remote sensing

MDPI

Review

Deep Learning on 3D Semantic Segmentation: A Detailed Review

Thodoris Betsas [1,*], Andreas Georgopoulos [1], Anastasios Doulamis [1] and Pierre Grussenmeyer [2]

[1] Laboratory of Photogrammetry, School of Rural, Surveying and Geoinformatics Engineering, National Technical University of Athens, 15772 Athens, Greece; drag@central.ntua.gr (A.G.); adoulam@cs.ntua.gr (A.D.)
[2] ICube Laboratory UMR 7357, CNRS, INSA Strasbourg, Université de Strasbourg, 67084 Strasbourg, France; pierre.grussenmeyer@insa-strasbourg.fr
* Correspondence: betsasth@mail.ntua.gr

Abstract: In this paper, an exhaustive review and comprehensive analysis of recent and former deep learning methods in 3D semantic segmentation (3DSS) is presented. In the related literature, the taxonomy scheme used for the classification of 3DSS deep learning methods is ambiguous. Based on the taxonomy schemes of nine existing review papers, a new taxonomy scheme for 3DSS deep learning methods is proposed, aiming to standardize it and improve the comparability and clarity across related studies. Furthermore, an extensive overview of the available 3DSS indoor and outdoor datasets is provided along with their links. The core part of this review is the detailed presentation of recent and former 3DSS deep learning methods and their classification using the proposed taxonomy scheme along with their GitHub repositories. Additionally, a brief but informative analysis of the evaluation metrics and loss functions used in 3DSS is included. Finally, a fruitful discussion of the examined 3DSS methods and datasets is presented to foster new research directions and applications in the field of 3DSS. In addition to this review, a GitHub repository is provided, including an initial classification of over 400 3DSS methods, using the proposed taxonomy scheme.

Keywords: 3D semantic segmentation; point clouds; deep learning; review

check for updates

Academic Editors: Jiaojiao Li and Shuying Li

Received: 18 November 2024
Revised: 18 December 2024
Accepted: 31 December 2024
Published: 16 January 2025

Citation: Betsas, T.; Georgopoulos, A.; Doulamis, A.; Grussenmeyer, P. Deep Learning on 3D Semantic Segmentation: A Detailed Review. *Remote Sens.* **2025**, *17*, 298. https://doi.org/10.3390/rs17020298

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Acquiring 3D information in the form of 3D point clouds, from various environments of the world, is becoming easier day to day due to the wealth of available sensors like Airborne (A) or Terrestrial (T) Laser Scanners (ALSs, TLSs), Light Detection And Ranging (LiDAR), Radio Detection and Ranging (RADAR), Digital Single-Lens Reflex cameras (DSLR), Sound Navigation and Ranging (SONAR), etc. Most of the available sensors produce 3D point clouds, which, before the post-processing steps, e.g., filtering, constitute their raw 3D information. In general, deep learning (DL) algorithms dominate several 2D tasks like classification, object detection, and semantic -instance and -panoptic segmentation. Meanwhile, the 3D point clouds have unique benefits in comparison to the 2D images, such as they capture the detailed geometry of the objects [1,2] and, intuitively, they are similar to the 3D world. However, the application of 2D DL methods directly into 3D space using 3D point clouds is not a straightforward process due to their properties.

In detail, the 3D point clouds are commonly unordered, unstructured, and irregular. More concretely, unordered means that the 3D points are not on a regular grid, like the pixels of the images, unstructured means that the 3D points do not carry the neighboring points' information, and finally, irregular means that some 3D point clouds contain regions

with different point densities, i.e., the 3D points are not evenly sampled across the different regions of the scene. Moreover, each point's neighborhood forms meaningful information about the characteristics of its region, i.e., the 3D points are not isolated. Also, the 3D point clouds are invariant under rigid transformation, i.e., rotating and translating the 3D points altogether do not alter the characteristics of the acquired scene. To conclude, we consider that there is terminology confusion in the meaning of the 3D point cloud properties among the 3D semantic segmentation (3DSS) papers, i.e., the meaning of the unordered, irregular, and unstructured properties; however, the actual properties remain the same.

In general, semantic segmentation (SS) is defined as the association of each element of the data under process with a meaningful label. In this regard, 2D semantic segmentation (2DSS) using images aims to assign a meaningful label to each pixel of the image, while 3DSS using point clouds aims to assign a meaningful label to each 3D point of the point cloud, etc. In fact, there is terminology confusion regarding 3DSS. To be more specific, 3DSS is commonly referred to as 3D point cloud classification, 3D point cloud per point classification, or 3D labeling. However, throughout this effort, the 3DSS terminology is preferred.

To sum up, the contributions of this effort are varied. First and foremost, Section 2 describes in detail and compares the taxonomy schemes of the 3DSS DL methods proposed by previous 3DSS review papers, resulting safely in a unified taxonomy scheme. Additionally, the general idea behind the selection of such categories, based on the examined 3DSS methods, is thoroughly described. Apart from the general categories of the 3DSS DL methods, the same analysis is conducted for their subcategories. Moreover, in Section 3, over 40 3DSS indoor and outdoor datasets are presented, including their links. In addition to the available datasets, the commonly used evaluation metrics of the 3DSS algorithms are presented in Section 4. Afterward, a detailed analysis of the collected 3DSS algorithms of each category and subcategory defined in Section 2 is presented in Section 5. Notably, there are also subcategories included for hybrid-based methods apart from the main categories, i.e., point-, dimensionality reduction-, discretization-, and graph-based methods. Then, a brief but informative analysis of 3DSS loss functions is presented in Section 6. Finally, in Section 7, a thorough analysis of the examined 3DSS algorithms and datasets is presented to foster new research directions and applications in the field of 3DSS. To the best of our knowledge, this effort is the first 3DSS review paper including a taxonomy scheme section, i.e., Section 2. Finally, a GitHub repository (Supp. Mat.: https://github.com/thobet/Deep-Learning-on-3D-Semantic-Segmentation-a-Detailed-Review, accessed on 30 December 2024), which includes an initial classification of over 400 3DSS algorithms, is included. In addition to this effort, the GitHub repository offers a valuable source for the investigation of both traditional [3–6] and deep learning 3DSS algorithms.

2. Taxonomy Scheme of Deep Learning 3D Semantic Segmentation Methods

In the last decade, many researchers published review and benchmark papers presenting the ongoing research on 3D semantic segmentation (3DSS), i.e., the open challenges and research questions, the difficulties presented in 3DSS applications, ideas for future work, and many 3DSS algorithms. In this section, we present a literature review of carefully selected [7–22] 3DSS review papers, and we aim to draw a safe conclusion about a taxonomy scheme of 3DSS deep learning methods. To be more specific, each review paper proposes a different categorization scheme of the existing 3DSS algorithms using different names for each category. But are those categories really different?

2.1. Previous 3DSS Review Papers

First and foremost, 3DSS algorithms could be divided into two broad categories: Regular Supervised Machine Learning Methods and Deep Learning Methods [18,19,22]. The

methods belonging to the first category are often called traditional methods, while the methods belonging to the second one are deep learning methods. The main difference between the two categories is the feature extraction approach. Traditional methods rely on handcrafted features, while deep learning approaches learn to extract features from the data without user involvement. Traditional methods contain a broad range of segmentation methods. An analysis of well-known traditional methods on the segmentation and classification of 3D point clouds can be found in Grilli E., Menna F., and Remondino F.'s (2017) [11] review paper. However, in this section, we focus on the deep learning methods for 3DSS because we aim to define their taxonomy scheme.

In general, deep learning methods for 3DSS using point clouds can be separated into four classes [8,12–14]:

- Point-based methods, which use the raw point cloud directly as input for the 3DSS algorithms.
- Projection-based methods, which project the point cloud onto a 2D grid, exploit mature 2D convolution techniques for 2DSS, and, finally, project the labels back into 3D space.
- Discretization-based methods, which transform the point cloud to a discrete representation on which the 3D convolution operator is applicable.
- Hybrid methods, which combine two or more techniques that belong to the previously described categories.

Hereafter, we refer to the previously described taxonomy as the main taxonomy scheme just for comparison purposes. In the following paragraph, we present a connection between the different category names included in each review paper with the main taxonomy scheme.

Regarding the main taxonomy scheme, the (i) Multi-View Convolution Neural Networks (CNNs) and Unordered Point Cloud Processing [10], (ii) multi-view-based [22], (iii) multi-view-based [18], (iv) multi-view [7], (v), image-based [9], (vi) RGB-D Image-based [23], (vii) dimensionality reduction [20], and (viii) multi-view [21] categories are subcategories of projection-based methods or include some projection-based methods along with others. Additionally, based on the main taxonomy scheme the (i) Volumetric, Unordered Point Cloud Processing and Ordered Point Cloud Processing [10], (ii) Volumetric Method [22], (iii) voxel-based [9,18,21,23], (iv) voxel and higher dimensional lattice [7], and (v) methods based on voxelization [20] categories are subcategories of discretization-based methods or include discretization-based methods along with others. Moreover, with respect to the main taxonomy scheme, the (i) Ordered Point Cloud Processing [10], (ii) Direct [22], (iii) Directly Process on Point Cloud Data [18], (iv) Deep Learning Directly on Raw Point Clouds [7], (v) point-based methods [9,23], (vi) primitive points. [20], and (vii) point cloud-based [21] categories are the same as point-based methods or include point-based methods along with others.

Bello et al. 2020 [7], Zhang et al. 2019 [22], and R. Zhang et al. 2023 [23] propose a more general taxonomy scheme for 3DSS deep learning methods, using only two broad categories:

- Indirect [22], Structured Grid-Based Learning [7], or Rule-Based methods [23];
- Direct [22], Deep Learning Directly on Raw Point Clouds [7], or point-based methods [23].

The authors used different names to describe the same category of methods. To be more specific, the first group of methods transform the given point cloud into a regular grid representation to apply the convolution operation on it, while the second group of methods use the point cloud directly as an input to the deep learning algorithms. Thus, the above scheme could be considered a superset of the main taxonomy scheme. Furthermore, R. Zhang et al. 2023 [23] include RGB-D Image-based methods in their taxonomy scheme.

This new subcategory is the core subcategory of projection-based methods in the main taxonomy scheme. We think that the name RGB-D Image-based methods is easily confused with the RGB-D methods presented in Griffiths and Boehm's 2019 review paper, which refer to the methods applied to data collected with RGB-D sensors, e.g., in indoor environments. Hence, projection-based methods seems to be a more relevant name for the category. Finally, A. Zhang et al. 2023 [20] include the Multiple Data Formats category, among others, and divide it into multi-representational methods and multimodal methods. In the main taxonomy scheme, multi-representational methods is the same category as hybrid methods, while the multimodal methods for 3DSS could be classified following the same taxonomy scheme as unimodal 3D semantic segmentation methods, i.e., point-based methods, discretization-based methods, etc. More concretely, multimodal 3DSS methods are classified with the same taxonomy scheme as unimodal 3DSS methods, i.e., multimodal methods are not treated as a subcategory of unimodal 3DSS methods but as an independent category of methods.

2.2. Proposed 3DSS Taxonomy Scheme

After the examination of the previously described 3DSS review papers, we found that the point-based methods of the main taxonomy scheme are usually included in categories like Direct Methods, Directly Process Point Cloud Data, Deep Learning Directly on raw Point Clouds, methods based on primitive points, etc., but without any confusion about the methods included in these categories, i.e., although the categories have a different name, they include almost the same methods. We chose the term point-based methods for our taxonomy scheme to classify these methods because we think that it describes better and compactly the general idea of the methods, i.e., to extract meaningful 3DSS features using as input the raw point cloud. Moreover, the graph-based methods are commonly included in point-based methods in the literature. In our taxonomy, graph-based methods are a separate category because the feature extraction process is based on the nodes and the edges of graphs, not only based on the 3D points as in point-based methods. Additionally, the graph-based methods transform the given point cloud into an alternative representation, i.e., a graph, and then perform the 3DSS.

Furthermore, we believe that the dimensionality reduction-based methods term introduced by A. Zhang et al., 2023 [20] describes better the methods included in the projection-based methods category in the main taxonomy scheme because projection methods can also be used to describe a subclass of methods that use different projections, e.g., bird's-eye view (BEV), cylindrical, polar, etc., to perform 3DSS. Thus, we adopt the dimensionality reduction-based methods term in our taxonomy scheme. Additionally, the discretization-based methods term describes better the methods that are commonly referred to as voxel, voxelization, etc., as well as those commonly referred to as permutohedral lattices, lattices, etc., because the main idea of these methods is the creation of a new discrete representation based on the 3D point cloud and then the use of 3D convolution in order to extract meaningful features for 3DSS. Finally, the hybrid methods category includes those methods which use a combination of techniques of the previously described categories.

