Cloud Connectivity Segmentation.

VR Virtual Reality.

Bibliography

[1] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla, «Segmentation and Recognition Using Structure from Motion Point Clouds», in *European Conference on Computer Vision (ECCV)*, 2008, pp. 44–57.

[2] A. Martinovic, J. Knopp, H. Riemenschneider, and L. Van Gool, «3D All The Way: Semantic Segmentation of Urban Scenes From Start to End in 3D», in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015.

[3] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[4] F. Rosenblatt, «The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain», *Psychological Review*, pp. 65–386, 1958.

[5] G. Garrido and P. Joshi, *OpenCV 3.X with Python By Example*. Packt, 2018.

[6] H. Zhu, F. Meng, J. Cai, and S. Lu, «Beyond pixels: A comprehensive survey from bottom-up to semantic image segmentation and cosegmentation», *Journal of Visual Communication and Image Representation*, vol. 34, pp. 12–27, 2016. DOI: https://doi.org/10.1016/j.jvcir.2015.10.012.

[7] M. Thoma, «A Survey of Semantic Segmentation», 2016. arXiv: 1602.06541.

[8] H. Ajmal, S. Rehman, U. Farooq, Q. U. Ain, F. Riaz, and A. Hassan, «Convolutional neural network based image segmentation: a review», vol. 10649, 2018. DOI: https://doi.org/10.1117/12.2304711.

[9] M. Naseer, S. Khan, and F. Porikli, «Indoor Scene Understanding in 2.5/3D for Autonomous Agents: A Survey», *IEEE Access*, vol. 7, pp. 1859–1887, Jan. 2019. DOI: 10.1109/ACCESS.2018.2886133.

[10] M. Weinmann, B. Jutzi, S. Hinz, and C. Mallet, «Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers», *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 105, pp. 286–304, 2015. DOI: https://doi.org/10.1016/j.isprsjprs.2015.01.016.

[11] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. G. Rodríguez, «A Review on Deep Learning Techniques Applied to Semantic Segmentation», 2017. arXiv: 1704.06857.

[12] D. Maturana and S. Scherer, «VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition», in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2015, pp. 922–928.

[13] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller, «Multi-view convolutional neural networks for 3D shape recognition», in *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, «ImageNet Classification with Deep Convolutional Neural Networks», *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017. DOI: 10.1145/3065386.

[15] A. Boulch, B. L. Saux, and N. Audebert, «Unstructured Point Cloud Semantic Labeling Using Deep Segmentation Networks», in *Eurographics Workshop on 3D Object Retrieval*, The Eurographics Association, 2017. DOI: 10.2312/3dor.20171047.

[16] T. Hackel, J. D. Wegner, and K. Schindler, «Fast semantic segmentation of 3D point clouds with strongly varying density», *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, vol. 3, no. 3, 2016.

[17] H. Thomas, F. Goulette, J. Deschaud, and B. Marcotegui, «Semantic Classification of 3D Point Clouds with Multiscale Spherical Neighborhoods», in *International Conference on 3D Vision (3DV)*, Sep. 2018, pp. 390–398. DOI: 10.1109/3DV.2018.00052.

[18] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz, «Rotation Invariant Spherical Harmonic Representation of 3D Shape Descriptors», in *Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 2003, pp. 156–164.

[19] Y. Shen, C. Feng, Y. Yang, and D. Tian, «Mining Point Cloud Local Structures by Kernel Correlation and Graph Pooling», Jun. 2018, pp. 4548–4557. DOI: 10.1109/CVPR.2018.00478.

[20] S. Biasotti, A. Cerri, A. Bronstein, and M. Bronstein, «Recent Trends, Applications, and Perspectives in 3D Shape Similarity Assessment», *Computer Graphics Forum*, vol. 35, no. 6, pp. 87–119, 2016. DOI: 10.1111/cgf.12734.

[21] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, and J. Wan, «3D Object Recognition in Cluttered Scenes with Local Surface Features: A Survey», *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2270–2287, Nov. 2014. DOI: 10.1109/TPAMI.2014.2316828.

[22] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, «Volumetric and Multi-View CNNs for Object Classification on 3D Data», 2016. arXiv: 1604.03265.