To sum up, we use the following taxonomy scheme to classify 3DSS deep learning methods (Figure 1):

- Point-based methods (Section 5.1. Point-Based Methods).
  - Point-based methods use raw 3D points to extract meaningful features for 3DSS.
- Dimensionality reduction-based methods (Section 5.2. Dimensionality Reduction-Based Methods).

- Dimensionality reduction-based methods transform the 3D point cloud into a lower dimensional space, e.g., images, perform semantic segmentation into that space, and, finally, project the labels back into 3D space.

- Discretization-based methods (Section 5.3. Discretization-Based Methods).
  - Discretization-based methods transform the point cloud into a discrete representation without dimensionality reduction and then apply the convolution operation for 3DSS.
- Graph-based methods (Section 5.4. Graph-Based Methods).
  - Graph-based methods transform the point cloud into a graph and use the nodes and the edges of the graph to extract meaningful features for 3DSS.
- Hybrid methods (Section 5.5. Hybrid Methods).
  - Hybrid methods combine two or more methods from the previously described categories.

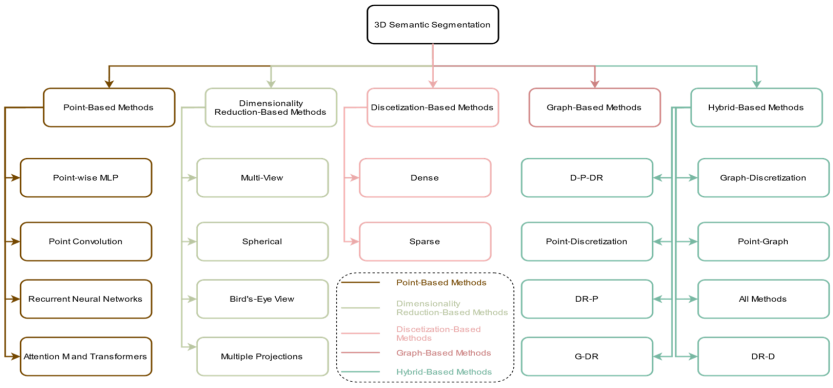


Figure 1. The proposed taxonomy scheme of the deep learning 3D semantic segmentation methods. Abbreviations: attention mechanism and transformers (Attention M. and Transformers), discretization-based methods (D), dimensionality reduction-based methods (DR), graph-based methods (G), point-based methods (P).

2.3. Point-Based Methods Taxonomy Scheme

Point-based methods use raw 3D points directly to extract meaningful features for 3DSS. In this subsection, the taxonomy scheme for the point-based methods is defined. Firstly, the point-wise MLP category contains the point-based 3DSS methods that apply a per-point process using several shared Multi-Layer Perceptrons (MLPs) to extract local features and then extract the global information using a symmetrical aggregation function [7–9,12,14,23]. In general, convolution is the fundamental operation of a broad category of 2D deep learning networks, i.e., CNNs, with remarkable results. However, the implementation of convolution operation using 3D point clouds is not a straightforward process due to their characteristics. The methods that investigate the implementation of convolution operation directly onto 3D point clouds are included in the Point Convolution category. Guo et al. (2021) [12] and Camuffo, Mari, and Milani (2022) [8] subdivide the convolution-based methods used directly onto 3D point clouds into continuous and discrete ones based on the space that is applied. Gao et al. (2022) [9], Zhang et al. (2023) [20], Zhang et al. (2023) [23], Jhaldiyal and Chaudhary (2023) [13], and Rauch and Braml (2023) [14] include the 3D convolution-based methods as a subcategory of point-based methods but without a further categorization. Thirdly, recurrent neural networks (RNNs) treat the given data, e.g., image, 3D point cloud, etc., as sequences of features. Furthermore, RNN-based methods aim to extract the contextual information between the data elements, i.e., pixel, 3D points, etc., by recalling the features gathered earlier in the sequence or by forgetting some of them [8,9,12,14,21,23]. The 3DSS methods that use RNNs are classified into the recurrent neural networks category. Finally, the 3DSS methods that use the trans-

former architecture and attention mechanism are included in the attention mechanism and transformers category [8,14,20,21,23].

2.4. Dimensionality Reduction-Based Methods Taxonomy Scheme

Dimensionality reduction methods transform the 3D point cloud into lower dimensional space data, e.g., images, apply semantic segmentation techniques using those data, and, finally, project and fuse the labels back into 3D space. In this subsection, the subcategories of dimensionality reduction methods are defined based on the projection used for the dimensionality reduction of the 3D point clouds. To be more specific, dimensionality reduction methods are further classified into the multi-view, spherical, and bird's-eye view (BEV) and Multiple Projections categories. Multi-view methods replicate the photography of objects, scenes, etc. To be more specific, they use the 3D data as the scene and then capture many images artificially around the data using a set of predefined viewpoints. Finally, the created images are used for 2DSS. Furthermore, spherical and BEV methods use spherical and BEV projections, respectively, to create the images that are finally fed into the 2DSS process [7–10,12–14,18,20–22]. Moreover, the Multiple Projection category includes the methods that combine different projections to perform 3DSS. In general, the created images are called range images when they are produced using LiDAR data without the assigned colors from a camera. Finally, point cloud serialization methods, which are commonly used in combination with discretization techniques, transform the given point cloud into a lower dimension regular structure that is furthered processed for 3DSS and so could be considered as a dimensionality reduction technique.

2.5. Discretization-Based Methods Taxonomy Scheme

Discretization-based methods transform the 3D point cloud into a discrete representation without dimensionality reduction and then apply the convolution operation for 3DSS. The main idea of these methods is to handle the unorder property of 3D point clouds by transforming the 3D point cloud into an appropriate representation for convolution operation in 3D space. In this subsection, the subcategories of discretization-based methods are defined. Most 3DSS review papers [9,10,18,19,21,22] focus on voxel-based methods. However, other review papers classify the discretization-based methods into the dense and sparse subcategories, while some of them also use a subcategory for the methods that transform the 3D point cloud into a higher dimensional lattice to finally perform 3DSS [7,8,12,14,20]. However, the proposed taxonomy scheme includes the methods that transform the 3D point cloud into a higher dimensional lattice before 3DSS into the sparse subcategory. Dense discretization methods ignore the distribution of points and discretize the entire space of the 3D point cloud using a 3D grid of a specific size. Sparse discretization methods take into account the distribution of 3D points and discretize only the occupied space, resulting in more efficient algorithms with respect to execution time. In this effort, only the sparse methods are presented because they dominate the category nowadays.

2.6. Graph-Based Methods Taxonomy Scheme

Graph-based methods, transform the point cloud into a graph and use the nodes and the edges of the graph to extract meaningful features for 3DSS. In this category, there is no further categorization of the methods proposed. Graph-based methods are commonly included as a subcategory of point-based methods. However, the proposed taxonomy scheme defines them as a separate class because they transform the point cloud into a new representation, i.e., a graph, and are not applied directly to the initial 3D point cloud.

2.7. Hybrid Methods Taxonomy Scheme

When a developed algorithm uses two or more techniques of the previously described categories, it is classified into the hybrid methods category. In this subsection, a further categorization of the hybrid methods is proposed. In fact, the proposed taxonomy scheme for the hybrid methods takes into account the techniques that are combined to create the developed algorithm, resulting in eight subcategories: all methods, discretization–point–reduction, graph–discretization, graph–reduction, point–discretization, point–graph, reduction–discretization, and reduction–point. In fact, there are more than eight possible categories, but we define those eight based on the examined papers.

3. Three-Dimensional Semantic Segmentation Datasets

In general, the benchmark datasets play a significant role in the proper investigation of the deep learning algorithms. The scope of this section is to provide a useful guide of the available 3DSS datasets along with their characteristics and their links. In 3DSS, the available datasets can be divided into indoor and outdoor ones. In Tables 1 and 2, general information about each indoor and outdoor 3DSS datasets is provided, respectively, along with their links.

Table 1. The indoor 3D semantic segmentation datasets.

Dataset Name	Year	Images	Mesh	Link
2D-3D-S S3DIS [24,25]	2017	√	√	https://redivis.com/datasets/9q3m-9w5pa1a2h
Freiburg Campus [26]	-	-	-	http://ais.informatik.uni-freiburg.de/projects/datasets/fr360/
Matterport3D [27]	2017	√	√	https://niessner.github.io/Matterport/#paper
ScanNet [28]	2018	-	-	http://www.scan-net.org/
RGB-D Scenes Dataset v2 [29]	2013	-	-	https://rgbd-dataset.cs.washington.edu/dataset/rgbd-scenes-v2/
SceneNet [30]	2017	-	√	https://robotvault.bitbucket.io/scenenet-rgbd.html
SceneNN [31]	2016	-	√	https://hkust-vgd.github.io/scenenn/
SUN RGB-D [32]	2015	√	×	https://rgbd.cs.princeton.edu/
SUN3D [33]	2013	-	×	https://sun3d.cs.princeton.edu/
The Replica Dataset [34]	2019	-	√	https://github.com/facebookresearch/Replica-Dataset
VIDRILO [35]	2015	√	-	https://www.rovit.ua.es/dataset/vidrilo/index.html

√: Available; ×: Not Available; -: No Data All links last accessed: 30 December 2024.

Table 2. The outdoor 3D semantic segmentation datasets.

Dataset Name	Date	Synthetic	Spatial Size	Resolution	Mesh	Links
Freiburg, Pittsburgh, Wachtberg [36]	2012	×	-	-	-	http://ais.informatik.uni-freiburg.de/projects/datasets/fr360/
Swiss3DCities [37]	2020	×	≈2.7 × 10^6 m^2	0.013 m	×	https://zenodo.org/records/4390295 https://zenodo.org/records/4390295
Paris-CARLA-3D [38]	2021	√	≈550 m (R) ≈5.8 km (S)	-	×	https://npm3d.fr/paris-carla-3d
LiDAR-CS [39]	2023	√	-	-	×	https://github.com/LiDAR-Perception/LiDAR-CS
SMARS [40]	2023	√	2 Cities	0.30–0.50 m	√	https://www2.isprs.org/commissions/comm1/wg8/benchmark_smars/
SEMANTIC3D [41]	2017	×	-	-	×	http://www.semantic3d.net/ https://github.com/nsavinov/semantic3dnet
SensatUrban [42]	2021	×	≈7.6 km^2	-	×	https://github.com/QingyongHu/SensatUrban
SWAN [43]	2023	×	≈150 km	-	×	https://ieee-dataport.org/documents/swan-3d-point-cloud-dataset
RELLIS–3D [44]	2022	×	-	-	×	https://github.com/unmannedlab/RELLIS-3D
DAPS3D [45]	2023	√	-	-	√	https://github.com/subake/DAPS3D
Hessigheim 3D (H3D) [46]	2021	×	-	800 pts/m^2	√	https://ifpwww.ifp.uni-stuttgart.de/benchmark/hessigheim/default.aspx
Campus3D [47]	2020	×	≈1.58 km^2	-	×	https://github.com/shinke-li/Campus3D
Learnable Earth Parser [48]	2023	×	≈7.7 km^2	-	×	https://romainloiseau.fr/learnable-earth-parser/
OpenGF [49]	2021	×	≈47 km^2	-	×	https://github.com/Nathan-UW/OpenGF
Paris-Lille-3D [50]	2018	×	≈2 km	-	×	https://npm3d.fr/paris-lille-3d
DLA-Net [51]	2021	×	≈3 km	-	×	https://github.com/suyanfei/DLA-Net
Toronto 3D [52]	2020	×	≈1 km	-	×	https://github.com/WeikaiTan/Toronto-3D
Min3D [53]	2023	×	≈0.63 km^2	-	×	https://3dom.fbk.eu/benchmarks
MSNet [54]	2018	×	-	5–10 pts/m^2	×	https://github.com/wleigithub/WHU_pointcloud_dataset

Table 2. *Cont.*

Dataset Name	Date	Synthetic	Spatial Size	Resolution	Mesh	Links
N3C-California [55]	2023	×	≈725 km^2	≥8 pts/m^2, 1 m	×	https://github.com/wymqqq/IKDNet-pytorch?tab=readme-ov-file
DFC 2018 [56]	2019	×	≈4.3 km^2	1 m, 0.05 m	×	https://ieee-dataport.org/open-access/2018-ieee-grss-data-fusion-challenge-%E2%80%93-fusion-multispectral-lidar-and-hyperspectral-data
DublinCity [57]	2019	×	≈2 km^2	≈299 pts/m^2 0.034 m	×	https://v-sense.scss.tcd.ie/DublinCity/
ArCH3D [58]	2020	×	-	-	×	https://archdataset.polito.it/
TerraMobilita/iQmulus [59]	2015	×	10 km	-	×	http://data.ign.fr/benchmarks/UrbanAnalysis/index.html
KITTI-360 [60]	2023	×	73.7 km	-	×	https://www.cvlibs.net/datasets/kitti-360/index.php
nuScenes [61]	2020	×	-	-	×	https://www.nuscenes.org/
Oakland3D [62,63]	2009	×	-	-	×	http://www.cs.cmu.edu/~vmr/datasets/oakland_3d/cvpr09/doc/
Paris-rue-Madame [64]	2014	×	0.16 km	-	×	https://people.cmm.minesparis.psl.eu/users/serna/rueMadameDataset.html
Waymo [65]	2019	×	76 km^2	-	×	https://waymo.com/open/
TUM-City-Campus [66]	2020	×	0.2 km^2	-	×	https://www.pf.bgu.tum.de/en/pub/tst.html
SemanticKITTI [67]	2019	×	73.7 km	-	×	http://www.semantic-kitti.org/

√: Available; ×: Not Available; -: No Data; All links last accessed: 30 December 2024.

Most of the indoor 3DSS datasets use an RGB-D sensor to generate 3D point clouds [24,25,27–35] predominantly in real environments. Also, there are other indoor 3DSS datasets that use Laser Range sensors [26] or are produced using a synthetic environment [30]. Commonly used sensors to create 3D point clouds in indoor environments are Matterport and Real Sense cameras created by Matterport inc. and Intel Corporation respectively, Microsoft Kinect, ASUS Xtion sensors, and SICK LMS LiDAR. In general, each indoor dataset provides both similar and different information compared to the others; for instance, the presence of normal vectors [24,25,34], 2D and 3D annotations [24,25,27,30–33], or camera poses [27,33]. Finally, in Figure 2, the number of the available classes, images, and scans are provided for each dataset.

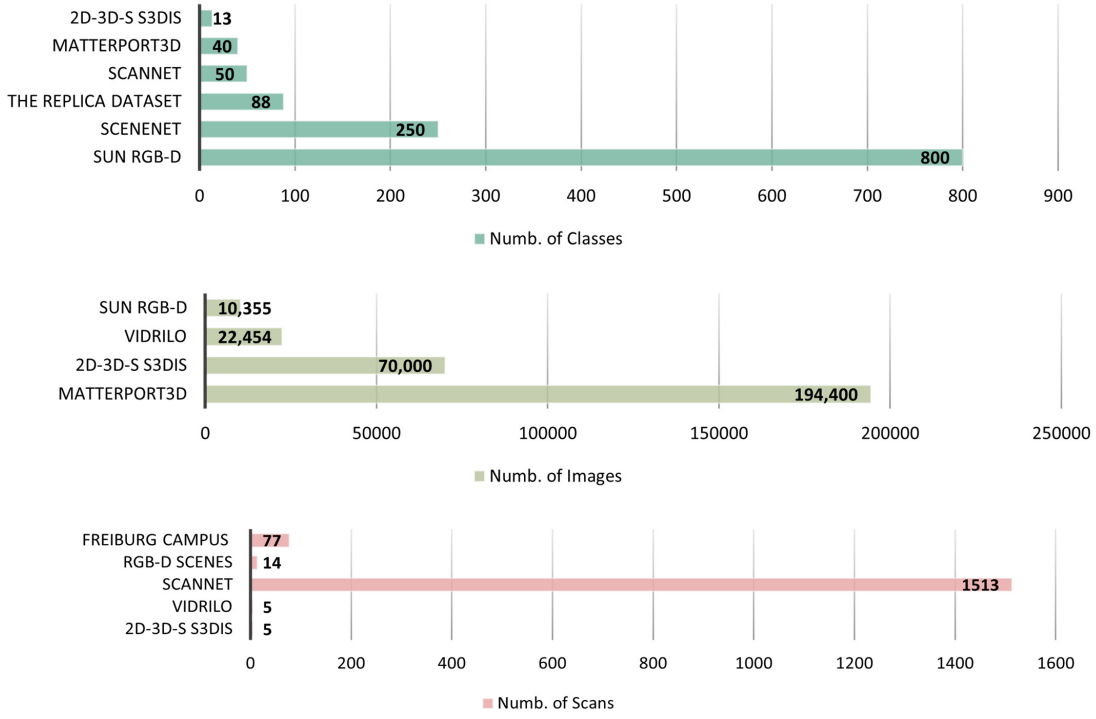


Figure 2. The number of classes, images, and scans provided in each indoor 3DSS dataset. The relevant references for each dataset can be found in Table 1.

After a careful examination of the available outdoor 3DSS datasets, it was concluded that a large variety of sensors is used to map the outdoor 3D environment providing raw 3D point clouds. Specifically, there are datasets created either using terrestrial, aerial, or mobile laser scanner–LiDAR [36,38,39,41,43–46,48–67], photogrammetry [37,40,42,44,47], or even using mobile range finders [36]. For instance, several datasets have been created using Velodyne LiDAR [36,38,39,50,53,60,61,64,66,67], Oyster LiDAR [43–45], Riegl LiDAR [46,51], DSLR cameras [46,57,58], and Unmanned Aerial Vehicles (UAVs), along with their built-in cameras [37,42,47,58], etc. In general, each outdoor dataset provides both similar and different information compared to the others. For example, providing data of the same area but with different resolution (500 k, 15 M, 223 M points) [37], using a tilted LiDAR acquisition process [38], capturing multi-temporal [46] or multimodal data [40], and data for a specific application such as the classification of cultural heritage monuments [58] or the semantic segmentation of facades [51]. Additionally, each photogrammetric dataset has a different Ground Sampling Distance (GSD), and each LiDAR dataset has a different ratio of pts/m^2, both of which play a significant role in each application (Table 2). In Figure 3, the number of the available classes, images, and scans are displayed for each dataset. Finally, in Figure 4, the statistics about the usage of the indoor and outdoor datasets is presented based on the examined SoTA methods in Section 5.

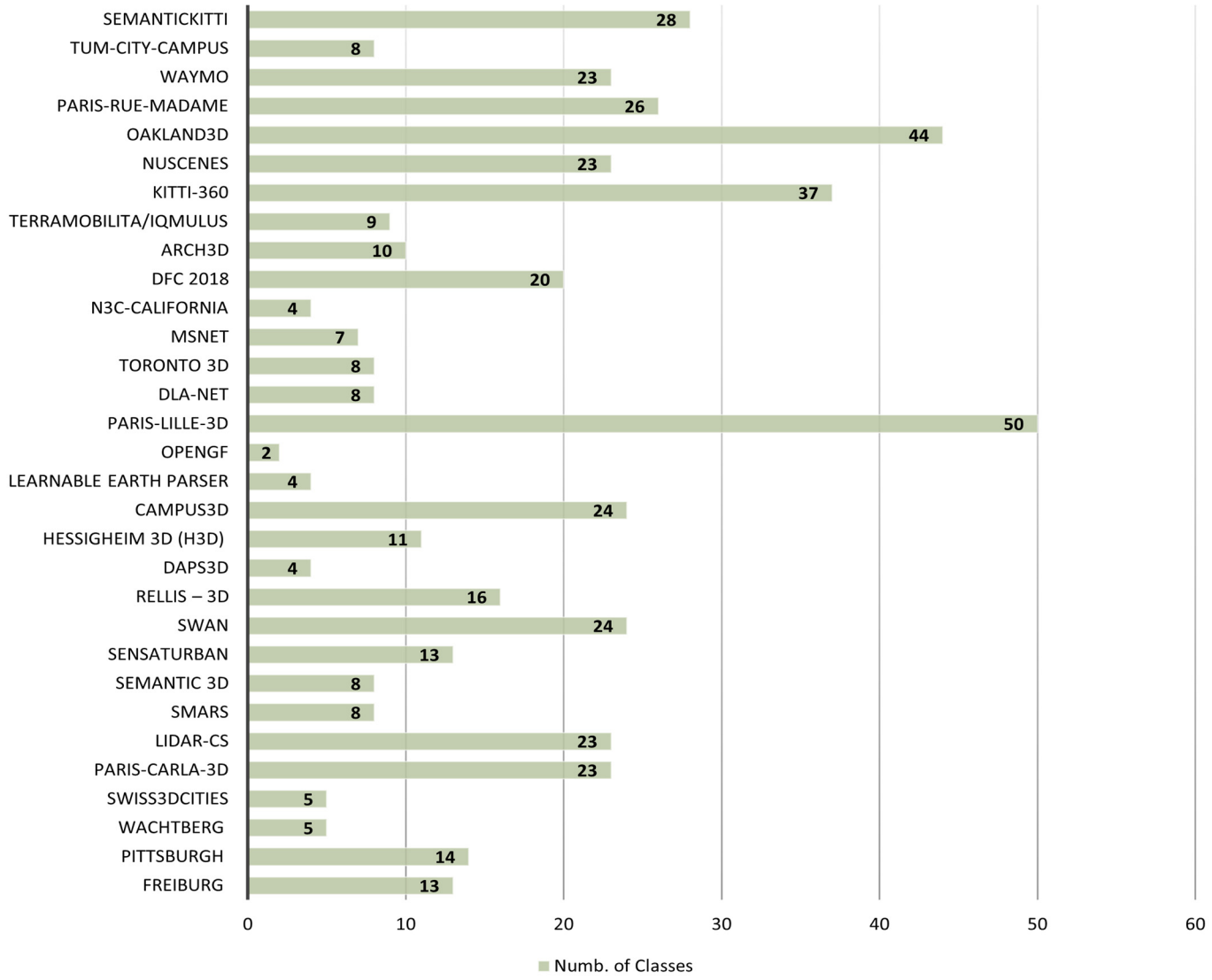


Figure 3. *Cont.*

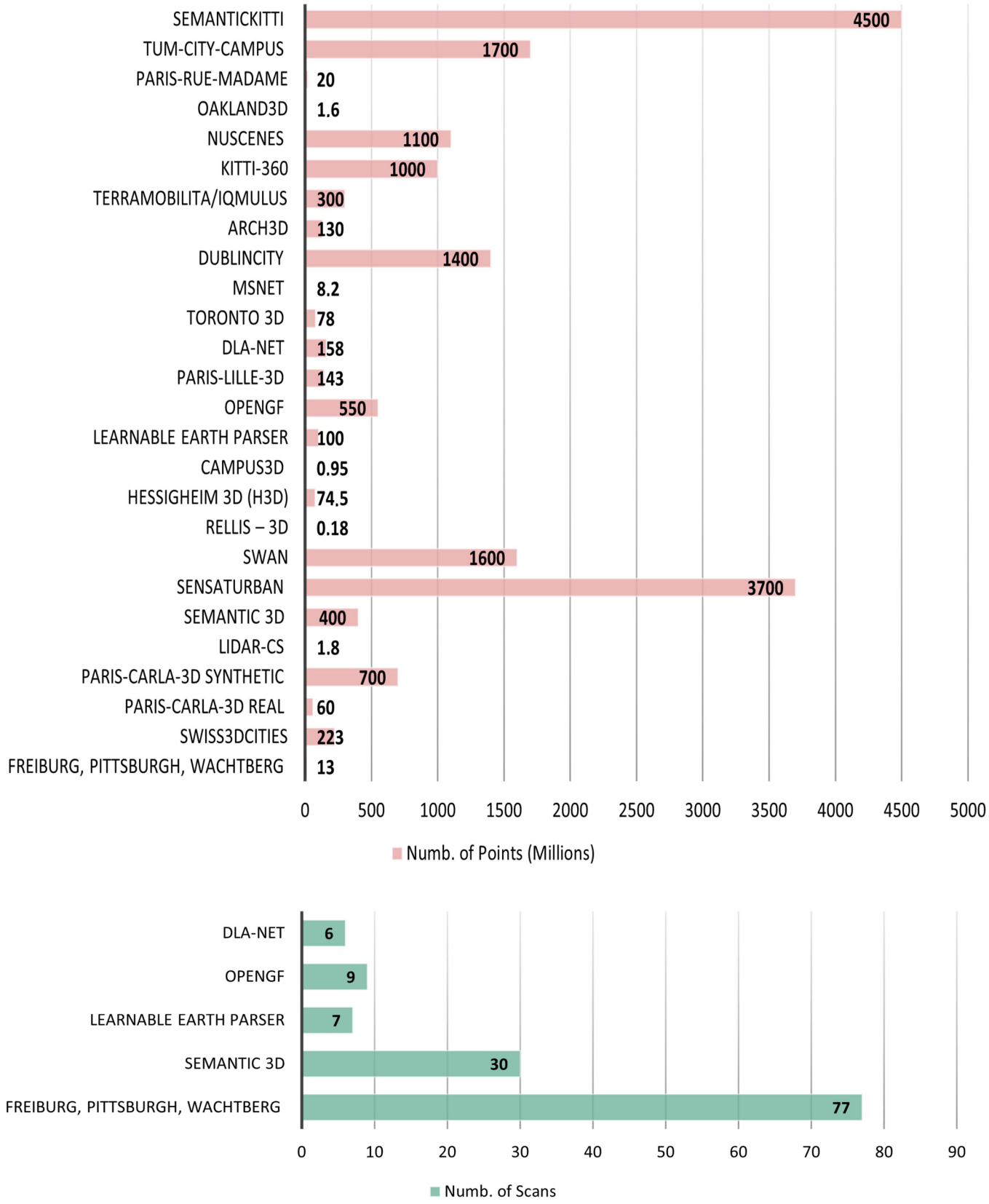


Figure 3. The number of classes, points, and scans of the outdoor 3D semantic segmentation datasets. The relevant references for each dataset can be found in Table 2.