[23] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, «PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation», in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 77–85. DOI: 10.1109/CVPR.2017.16. arXiv: 1612.00593.

[24] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, «PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space», in *Conference on Neural Information Processing Systems (NIPS)*, Dec. 2017, pp. 5099–5108. arXiv: 1706.02413.

[25] D. G. Lowe, «Distinctive Image Features from Scale-Invariant Keypoints», *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004. DOI: 10.1023/B:VISI.0000029664.99615.94.

[26] F. Engelmann, T. Kontogianni, A. Hermans, and B. Leibe, «Exploring Spatial Context for 3D Semantic Segmentation of Point Clouds», in *IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct. 2017.

[27] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, «Dynamic Graph CNN for Learning on Point Clouds», 2018. arXiv: 1801.07829.

[28] X. Ye, J. Li, H. Huang, L. Du, and X. Zhang, «3D Recurrent Neural Networks with Context Fusion for Point Cloud Semantic Segmentation», in *European Conference on Computer Vision (ECCV)*, Sep. 2018.

[29] L. Tchapmi, C. Choy, I. Armeni, J. Gwak, and S. Savarese, «SEGCloud: Semantic Segmentation of 3D Point Clouds», in *International Conference on 3D Vision (3DV)*, Oct. 2017, pp. 537–547. DOI: 10.1109/3DV.2017.00067.

[30] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr, «Conditional Random Fields as Recurrent Neural Networks», in *IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015, pp. 1529–1537. DOI: 10.1109/ICCV.2015.179.

[31] T. Kohonen, «The self-organizing map», *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, Sep. 1990. DOI: 10.1109/5.58325.

[32] J. Li, B. M. Chen, and G. Hee Lee, «SO-Net: Self-Organizing Network for Point Cloud Analysis», in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018.

[33] Z. Wu, S. Song, A. Khosla, X. Tang, and J. Xiao, «3D ShapeNets: A Deep Representation for Volumetric Shapes», 2014. arXiv: 1406.5670.

[34] L. Landrieu and M. Simonovsky, «Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs», vol. 1711.09869, 2017.

[35] Y. Xu, Z. Sun, L. Hoegner, U. Stilla, and W. Yao, «Instance Segmentation of Trees in Urban Areas from MLS Point Clouds Using Supervoxel Contexts and Graph-Based Optimization», in *IAPR Workshop on Pattern Recognition in Remote Sensing (PRRS)*, Aug. 2018, pp. 1–5. DOI: 10.1109/PRRS.2018.8486220.

[36] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, «PointCNN: Convolution On X-Transformed Points», in *Advances in Neural Information Processing Systems 31*, Curran Associates, Inc., 2018, pp. 820–830.

[37] Q.-H. Pham, D. T. Nguyen, B.-S. Hua, G. Roig, and S.-K. Yeung, «JSIS3D: Joint Semantic-Instance Segmentation of 3D Point Clouds with Multi-Task Pointwise Networks and Multi-Value Conditional Random Fields», in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[38] L. Breiman, «Random Forests», *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001. DOI: 10.1023/A:1010933404324.

[39] A. Montillo, J. Shotton, J. Winn, J. E. Iglesias, D. Metaxas, and A. Criminisi, «Entangled Decision Forests and Their Application for Semantic Segmentation of CT Images», in *Information Processing in Medical Imaging*, 2011, pp. 184–196.

[40] D. Wolf, J. Prankl, and M. Vincze, «Fast semantic segmentation of 3D point clouds using a dense CRF with learned parameters», in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 4867–4873. DOI: 10.1109/ICRA.2015.7139875.

[41] ——, «Enhancing Semantic Segmentation for Robotics: The Power of 3-D Entangled Forests», *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 49–56, Jan. 2016. DOI: 10.1109/LRA.2015.2506118.

[42] M. Antonello, D. Wolf, J. Prankl, S. Ghidoni, E. Menegatti, and M. Vincze, «Multi-View 3D Entangled Forest for Semantic Segmentation and Mapping», in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 1855–1862. DOI: 10.1109/ICRA.2018.8460837.

[43] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, «Vision meets Robotics: The KITTI Dataset», *International Journal of Robotics Research (IJRR)*, 2013.