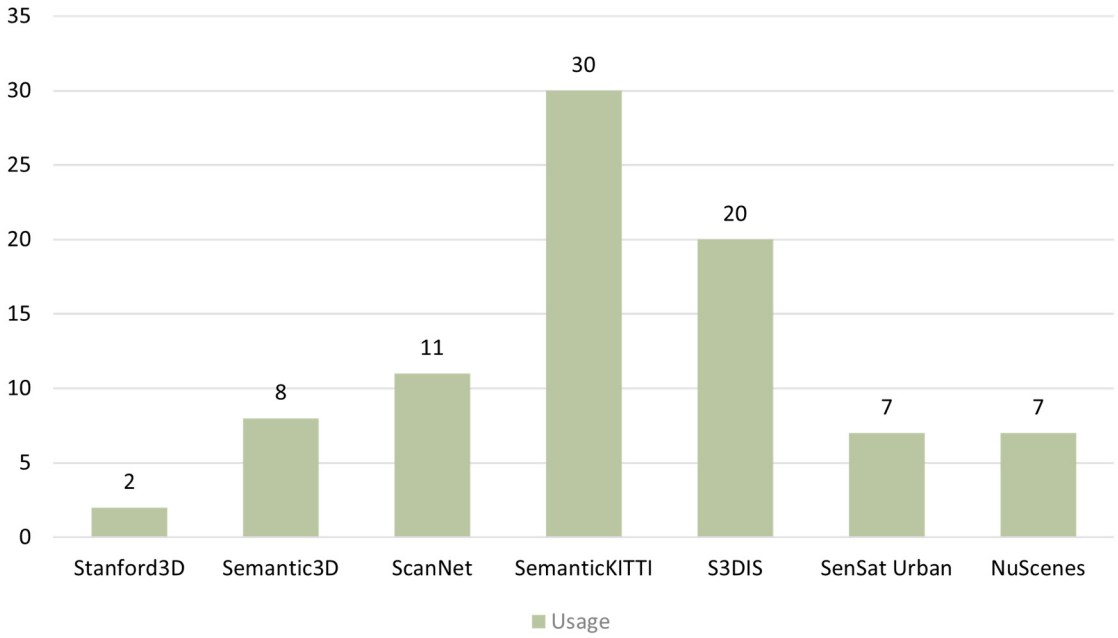


Figure 4. Statistics of datasets usage based on the SoTA methods examination in Section 5. The relevant references for each dataset can be found in Table 1 and Table 2.

4. Evaluation Metrics in 3D Semantic Segmentation

In general, the performance of the developed algorithms is examined using different metrics regarding the under-investigation task. Basically, every benchmark dataset provides a set of test data and evaluation metrics for comparison purposes. Every upcoming method that uses the benchmark is evaluated using the data and the metrics. In fact, there are several metrics like accuracy, recall, precision, and F1-Score. In 2D and 3D semantic segmentation, the most common metric is Intersection over Union (IoU), which better assesses the performance of the model than the others. In this section, the commonly used evaluation metrics are described. But firstly, the confusion matrix should be examined.

4.1. Confusion Matrix

The confusion matrix is the basis on the evaluation of an algorithm using test data. It is created measuring the True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) for each class. In fact, such metrics are further used for the calculation of the evaluation metrics presented earlier. Commonly, the measured metrics, i.e., TP, TN, etc., are visualized using the confusion matrix. An example of a confusion matrix of two classes is presented below.

Comparing the predicted values with the actual values, the TP, FP, FN, and TN values could be derived forming the confusion matrix. While in Figure 5, a 2×2 confusion matrix is presented, most of the time, there are more than two classes, e.g., n, and hence, the confusion matrix could be an $n \times n$ matrix regarding the number of them. In Figure 6, an $n \times n$ confusion matrix is presented. The TP, TN, FP, and FN values can be derived by examining the $n \times n$ confusion matrix for each class. For instance, for Class 3, the TP are defined by the value included in cell (3,3), the FP are defined by adding the values of the third column, excluding cell (3,3), the FN are defined by adding the values of the third row, excluding the cell (3,3), and, finally, the TN are defined by subtracting the TP, FP and FN from the total instances.

		Actual Values	
		Positive	Negative
Predicted Values	Positive	TP	FP
	Negative	FN	TN

Figure 5. A 2 × 2 confusion matrix. Negative predictions (red); positive predictions (green).

		Predicted Classes						
		Class 1	Class 2	Class 3	Class 4	Class 5	…	Class n
True Classes	Class 1							
	Class 2							
	Class 3							
	Class 4							
	Class 5							
	…							
	Class n							

Figure 6. An n × n confusion matrix. Negative predictions (red); positive predictions (green).

4.2. Evaluation Metrics

In general, the most straightforward metric is accuracy, i.e., the ratio between the correctly predicted values divided by the total number of values. However, there are different accuracy metrics presented in the literature, i.e., the Overall Accuracy (OAcc), mean class accuracy (mcAcc), and mean average accuracy (maAcc). More concretely, the OAcc or Acc, is the predefined ratio for the test set (Equation (1)), the mcAcc is the mean accuracy of a class for a specific number of iterations (Equation (2)), and the maAcc is the mean accuracy of all the classes divided by the number of classes (Equation (3)). In fact, there is confusion between the different accuracy metrics, especially for multi-class experiments. Hence, the reader should always examine the formulas used in any case.

$$\text{OAcc} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$\text{m}_\text{c}\text{Acc} = \frac{1}{\#n}\sum_n \text{OAcc} \tag{2}$$

$$\text{m}_\text{a}\text{Acc} = \frac{1}{\#c}\sum_c \text{OAcc} \tag{3}$$

where

- TP, TN, FP, FN: True Positive, True Negative, False Positive and False Negative;
- $\#n$: Number of iterations;
- n: Iteration;
- $\#c$: Number of classes;
- C: Class.

In fact, accuracy gives informative results when it is calculated using balanced data regarding the classes. Hence, there are more evaluation metrics derived using the confusion matrix, i.e., precision, recall, and F1-score. Intuitively, precision measures how vulnerable the model is in positive predictions or, in other words, the quality of the positive predictions; recall measures how prone the model is to correctly perform a positive prediction; and the F1-score is the harmonic mean between precision and recall. Another terminology about recall is the True Positive rate or sensitivity.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5}$$

$$\text{F1} - \text{score} = 2\frac{Precision \times Recall}{Precision + Recall} \tag{6}$$

In most cases, the IoU metric is calculated to assess the performance of 2D and 3D semantic segmentation models. There are different IoU metrics like the mean IoU (mIoU) and the frequency-weighted IoU. A different terminology of the IoU is the Jaccard Index. Intuitively, the IoU quantifies the degree of overlap between the predicted and the actual class, e.g., of the masks. Moreover, the frequency-weighted IoU considers the amount of data in each class to handle the class imbalance problem, e.g., in an urban area, there will be fewer points representing the class bicycle than those representing the class street. In fact, the class imbalance problem is a crucial challenge for ML-DL applications, while the commonly used metrics, i.e., accuracy, precision, recall, and IoU, are affected by the imbalance among the semantic classes. Finally, using the confusion matrix, the IoU of each class can be calculated as follows.

$$\text{IoU} = \frac{TP}{TP + FP + FN} \tag{7}$$

$$\text{mIoU} = \frac{1}{\#c}\sum_c IoU \tag{8}$$

5. Three-Dimensional Semantic Segmentation

5.1. Point-Based Methods

5.1.1. Point-Wise MLP

Up to the presentation of the PointNet [68] architecture, the extraction of 3D point cloud features was performed manually. PointNet [68] was the first architecture that enabled the extraction of deep learning features directly using 3D point clouds. To be more specific, the input of the PointNet architecture was a point cloud with n $\times$ 3 dimensions. Firstly, the authors introduced the T-Net, which learns to transform the given point cloud, e.g., rotate, translate, and change its scale. Secondly, the transformed point cloud was fed into multiple shared Multi-Layer Perceptrons (MLPs), resulting in a point cloud with n $\times$ 64, i.e., local features, dimensions that finally were transformed to n $\times$ 1024 dimensions, i.e., global features. Thirdly, the max-pooling operation was applied on the n $\times$ 1024 point cloud, selecting the maximum value between the 1024 values. The idea was that the maximum value was independent to the values order; thus, it handles the unorder characteristic of point clouds. However, PointNet has several limitations, such as its inability to exploit the information of the local structures in point clouds. Hence, Qi et al. (2017) [69] proposed an extension of PointNet called PointNet++. The main contribution of PointNet++ was the proposed hierarchical structure that decodes the information of point clouds using three abstraction levels. The first one was implemented by the Sampling Layer, which finds the centroids of local regions by subsampling the given point cloud using the Farthest Point Sampling (FPS) algorithm. The second one was implemented by the Grouping Layer, which creates point neighborhoods using the Ball Query algorithm. Each point in the neighborhood was defined with relative coordinates to the neighborhood centroid, achieving the extraction of the neighborhood's local characteristics. Finally, the PointNet++ architecture, presented in Figure 7, used several mini-PointNet layers to extract meaningful features from the point sets. PointNet++ used several abstraction levels for the feature extraction, i.e., on multiple scales, and thus achieved the extraction of both fine-grained and global features, resulting in an improved version of PointNet.

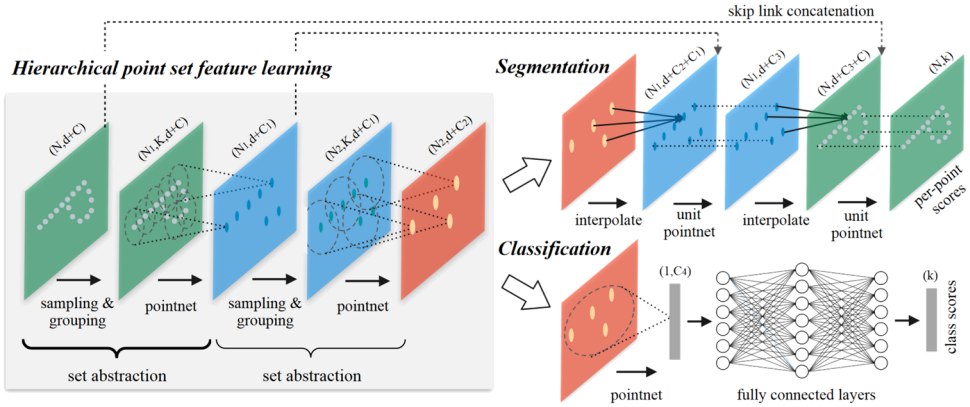


Figure 7. The PointNet++ architecture [69].

Since the introduction of PointNet and PointNet++, several methods have been proposed to improve the extraction of richer local information from point clouds, especially with computationally efficient approaches such as the RandLA-Net [70] architecture. Specifically, Hu et al. (2020) [70] tried to expand the application of point-wise MLP methods from small point clouds, e.g., thousands of points, to the large ones, e.g., millions of points, without using pre/post-processing steps and by replacing the computationally heavy sampling methods with lighter ones. To be more specific, Hu et al. (2020) stated that the majority of methods presented so far used complicated sampling methods, categorized into (i) Heuristic Sampling methods (Farthest Point Sampling (FPS), Inverse Density Importance Sampling (IDIS), and Random Sampling (RS)) and (ii) learning-based sampling methods (Generator-based Sampling (GS), Continuous Relaxation-based Sampling (CRS), and Policy Gradient-based Sampling (PGS)). Furthermore, the existing local feature learners were based on point cloud transformations, e.g., voxelization and graphs, and they did not decompose the complicated local information of point clouds of highly detailed objects, resulting in time-consuming approaches with low performance using such data. The authors proposed the local feature aggregation (LFA) module accompanied by Random Sampling to overcome the aforementioned limitations. More precisely, the LFA module included the Local Spatial Encoding (LocSE) and the attentive pooling (AP) submodules. The LocSE submodule firstly found the k-nearest neighbors for each center point using the Euclidean distance, then defined the relative point positions of the neighboring points to the center point, and, finally, concatenated the features of the neighboring points with the relative positions to create an augmented feature vector. The AP submodule was introduced on behalf of max/mean pooling approaches. It first received the created feature vector and computed attention scores using several shared MLPs and SoftMax. Finally, the important features were selected based on a weighted summation approach. In addition to the aforementioned submodules, the authors presented the Dilated Residual Block (DRB), which consists of multiple LocSE, AP, and skip connections in order to collect information using multiple receptive fields, preserving the local details of points' neighborhoods. In summary, the RandLA-Net architecture was constructed using several LFA modules and Random Sampling and was evaluated on 3D semantic segmentation benchmarks, achieving state-of-the-art results.

Fan et al. (2021) [1] proposed the SCF module, aiming to provide a robust method for the 3D semantic segmentation of large point clouds. The SCF module was composed of three main blocks: Local Polar Representation (LPR), Dual-Distance Attentive Pooling (DDAP), and Global Contextual Feature (GCF). More precisely, LPR aimed to find a representation of the local neighborhood of points that was invariant to Z-axis rotation. In fact, the objects belonging to the same category, e.g., chairs in a scene like an office, were pre-

sented with different orientations, resulting in 3DSS features that were orientation-sensitive. Thus, using LPR, the neighborhood of each point was represented using polar instead of Cartesian coordinates. More precisely, the points' initial local representation was defined using the Euclidean distance. Then, the local direction of each point in the neighborhood was extracted by calculating and then updating two angles: φ and θ. Specifically, φ was defined among each 3D point in the neighborhood of the under-process 3D point, the 3D point, and the X-axis in the XY-plane. Moreover, θ was defined among each 3D point in the neighborhood of the under-process 3D point, the 3D point, and the Z-axis in the XY-plane. θ and φ were then updated based on the barycenter point of each neighborhood. The LPR module answered the research question of how to represent the local context of a 3D point cloud. Afterward, the DDAP module was proposed to learn local context features using the LPR representation of points' neighborhoods. To be more specific, the DDAP module used the geometric distance in the world space, the feature distance in the feature space, and geometric patterns. The feature distance was the mean value of the subtraction between the features of the central point and the features of the under-process neighboring point. Furthermore, the geometric patterns were created by concatenating the LPR representation with the original coordinates and feeding them into shared MLP layers. Afterward, the MLP output was fed into SoftMax, forming the attentive pooling layer, which was learnable. Finally, the local contextual features were gathered using a weighted sum operation between the neighboring point features and the output of the attentive pooling layer. The DDAP module answered the research question of how to learn local contextual features. Apart from local features, 3DSS methods also need global features to achieve high-end results. To this end, the authors proposed the GCF module, which combined the (i) local x, y, and z coordinates with (ii) a ratio, defined as the number of points in the local neighborhood divided by the number of points in the global neighborhood, using MLP. The GCF module answered the research question of how to learn global contextual features. Finally, the SCF architecture was defined using multiple SCF modules in an encoder–decoder structure.

Qiu, Anwar, and Barnes (2021) [71] proposed a 3DSS network aiming to solve several limitations of SoTA approaches. To be more specific, the authors stated that SoTA methods were time-consuming or they created intermediate representations like graphs or voxels, which cause a partial loss of information. To this end, they proposed a point-based method that aimed to remedy three major drawbacks: (i) ambiguity in close points, (ii) redundant features, and (iii) inadequate global representations. Firstly, ambiguity in close points refers to the limitations that occur due to the neighborhood selection process, like overlapping regions or outlier selection. To alleviate the aforementioned disadvantage, the authors proposed an augmentation of the points' neighborhood. Secondly, redundant features refer to the models that use the same features multiple times to increase their performance. The authors stated that this redundant information increased the complexity of the models rather than improving their performance. Hence, they proposed to categorize the model's input, e.g., geometric or semantic, in order to fully utilize it. Finally, inadequate global representations refer to the limitations of SoTA feature maps with regard to 3DSS. Specifically, the authors stated that SoTA encoder–decoder methods decreased the details of the created feature maps, resulting in difficulties for 3DSS. Thus, the authors proposed a multi-resolution approach in order to preserve the fine-grained details through the model. The aforementioned improvements proposed by the authors were included in the Bilateral Context Module and the Adaptive Fusion Module. Firstly, the Bilateral Context Module was decomposed into Bilateral Augmentation, Augmentation Loss, and Mixed Local Aggregation. In the Bilateral Augmentation unit, the neighborhood of each centroid was defined using the kNN algorithm with 3D Euclidean distance. Then, the local

context (geometric and semantic) of each centroid was defined by combining its absolute coordinates with the relative of the centroid coordinates for each neighboring point. The local context was augmented by using bilateral offsets, which were guided by the Augmentation Loss using penalties during the learning process. Finally, the Mixed Local Aggregation unit was used to construct a neighborhood representation that precisely demonstrated the local distinctness of neighborhoods. Secondly, the Adaptive Fusion Module was used to extract feature maps for 3DSS using multi-resolution features. Specifically, several point clouds with descending resolutions were processed using the Bilateral Context Module, extracting multiple scales of information. Then, each feature map was gradually upsampled to the original representation, forming an upsampled set of feature maps that were finally fused adaptively at the point level. Overall, the authors proposed the BAAF-Net, which included the aforementioned modules, units, and losses and was evaluated using different benchmarks, metrics, and training schemes.

5.1.2. Point Convolution

Li et al. (2018) [72] proposed the PointCNN architecture, aiming to handle the irregularity and unorder characteristics of point clouds for 3D learning applications. First and foremost, the authors provided a detailed analysis of the limitations regarding the extension of conventional 2D CNN to 3D space, which was the main research question of Point Convolution methods. Thus, they proposed to extend the application of the typical CNN to 3DSS features rather than the 3D point cloud. PointCNN aimed to weight and permute the extracted features a predefined number of times and then to use the convolution operation on them. To be more specific, PointCNN was fed with a set of 3D points, along with their corresponding features. The X-Conv operator was proposed by the authors to be the main unit of the PointCNN architecture. It was used to create a new representation based on the input features, which had more depth in features and fewer points, e.g., it was a more abstracted and informative representation of the data than the first one. X-Conv for 3DSS exploited the furthest point sampling on an unorder local region of points, which were first transformed from the global to a local coordinate system. Receptive field was crucial for 2D CNNs, contributing seriously to the performance of 2DSS algorithms. PointCNN defined the receptive field in 3DCNN by calculating a ratio (K/N) using the number of neighboring points of a 3D point in the current (K) and previous layer (N). To conclude, PointCNN was an encoder–decoder architecture that used the X-Conv operator in both the encoder and decoder part. The main difference was that in the encoder phase, the created representations had fewer points and richer features, while in the decoder phase, they were the opposite, with the addition of skip connections.

Thomas et al. (2019) [73] presented the Kernel Point Convolution (KPConv) architecture, which was inspired from the typical 2D convolution operator. A comparison between image convolution and KPConv convolution is presented in Figure 8. Specifically, KPConv weights were carried by the 3D points in Euclidean space in the same manner as the features. In general, the rigid KPConv operator was applied on the 3D points close to the convolution location. To be more specific, the neighborhood of the convolution location was defined using a specific radius. The KPConv operator took as an input the neighborhood of points with coordinates relative to the convolution location. The convolution kernel was defined by points that were linked with their weights and aimed to transform the features of the given neighborhood points to a new set of features. Specifically, the transformation of feature dimensions was performed using a correlation function between the input and the kernel points. The authors used the linear correlation function. Additionally, they presented a comparison between the Gaussian correlation and the linear correlation to support their choice. Moreover, the authors proposed a deformable version of the KPConv

operator, which considered the local geometry of the points by applying different shifts at the convolution location. In fact, the convolution location was critical for the proposed operator. Furthermore, the authors included grid subsampling to handle the varying density issue of some 3D point clouds, explained the pooling layer and the KPConv layer, and then analyzed the selection of the network parameters. Finally, the authors proposed two architectures. The first one was about classification, while the second one was about segmentation. To be more specific, the segmentation architecture included the classification architecture in the encoder part, while the decoder part applied nearest upsampling in combination with skip connections between the encoder and decoder features. To conclude, the proposed KPConv architectures were evaluated on different benchmarks for classification and segmentation purposes.

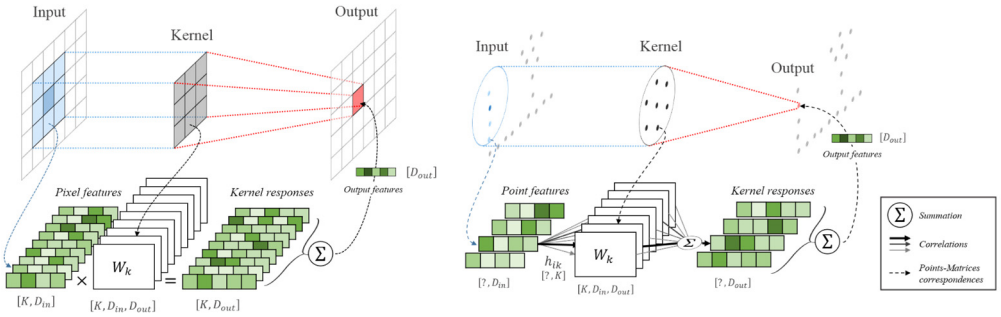


Figure 8. Comparison of image convolution and KPConv convolution in 2D space [73].

Liu et al. (2021) [74] introduced the FG-Net, a general point-based 3D deep learning architecture that can be used in different downstream applications like 3DSS or 3D classification. First and foremost, the authors presented several limitations of SoTA methods. For example, the SoTA methods contained time-consuming operations, like Farthest Point Sampling, or intermediate representations, such as graphs or voxels, they could not be applied to large-scale point clouds, and they had difficulty understanding detailed geometry. To address the limitations of the existing methods, the authors proposed the FG-Net architecture, in which the core module was called FG-Conv. Specifically, the FG-Conv module was composed of three main parts: point-wise correlated feature mining (PFM), Geometric Convolutional Modeling (GCM), and Attentional Aggregation (AG). Before the PFM, the authors proposed a Noise and Outlier Filtering (NOF) approach. To be more specific, the neighborhood of each point was determined based on a radius NN ball query algorithm. Afterward, an estimation of a point's neighborhood density was calculated. If the estimated density was lower than a threshold, the point was characterized as isolated and was deleted. Otherwise, the neighborhood of points was modeled as a Gaussian Distribution (μ, σ). Finally, neighborhood points were removed if the mean distance was not in the confidence interval of the Gaussian Distribution. After NOF, the point cloud was represented using the 3D coordinates along with the associated features like normals, colors, and learnt latent features. Using the filtered point cloud and knowing the neighborhood of each point, PFM defined a similarity score, in both 3D and features space, between the center point and each neighboring point by calculating the inner product between them. Afterward, the calculated similarity vector was passed into a learnt attention mechanism to transform the similarity scores based on the applied downstream 3D task, e.g., 3DSS. The new similarity scores were multiplied element-wise, with the 3D points resulting in the augmented attentional feature matrix, which was concatenated with the original features. PFM aimed to construct a feature space in which the relevant features were enhanced while the irrelevant features were attenuated. Moreover, GCM mimicked the deformable convolution from 2D space to 3D point cloud data and aimed to model the

geometric structure of points. Firstly, a correlation function was defined to calculate the association between the convolution kernel points and the local geometry, i.e., the points' neighborhood in a local coordinate system. The correlation function aimed to overcome the unstructured and unorder characteristics of 3D point clouds by increasing their value as the kernel points were closer to the local geometry. Then, the kernel function was defined as the sum of the correlation values with learnable weights and applied to capture the local geometry. Finally, the Attentional Aggregation component was defined to decrease the information loss when using the geometric and feature patterns by finding the meaningful features and aggregating them. The FG-Net was an encoder–decoder-based architecture that leveraged multi-resolution point clouds. To be more specific, the encoder was based on residual learning blocks (RLB) inspired by ResNet [75]. In fact, the FG-Conv operator was the core of the RLB. Between the encoder and the decoder, there was Point Clouds Global Feature Extraction, which aimed to capture the global dependencies in point clouds using the Nonlocal Attentive Module. Finally, they proposed a learning sampling approach called IGSAM, using the advantages of inverse density (IDS) and Gumbel SoftMax sampling instead of time-consuming point sampling methods. To conclude, the FG-Net was evaluated over different benchmarks for 3D classification and 3DSS using the mIoU metric. Additionally, the authors evaluated the performance of different models in 3DSS regarding their sampling technique, as well as the computation and memory consumption.

5.1.3. Recurrent Neural Networks

Huang, Wang, and Neumann (2018) [76] observed that the local information incorporated into 3D point clouds was not sufficiently exploited by SoTA methods. Additionally, they stated that SoTA methods spent a wealth of time on the computation of local dependencies. Thus, they proposed the RSNets, a series of a Slice Pooling Layer (SPL), recurrent neural network (RNN) Layers, and a Slice Unpooling Layer (SUL). The aforementioned components were incorporated into the Local Dependency Module. Firstly, the underprocess 3D point cloud was fed into the Input Feature Block (IFB). More concretely, the IFB transformed an unordered 3D point cloud into a set of unordered features. The unordered features were then input into the SPLs. More specifically, there were three SPL layers, one for each direction, i.e., x, y, and z, called slices. Each slice contained a set of 3D points. Afterward, a global feature vector for each slice was produced by aggregating the slice points' features. Finally, a set of an ordered sequence of features was created, which was the input of the RNN layer. In fact, the SPL created a representation of the features that can be exploited by the RNN bidirectional layers. The output features of the RNN layers were fed into the SUL to assign them to each point. To conclude, the RSNets architecture was evaluated using different large-scale point cloud benchmarks after a thorough analysis of ablation studies and experiments.