[44] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, «The Cityscapes Dataset for Semantic Urban Scene Understanding», in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[45] A. Dai, A. X. Chang, M. Savva, M. Halber, T. A. Funkhouser, and M. Nießner, «ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes», 2017. arXiv: 1702.04405.

[46] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, «Indoor segmentation and support inference from RGBD images», in *European Conference on Computer Vision (ECCV)*, 2012, pp. 746–760. DOI: 10.1007/978-3-642-33715-4_54.

[47] S. Song, S. P. Lichtenberg, and J. Xiao, «SUN RGB-D: A RGB-D scene understanding benchmark suite», in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 567–576. DOI: 10.1109/CVPR.2015.7298655.

[48] B.-S. Hua, Q.-H. Pham, D. T. Nguyen, M.-K. Tran, L.-F. Yu, and S.-K. Yeung, «SceneNN: A Scene Meshes Dataset with aNNotations», in *International Conference on 3D Vision (3DV)*, 2016.

[49] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese, «Joint 2D-3D-Semantic Data for Indoor Scene Understanding», Feb. 2017. arXiv: 1702.01105.

[50] J. Xiao, A. Owens, and A. Torralba, «SUN3D: A Database of Big Spaces Reconstructed Using SfM and Object Labels», in *IEEE International Conference on Computer Vision (ICCV)*, Dec. 2013.

[51] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, «Matterport3D: Learning from RGB-D Data in Indoor Environments», in *International Conference on 3D Vision (3DV)*, 2017.

[52] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. K. Brilakis, M. Fischer, and S. Savarese, «3D Semantic Parsing of Large-Scale Indoor Spaces», in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1534–1543.

[53] O. Vinyals, S. Bengio, and M. Kudlur, «Order Matters: Sequence to sequence for sets», Nov. 2015. arXiv: 1511.06391.

[54] S. Ioffe and C. Szegedy, «Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift», in *International Conference on Machine Learning (ICML)*, vol. 37, 2015, pp. 448–456.

[55] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, «Dropout: A Simple Way to Prevent Neural Networks from Overfitting», *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[56] L. Landrieu and M. Boussaha, «Point Cloud Oversegmentation With Graph-Structured Deep Metric Learning», in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1904.02113, Jun. 2019.

[57] J. Demantké, C. Mallet, N. David, and B. Vallet, «Dimensionality Based Scale Selection in 3d LIDAR Point Clouds», *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, vol. XXXVIII-5/W12, pp. 97–102, Sep. 2011. DOI: 10.5194/isprsarchives-XXXVIII-5-W12-97-2011.

[58] S. Guinard and L. Landrieu, «Weakly supervised Segmentation-aided Classification of Urban Scenes from 3D LiDAR point clouds», *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, vol. XLII-1/W1, pp. 151–157, May 2017. DOI: 10.5194/isprs-archives-XLII-1-W1-151-2017.

[59] L. Landrieu and G. Obozinski, «Cut Pursuit: fast algorithms to learn piecewise constant functions on general weighted graphs», *SIAM Journal on Imaging Sciences*, vol. 10, no. 4, pp. 1724–1766, 2017. DOI: 10.1137/17M1113436.

[60] L. Landrieu, H. R. Raguet, B. Vallet, C. Mallet, and M. Weinmann, «A structured regularization framework for spatially smoothing semantic labelings of 3D point clouds», *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 132, pp. 102–118, Oct. 2017. DOI: 10.1016/j.isprsjprs.2017.08.010.

[61] M. Simonovsky and N. Komodakis, «Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs», in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 29–38. DOI: 10.1109/CVPR.2017.11.

[62] J. Papon, A. Abramov, M. Schoeler, and F. Worgotter, «Voxel Cloud Connectivity Segmentation - Supervoxels for Point Clouds», in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2013, pp. 2027–2034. DOI: 10.1109/CVPR.2013.264.

[63] M. Abadi *et al.*, «TensorFlow: A system for large-scale machine learning», in *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2016, pp. 265–283.

[64] A. Paszke *et al.*, «Automatic Differentiation in PyTorch», in *NIPS Autodiff Workshop*, 2017.

[65] D. Comaniciu and P. Meer, «Mean Shift: A Robust Approach Toward Feature Space Analysis», *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, May 2002. DOI: 10.1109/34.1000236.