5.1.4. Attention Mechanism and Transformers

In general, the attention mechanism included in the transformer architecture is promising for 3DSS, as it is independent of 3D point cloud characteristics such as irregularity and disorder [14,23]. Based on the SoTA performance of transformers on different tasks [77], Zhao et al. (2021) [78] stated that the self-attention mechanism appears particularly relevant to be used in 3D tasks and proposed the Point Transformer architecture (Figure 9). First and foremost, the authors stated that the self-attention mechanism can be categorized into scalar and vector attention. In fact, the difference between scalar and vector attention is the creation of scalar and vector scores, respectively. Thus, the vector attention mechanism was selected for the Point Transformer architecture due to its ability to capture more detailed information about neighboring 3D points [78]. The vector attention equation components

were the three point-wise feature transformations (φ, ψ, α) like MLPs or linear projections, the position encoding function δ, a normalization function ρ like SoftMax, the relation function β, and the mapping function γ. To be more specific, the vector attention mechanism of the proposed Point Transformer Layer (PTL) was applied to a local neighborhood of points defined using the kNN algorithm at a specific location, with subtraction as the relation function; an MLP, two linear layers, and ReLU nonlinearity as the mapping function; and position encoding δ into both functions γ and α. Furthermore, positional encoding was a very important component of the transformer architecture since it was the counterpart of convolution and recurrence operations [77]. Specifically, in Point Transformer, the positional encoding function was a trainable function based on the 3D point coordinates. Basically, positional encoding was an MLP with two linear layers and ReLU nonlinearity aimed to define the neighboring points' intra-relationships. The aforementioned units consist of the Point Transformer Layer, which was the core of the Point Transformer Block. In fact, the Point Transformer architecture was created in encoder–decoder fashion, with skip connections between the encoder layer and their corresponding decoder layers. More concretely, the encoder was constructed using the Point Transformer Block and the Transition Down Layer, while the decoder was constructed using the Point Transformer Block and the Transition Up Layer. The Transition Down and Transition Up aimed to find a subset and superset of the input points, respectively. In 3DSS, the Output Head mapped the decoder output to the predicted class using an MLP. To conclude, the Point Transformer architecture was evaluated using different experiments in multiple tasks like 3DSS, classification, and different metrics.

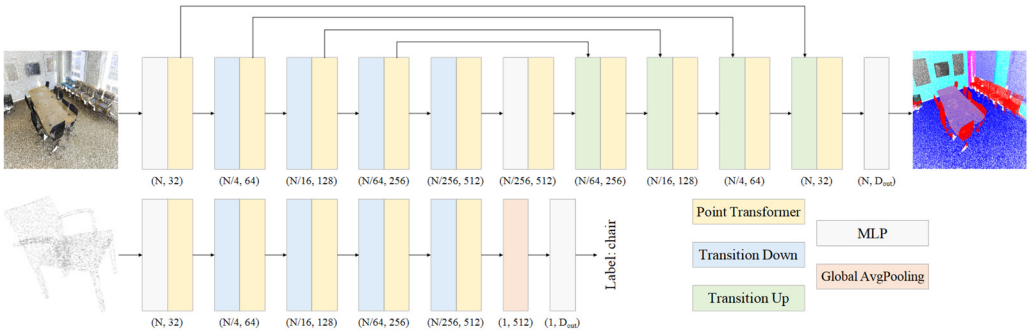


Figure 9. The Point Transformer architecture [78]. Point Transformer layer (yellow), Transition Down (blue), Transition Up (green), MLP (Grey) and Global Average Pooling (Pink).

Wu et al. (2022) [79] stated that the Point Transformer architecture increases the number of channels and the weight encoding parameters as it goes deeper, resulting in overfitting and restricting the model to go deeper. To overcome the aforementioned problem, the authors introduced a new attention mechanism called Group Attention (GA) instead of the vector attention that was used in the Point Transformer architecture. GA was characterized as a more general case of the vector aimed to reduce overfitting and enhance the generalization of the model. The authors stated that neighborhood attention performs better than shifted-grid attention. To be more specific, gathering the local neighborhood of points using the kNN function outperforms the methods that construct the local neighborhood using uniform non-overlapping cells due to the different point density of 3D point clouds. Additionally, the authors tried to better reveal the 3D point relationships by adding an additional positional encoding mechanism aiming to fully exploit the geometric knowledge encapsulated into the 3D point coordinates. In general, the traditional-based pooling procedures did not consider point density and overlapping. Hence, they proposed an improved pooling operation using uniform grid partitioning to replace the commonly used pooling approaches such as FPS or Ball Query. Furthermore, the authors presented

extended experiments and ablation studies to strengthen the advantages of the proposed architecture. To conclude, Point Transformer V2 was evaluated using different benchmarks and metrics under different 3D tasks like 3DSS and 3D Shape Classification. Finally, the performance of the presented point-based methods in different benchmark datasets is presented in Table 3.

Table 3. Mean Intersection over Union (mIoU) and Overall Accuracy (OA) of different benchmark datasets for point-based methods based on the papers examined in Section 5.1.

Algorithm	Year	Stanford3D		Semantic3D		ScanNet		SemanticKITTI		S3DIS		SenSat Urban	
		mIoU	OA	mIoU	OA	mIoU	OA	mIoU	OA	mIoU	OA	mIoU	OA
PointNet	2017	47.71	78.62	14.6	-	55.7	52.6	14.6	-	47.6	78.6	23.71	80.78
PointNet++	2017	-	-	63.1	85.7		84.5	20.1	-	54.5	81.0	39.97	84.30
RandLA-Net	2020	-	-	77.4	94.8	64.5	-	53.9	88.8	70.0	88.0	52.69	89.78
SCF-Net	2021	-	-	-	-	-	-	-	-	71.6	88.4	-	-
BAAF-Net	2021	-	-	75.4	94.9	-	-	59.9	-	72.2	88.9	-	-
PointCNN	2018	-	-	-	-	45.8	-	-	-	65.4	88.1	-	-
KPConv	2019	-	-	74.6	92.9	68.6	-	58.8	90.3	70.6	-	57.58	93.20
FG-Net	2021	-	-	-	-	-	-	-	-	70.8	-	-	-
RSNets	2018	-	-	-	-	-	76.5	-	-	56.5	85.7	-	-
PTv1	2021	-	-	-	-	-		-	-	73.5	90.2	-	-
PTv2	2022	-	-	-	-	-	75.2	-	-	71.6	91.1	-	-

-: No Data.

5.2. Dimensionality Reduction-Based Methods

5.2.1. Multi-View

Tatarchenko et al. (2018) [80] proposed tangent convolution in order to perform 3DSS using point clouds with millions of points. In fact, the tangent convolution approach was an extreme case of the multi-view CNNs proposed by Su et al. (2015) [81]. Tangent convolution was proposed for dense prediction tasks instead of the shape recognition task that multi-view CNNs remedy. Tangent convolution can be used on different types of 3D data, e.g., mesh, point clouds, and polygon soup, with the only constraint being the ability to estimate the normal vectors using them. In general, the authors stated that the 3D data, captured using different sensors, represent 2D structures embedded in 3D space, and thus, tangent convolution was based on the concept that the data were drawn from local Euclidean surfaces. To be more specific, tangent convolution first defined a tangent plane around every point and projected the local geometry on it, creating a tangent image, e.g., the tangent plane was exploited as an orthogonal to the point's neighborhood, virtual camera, i.e., along its normal vector. Moreover, for each 3D point, the orientation of the tangent plane was derived by covariance analysis of its local neighborhood. The local neighborhood was defined using a radius around the under-process point. Afterward, the covariance matrix of the neighborhood was estimated, resulting in the estimation of the neighborhood normal vector, i.e., the eigenvector of the smallest eigenvalue, and the X-axis and Y-axis of the tangent plane, i.e., the other two eigenvectors. However, to create virtual images using the tangent planes, the point signals must be used to estimate the image signals. The neighborhood points were projected onto the tangent plane of the under-process 3D points, resulting in a set of projected points. Hence, the projected points were a sampling of the continuous image space, and so, the authors investigated different interpolation approaches, e.g., nearest neighbor, full Gaussian mixture, or Gaussian mixture with the top three neighbors to formulate the final tangent images. They stated that the sophisticated interpolation methods did not result in a significant improvement in the tangent images, and thus, they proposed using simple nearest neighbor interpolation. In practice, the continuous space was replaced by a discrete space, i.e., a grid, resulting in the

tangent images used to perform semantic segmentation. Furthermore, the authors proposed a UNet-like network for the semantic segmentation of the tangent images. First, they defined the core network operations, e.g., pooling and unpooling. To be more specific, the pooling operation was defined by hashing the 3D points onto a progressively coarser 3D grid with a predefined grid resolution, using modular arithmetic on individual point coordinates. More precisely, the points that were hashed together in the 3D grid were used to pool their signal. Hashing deals with the irregularity property of 3D point clouds. Unpooling was performed by reusing the hash indices of the pooling operation. Additionally, they proposed Local Distance Features as the mean distance between the neighborhood points and the tangent plane. The Local Distance Features were used to create distance images that were exploited as an additional channel of the data. The proposed encoder–decoder architecture had two pooling layers and two unpooling layers using 3×3 kernels followed by Leaky ReLU with skip connections, while the last layer exploited 1×1 convolutions to assign the final classes. The pixel size of the tangent images along with the radius used to define the points neighborhood were used to define the receptive field of the convolutional layers. To conclude, the authors evaluated their algorithm using different benchmark datasets and metrics.

5.2.2. Spherical

Wu et al. (2017) [82] introduced the SqueezeSeg architecture, which utilized 3D-2D spherical projection in combination with mature 2D semantic segmentation techniques to achieve real-time, high-end results in the 3DSS of road objects. First and foremost, the authors described the core steps of the existing 3DSS SoTA approaches as Ground Filtering, point grouping, hand-crafted feature extraction for each group and group classification steps. However, they mentioned that the existing ground removal approaches were characterized by poor generalization and time-consuming post-processing steps, or they relied on iterative algorithms like RANSAC [83], which depended on the quality of the random initialization. Additionally, the multi-stage existing methods included limitations, e.g., error aggregation phenomena. Thus, the authors proposed a dimensionality reduction learning algorithm that was based on a combination of 2D convolutional neural networks (CNNs) and conditional random fields (CRFs). The general idea was to transform the input LiDAR point cloud into a compact representation, feasible for the convolution operation to extract the semantic labels and then to be refined using a CRF. In fact, the 3D point clouds' properties made it difficult to apply convolution directly on them. Hence, the authors proposed projecting the input 3D point cloud onto a sphere to create a dense 2D grid base, similar to the ordinary image's representation. The created spherical projection had five features for each point, i.e., 3D Cartesian coordinates, intensity, and range. Moreover, the proposed network structure was inspired by the SqueezeNet [84] 2D architecture, which investigated the reduction of the AlexNet [85] parameters while preserving the performance of it. The SqueezeSeg encoder–decoder architecture was based on the fireModules and fireDeconvs layers instead of the traditional convolution and deconvolution layers, aiming to reduce the parameters of the model. Specifically, the fireModule input was a spherical-projected 3D point cloud. Then, the spherical image was fed into a 1×1 convolution, resulting in a reduced size of the feature channels. Afterward, a parallel application of 3×3 and 1×1 convolutions were applied to recover the channel's size. The first layer was called the squeeze layer, while the second one was called the expand layer. The fireDeconvs were the same as the fireModules but with a deconvolution layer in between the squeeze layer and the expand layer. In general, 2DSS label maps suffer from blurry regions, especially between different classes, due to the downsampling operation. Thus, the authors introduced a CRF to refine the produced label maps. More concretely, they proposed an energy

function that has a unary potential term and a binary potential term, along with other terms. The latter was introduced as a punishment for labeling similar points with different labels. More precisely, the binary potential term was defined using two Gaussian kernels: the former used both the spherical and Cartesian coordinates and the latter used only the spherical coordinates. Additionally, the Gaussian kernels contained a set of empirical parameters. Overall, the labels' refinement process was performed by trying to minimize the aforementioned energy function using the mean-field iteration algorithm defined as an RNN. The refined labels were finally transferred into 3D space, resulting in the real-time 3DSS of road objects. To conclude, the SqueezeSeg architecture was trained using both real and synthetic LiDAR data and evaluated using different metrics.

However, Wu et al. (2018) [86] stated that the SqueezeSeg algorithm needed an improvement regarding its accuracy in order to be applied in real-world scenarios, while the manual creation of 3D training data for 3DSS was an extremely tedious process. To this end, Wu et al. (2018) [86] proposed the SqueezeSegV2 algorithm (Figure 10) to firstly improve the performance of SqueezeSeg and, secondly, to investigate the improvement in the synthetic training data creation. To be more specific, the authors stated that the accuracy degradation was mainly due to the dropout noise of real LiDAR data caused by several circumstances like a limited sensing range or mirror reflection. To alleviate the accuracy degradation, the authors proposed the Context Aggregation Module (CAM), a CNN module that leverages larger receptive fields in order to aggregate contextual information and thus to be robust against missing points. Additionally, the authors changed the cross-entropy loss with the focal loss to handle the imbalanced distribution of point categories throughout the LiDAR point cloud. Furthermore, the authors enriched the created spherical images with an extra channel indicating if a pixel was missing or existing, which improved the segmentation accuracy. Additionally, batch normalization was included in the SqueezeSegV2 architecture to handle the internal covariate shift phenomenon. In addition to the accuracy improvement, the authors tried to deal with the domain shift problem. To be more specific, domain shift referred to the phenomenon in which the NN was trained into a different domain than the applied domain, i.e., the source domain was different from the target domain, and so, the generalization was poor. To alleviate the domain shift problem, SqueezeSegV2 synthetic data rendered an intensity channel in addition to the rest by training a neural network in a self-supervised manner, taking point coordinates and depth as an input and outputting the intensity values, aiming to mimic real-world data. Additionally, they proposed geodesic correlation alignment and progressive domain calibration to reduce the gap between the source and the target domain. Specifically, in each training step, the SqueezeSegV2 network was fed with both synthetic and real data. Furthermore, they computed the focal loss on the synthetic batch to capture the semantic information and the geodesic distance of the output distributions of the synthetic batch and the real batch and reduce discrepancies between the statistics of the source and the target domain. In progressive domain calibration, each layer of the network was calibrated progressively, from the first to the last, without the previous layer impacting the others. To be more specific, for each layer, output statistics were calculated. Afterward, the output mean was re-normalized to 0, and the standard deviation to 1, and, in parallel, the batch normalization parameters were updated with the new statistics. Overall, the new components were evaluated by the authors with ablation studies, resulting in better performance than the SqueezeSeg architecture.

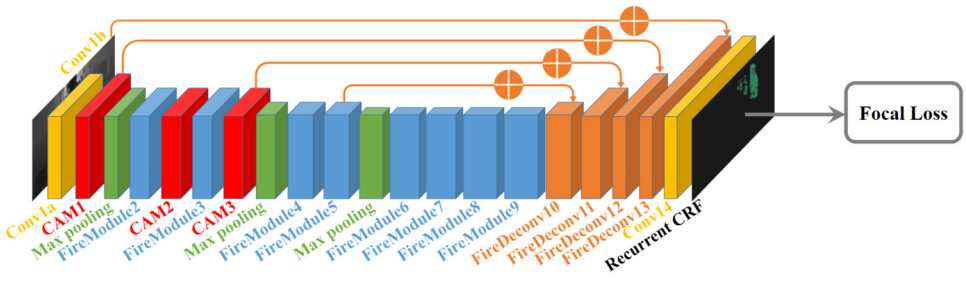


Figure 10. The SqueezeSegV2 architecture [86].

Milioto et al. (2019) [87] stated that 3DSS LiDAR-only SoTA methods were time-consuming, while their models did not have enough representational capacity, i.e., the number of neurons and learnable parameters was low. In addition to time-consuming SoTA approaches, the authors stated that there were not enough publicly available datasets for 3DSS and that SoTA methods, e.g., [68,69,80,88] cannot be applied in real-time scenarios using large-scale point clouds. Thus, the authors proposed the RangeNet++ architecture, which utilizes spherical projection to facilitate fast scene assessment and decision-making of autonomous machines. In fact, the authors claimed that due to autonomous vehicle movement, the produced point clouds were affected by skewing, i.e., the same as the rolling shutter effect in images. Thus, they de-skewed the LiDAR point cloud before the projection. The spherical projection output was an enriched range image representation, e.g., tensor, which contained the x, y, z, range, and remission information. Afterward, the authors proposed an hourglass 2DSS architecture, i.e., an encoder–decoder to perform 2DSS using the created representation. To be more specific, the encoder was a modification of the Darknet [89] backbone, e.g., to accept five channel images instead of three. In general, 2DSS experiences difficulties, e.g., blurry areas, especially in objects' borders. Additionally, the projection of the 3D point cloud onto a 2D plane resulted in a loss of information. Other methods, e.g., SqueezeSeg, used CRF to improve the 2DSS output and hence to improve 3DSS, as described earlier. However, the authors of RangeNet++ stated that the improvement in 2DSS using a CRF did not automatically result in better 3DSS, especially due to the method applied to recover the 3D labels using the 2D ones, formulated as the label re-projection problem. To be more specific, each pixel of the 2D representation had a corresponding label; however, multiple 3D points were assigned to each pixel of the image, resulting in assigning the same label, which was not accurate. Additionally, the label re-projection problem effect increased when using images with smaller resolution, which were crucial for real-time applications. To recover all the information using the created 2D representation, the authors indexed them with the corresponding image coordinates to every point of the 3D point cloud, resulting in a loss-less recovery of the 3D labels. Furthermore, the authors proposed a GPU-based kNN algorithm to cope with the label re-projection error. More concretely, for each point used to create the 2D spherical image, a window representing its neighborhood was empirically defined. Afterward, the indices stored during the creation of the 2D representation were exploited to extend the neighborhood to contain the entire set of the range neighborhoods. Then, the range readings of the central line were replaced from the unwrapped to the real ones. This representation was the key of RangeNet++ to achieve real-time performance. An analogous matrix representation was constructed for the labels. Afterward, they created a matrix that represented the range difference between the query point and its neighborhood points. Finally, the k nearest points that voted for the label of the query point were collected by using the inverse Gaussian kernel, argmin operation, and cut-off thresholding. The aforementioned post-processing kNN-based approach was adopted from many upcoming

architectures. To conclude, the authors evaluated the RangeNet++ algorithm using metrics such as border-IoU, as well as in-detail ablation studies and figures.

Xu et al. (2020) [90] described the issue of the spatially varying feature distribution of LiDAR images in detail. More concretely, the authors stated that the feature distribution along LiDAR images was different from the feature distribution of RGB images, especially for the SemanticKITTI [67] dataset due to the spherical projection applied on the data, resulting in the poor performance of the convolution operator. To be more specific, the feature distribution of LiDAR images was not identical across the images, while some features may exist only in local image regions. Additionally, the authors visualized the mean activation value of different layers of the RangeNet++ architecture, depicting the sparse filter activation across the image. To this end, Xu et al. (2020) presented the SqueezeSegV3 architecture, which utilized the Spatially Adaptive Convolution (SAC) operator. On the one hand, the traditional convolution operator did not change the kernel weights across the image. On the other hand, SAC behaved differently on each part of the image, as the filters follow the feature variations. The authors proposed different variations of SAC. The SqueezeSegV3 architecture was based on the RangeNet++ [87] architecture, using a multi-layer cross-entropy loss, aiming to use features with semantic meaning. To conclude, using different variations of SAC, the authors achieved better performance on the 3DSS of many classes using the SemanticKITTI dataset.

Cortinhal, Tzelepis, and Aksoy (2020) [91] stated that SoTA 3DSS methods predicted 3D labels without calculating uncertainty measurements, and thus, they proposed the SalsaNext architecture (Figure 11) for real-time uncertainty-aware 3DSS of LiDAR point clouds. In line with the RangeNet++ [87] architecture, SalsaNext exploited the same spherical representation but used intensity instead of remission. Furthermore, they introduced an improved SalsaNet [92] architecture to perform 2DSS using the proposed spherical representation of the 3D point cloud. More concretely, they proposed a Context Module in the first layers of the encoder, a Residual Dilated Convolution Block instead of traditional ResNet blocks, a pixel-shuffle Layer, a Central Encoder–Decoder Dropout operation, and an Average Pooling Downsampling layer instead of strided convolution to allocate less memory than SalsaNet. The Context Module had multiple dilated convolutions, i.e., with different kernel sizes, which fused various perceptive fields on different scales, aggregating global context information. Furthermore, the pixel-shuffle layers were introduced instead of transposed convolution layers during upsampling to decrease the computation complexity of the network. Finally, they added dropout after both the encoder and decoder layers, except for the first and the last layers, resulting in improved network performance. Furthermore, the authors provided an in-depth analysis of uncertainties, divided them into aleatoric, which refers to the data uncertainty, and epistemic, which refers to the model uncertainty. Moreover, aleatoric uncertainty can be divided into homoscedastic, which refers to the aleatoric uncertainty that was independent of the different types of input data, and heteroscedastic, which refers to the uncertainty that depends on them. If the sensor noise characteristics were known, a modified NN using assumed density filtering can be used to provide predictions along with their aleatoric heteroscedastic uncertainties. Finally, the authors estimated the epistemic uncertainty using Monte Carlo sampling during inference. In fact, the main drawback of dimensionality reduction methods was the information loss due to projection from 3D to a lower space. The authors used the kNN-based post-processing technique proposed in RangeNet++, applied during inference, to cope with the aforementioned drawback. To conclude, the authors provided informative visualizations of the epistemic and aleatoric uncertainties in the images of the SemanticKITTI dataset in addition to detailed ablation studies and evaluation of the SalsaNext architecture.

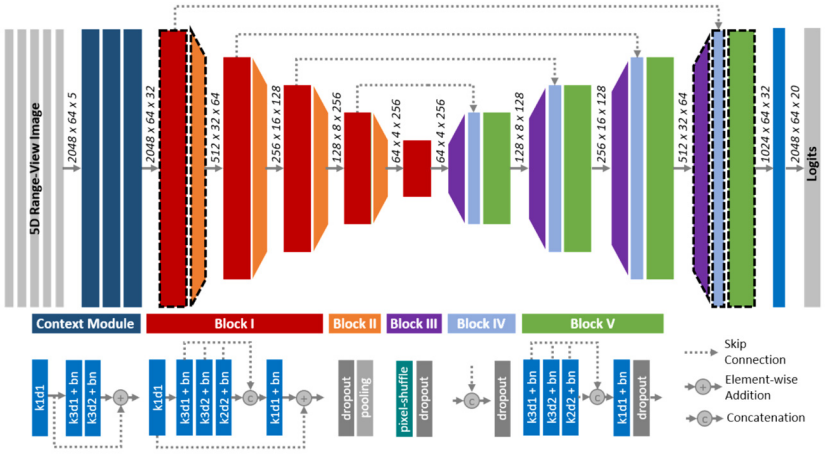


Figure 11. The SalsaNext architecture [91].

Xiao et al. (2021) [93] observed that dimensionality reduction SoTA methods usually created multi-channel images by stacking position channels, depth, intensity, and remission, commonly creating five-channel images, and processed them simultaneously without considering the distinct characteristics of each modality. Thus, the authors proposed the FPS-Net, aiming to handle the aforementioned modality gap problem by decomposing five-channel images into three modalities, thus improving the performance of 3DSS applications. Moreover, they mentioned that the naïve stacking of pixel values along channels with different distribution may result in the models being trained on modality-agnostic features. To this end, they proposed a Modality-Fused Convolution Network, which firstly learned modality-specific features and finally fused them using a high-dimensional feature space representation in an encoder–decoder fashion. More specifically, modality-specific features were encoded by using a multiple receptive field residual dense block (MRF-B) and decoded by using a recurrent convolution block (RCB). In detail, the MRF-B was composed of multidimensional convolutions, e.g., 1×1 and 3×3, along with concatenation, batch normalization, and ReLU, while the RCB was a recurrent neural network with 3×3 convolution, addition operation, batch normalization, and ReLU. The modal-specific high-dimensional features produced by MRF-B were concatenated and fed into a 1×1 convolution operation. After several MRF-B and downsampling layers, the features were passed into multiple upsampling and RCB layers. Moreover, the classifier output the predictions. Finally, a post-processing approach similar to RangeNet++ [87] was adopted to provide the final 3D labels. To conclude, the authors provided an evaluation of the proposed algorithms along with ablation studies and experiments on well-known benchmark datasets.

Li et al. (2021) [94] mentioned that SoTA methods were characterized by very expensive operations, especially for applications requiring embedded platforms. They had low accuracy, or they included millions of parameters; hence, they proposed the Multi-scale Interaction Network (MINet) (Figure 12) to handle them. They introduced a lightweight network that included multiple scale paths, each of which extracted features of different levels, e.g., low or high, regarding the scale of the input image. Additionally, the top scale paths were densely connected with all the lower scale paths. However, the authors proposed a computation strategy to avoid redundant computations. In line with the RangeNet++ [87] architecture, the authors created 5D spherical projection images, i.e., x, y, z, depth, and remission, as the input to the MINet architecture. Furthermore, the MINet architecture was composed of three modules: the Mini Fusion Module (MFM), the Multi-scale Interaction Module (MIM), and the Up-Fusion Module (UFM). Apart from the three modules, the MINet architecture included two different blocks: the MobileBlock and the BasicBlock. The former had fewer parameters, as it utilized depth-wise convolutions, while the latter was

more expensive. However, the MobileBlock was usually exploited using high-resolution images, aiming to extract detailed features, while the BasicBlock exploited using lower-resolution images, aiming to extract more abstracted features, to equalize the computation complexity. Firstly, the input 5D image was imported into the MFM. However, due to the different feature distributions of each modality, including x, y, and z coordinates as separate modalities, they were mapped into a different feature space using the convolution operation. Then, the features created from each modality were concatenated and fused using many MobileBlocks. After MFM, MIM was applied. Specifically, MIM included three paths, the top, middle, and bottom, with different scales. In each path, the scale was decreased, applying the average pooling operation, while the receptive field was increased, to gather more abstracted features. The authors showed that the previously described strategy, i.e., the decreasing resolution along with the increasing receptive field among the scale paths, resulted in efficient operations. Furthermore, the extracted feature maps on each scale were first resized using average pooling and then were passed to all the lower paths, allowing subsequent scale paths to focus on features that had not been extracted yet. Lastly, UFM fused the features extracted from the first layer of MFM to gather low-level spatial information and each path of MIM to gather multi-scale information. Then, the fused features were upsampled to the original resolution, further processed, and added. Finally, the 2D predictions were re-projected back into 3D space. To conclude, the authors included several experiments and comparisons among SoTA methods and their own methods by retraining each SoTA method from scratch and by using different metrics.

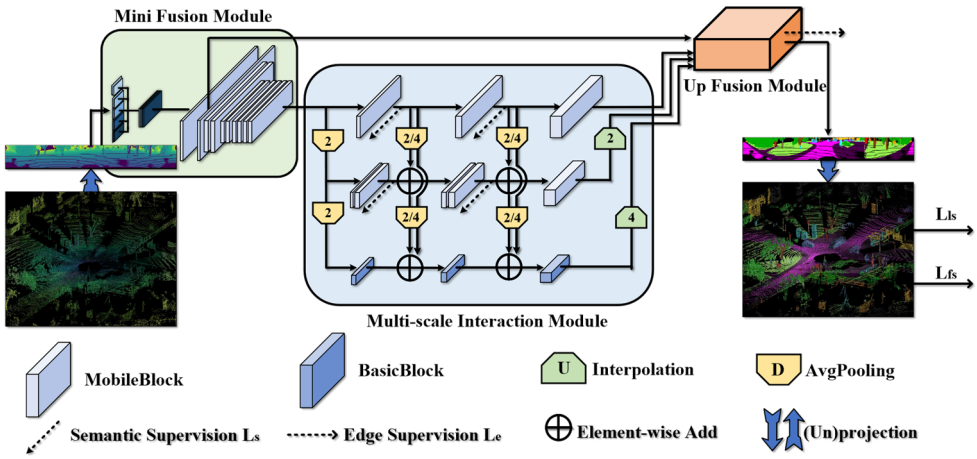


Figure 12. The MINet architecture [94].

5.2.3. Bird's-Eye View

Zou and Li (2021) [95] observed that recent works introduced urban-level datasets acquired using UAVs and included both images and 3D point clouds. In general, the created photogrammetric 3D point clouds present differences to those acquired using LiDAR sensors; for example, the existence of RGB values along with 3D geometric information. As such, they compared the category overlap of the points along the Z-axis in the photogrammetric and LiDAR point clouds and concluded that most overlapping points had the same class as the top one. Hence, they proposed the bird's-eye view (BEV) projection as the more suitable one for the 3DSS of the photogrammetric point clouds created using UAVs. To be more specific, the creation of BEV images was conducted by using a sliding window process over the 3D points. Firstly, the scale, size, and moving step hyperparameters of the sliding window were empirically defined after experiments. Then, the points in each sliding window were sorted according to their x and y coordinates to find the minimum and maximum values. Finally, each sliding window created a BEV image that included

the RGB and altitude information. Afterward, a sparse BEV image completion process was applied using the 2D max pooling operation to improve the sparse information of the projected points onto the XY-plane, resulting in the final BEV images. Furthermore, Zou and Li (2021) proposed a multimodal 2D segmentation UNet, which exploits both the RGB and altitude modalities. Additionally, the authors stated that the RGB values played a significant role in the segmentation performance. To conclude, the authors evaluated their approach over different SoTA methods (test set) using the validation set of the Sensat-Urban [96] benchmark.

5.2.4. Multiple Projections

Alnaggar et al. (2020) [97] stated that both BEV and spherical projection images provide useful features for 3DSS; hence, they can complement each other to decrease the loss of information due to the projection of 3D point clouds onto 2D space. The spherical projection image representation is similar to the RangeNet++ [87]. The BEV representation was a projection onto the XY-plane and discretization based on a 2D grid with specific dimensions, while the channels were similar to the spherical image. Moreover, they proposed a two-branch network, one for each projection, providing two different predictions that were finally fused by adding them. To be more specific, the spherical branch was an encoder–decoder architecture using the MobileNetV2 [98] backbone, while the BEV branch was based on a UNet [99] architecture. Before fusion, the segmented images were inserted into a post-processing step. Specifically, the 3D points were projected onto each segmented image. Then, the neighborhood of each point was defined using a 2D square window around the projected point. Afterward, a score vector for each point was calculated using a weighted sum of the SoftMax probabilities of all the pixels in the neighborhood. The weights were calculated based on the distance between the 3D point under investigation and those represented by the pixels in the neighborhood. More concretely, the authors paid attention to the nearest points rather than the distant ones by defining different weight values based on their distance. The final score vectors, one for each projection, were defined in the fusion step. To conclude, the authors provided an evaluation of the proposed approach in addition to an experimentation using data augmentation techniques.

Qiu, Yu, and Tao (2022) [2] observed that there were two widely used, complementary projections by the dimensionality reduction methods for 3DSS: spherical range images (RV) and top-down images, i.e., BEV. Additionally, they mentioned that the methods that exploited both representations generally used late fusion of the predicted labels, ignoring the complementary geometric information between the different views. Hence, the authors introduced the Geometric Flow Network (GFNet), aiming to exploit the geometric correspondences of each view, using the original point cloud as a bridge, in an align-before-fuse fashion. The proposed real-time network processed each view separately by a two-branch ResNet-based [75] encoder–decoder network, which adopted an ASPP [100] module at the bottleneck. To be more specific, the range view images were created by using an improved spherical projection approach proposed by Triess et al. (2020) [101]. Each spherical image had five channels, similar to Milioto et al. (2019) [87]. Furthermore, the BEV images were created using the top-down orthogonal projection, replacing the Cartesian coordinates with relative polar coordinates. Each BEV image had nine channels, i.e., three cylindrical distances (x, y, z) to the center of the BEV grid, three cylindrical coordinates (x, y, z), two Cartesian coordinates (x, y), and remission. Moreover, the authors proposed the Geometric Flow Module (GFM), which was divided into Geometric Alignment (GA) and Attention Fusion (AF). In fact, GA included the geometric transformation from the RV to BEV, and vice versa, for feature fusion. To calculate the transformation matrices, the authors used the original point cloud (PC) as a bridge, i.e., RV to PC to BEV. Furthermore,

the AF module concatenated the single-view features, e.g., from the RV image, with the transformed features, e.g., BEV to RV, then applied self-attention on the concatenated features to obtain attention scores, and, finally, combined them with the single-view features by using a residual connection, resulting in the final features. In general, several GFMs were located between the two ResNet decoders, obtaining a feature map for each view. Finally, the 2D predictions of each branch were utilized along with grid sampling and KPConv [73] to find the per-point predictions. In fact, the KPConv layer was exploited instead of the kNN post-processing approach presented by Milioto et al. (2019) to create an end-to-end learnable approach. Finally, the performance of the presented dimensionality reduction-based methods in different benchmark datasets is presented in Table 4.

Table 4. Mean Intersection over Union (mIoU) and Overall Accuracy (OA) of different benchmark datasets for dimensionality reduction-based methods based on the papers examined in Section 5.2.

Algorithm	Year	Semantic3D		ScanNet		SemanticKITTI		S3DIS		Sensat Urban	
		mIoU	OA	mIoU	OA	mIoU	OA	mIoU	OA	mIoU	OA
TangentConv	2018	66.4	89.3	43.8	55.1	40.9	-	52.8	82.5	33.30	76.97
SqueezeSeg	2017	-	-	-	-	30.8	-	-	-	-	-
SqueezeSegV2	2018	-	-	-	-	39.7	-	-	-	-	-
RangeNet++	2019	-	-	-	-	52.2	89.0	-	-	-	-
SqueezeSegV3	2020	-	-	-	-	55.9	89.5	-	-	-	-
SalsaNext	2020	-	-	-	-	59.5	90.0	-	-	-	-
FPS-Net	2021	-	-	-	-	57.1	-	-	-	-	-
MINet	2021	-	-	-	-	55.2	-	-	-	-	-
Efficient BEV	2021	-	-	-	-	-	-	-	-	-	91.37
MPF	2020	-	-	-	-	55.5	-	-	-	-	-
GFNet	2022	-	-	-	-	65.4	92.4	-	-	-	-

-: No Data.

5.3. Discretization-Based Methods

Riegler, Ulusoy, and Geiger (2017) [102] observed that SoTA discretization-based methods required exhaustive dense convolution operations, i.e., they took into account the 3D empty space, ensuing slow computations on downstream applications like 3DSS. In this regard, Riegler, Ulusoy, and Geiger (2017) proposed the OctNet, a 3D convolution-based network that avoided the empty space, proposing a new intermediate sparse representation of the 3D data. To be more specific, the new representation of the 3D data was created by subdividing the high-resolution 3D data hierarchically into octrees, taking into account the density of the 3D points and stopping when achieving the predefined resolution. Moreover, the authors proposed the Hybrid Grid–Octree Data Structure (HGODS) by stacking several shallow octrees, similar to Miller, Jain, and Mundy's (2011) [103] representation. The scope of the HGODS was the creation of a sparse representation that permitted rapid data access using bit strings because several downstream applications like 3DSS commonly require the definition of points' neighborhoods, e.g., for convolution or pooling. To this end, the authors stated that the discretization of the data followed their density, avoided the empty space, concentrated the computations only on the non-empty regions, and improved the computational and memory requirements. Furthermore, the authors presented the basic OctNet network operations, i.e., convolution, pooling, and unpooling, along with different applications like 3D classification and 3DSS. The 3DSS OctNet was an encoder–decoder UNet-shaped network. To conclude, the OctNet network was trained using different voxel sizes and features (RGB, normal vector, binary voxel occupancy, height above ground) in combination with data augmentation.

Su et al. (2018) [88] proposed the SPLATNet architecture (Figure 13), which exploited high-dimensional lattice space to perform the convolution operation on a set of features.

To be more specific, the authors stated that SoTA 3DSS methods usually transformed the raw 3D point clouds into 2D–3D grid representations to use the convolution operation on them. However, those transformations resulted in a loss of information, and thus, the authors proposed the high-dimensional lattice space for the convolution operation, aiming to reduce the loss of information of the SoTA methods. Hence, they proposed the SPLATNet architecture, stepping on the Bilateral Convolution Layers (BCLs) introduced by Miller, Jain, and Mundy (2011). In fact, BCLs can easily be operated in high-dimensional lattice spaces, e.g., the six-dimensional filtering space XYZRGB, which was a strong property in order to be chosen as the main operation of the SPLATNet architecture. More concretely, a BCL was incorporated into the Splat, Convolve, and Slice steps, each of which was written as matrix multiplication. Firstly, the Splat operation projected the input features into the permutohedral lattice space defined by the lattice features, using barycentric interpolation, along with a scale factor. Secondly, the Convolve operation was applied on the lattice space, defining N-dimensional filter weights similar to the conventional convolution operation. Finally, the Slice operation was the opposite of the Splat operation, but with the opportunity to choose if the resulting point cloud would be the same or different from the input one. The authors included an in-depth analysis of BCL advantageous properties regarding the operation using 3D point clouds. Furthermore, they performed 3DSS using two variations of SPLATNet: one using only 3D point clouds and one including images in addition to the 3D point cloud data. A unique characteristic of the lattice space was the scale, which was strongly associated with the receptive field of the convolution operation. The authors included a detailed analysis regarding the lattice scale factor. To conclude, the SPLATNet architecture was evaluated on a series of downstream applications like 3DSS and 3D part segmentation.

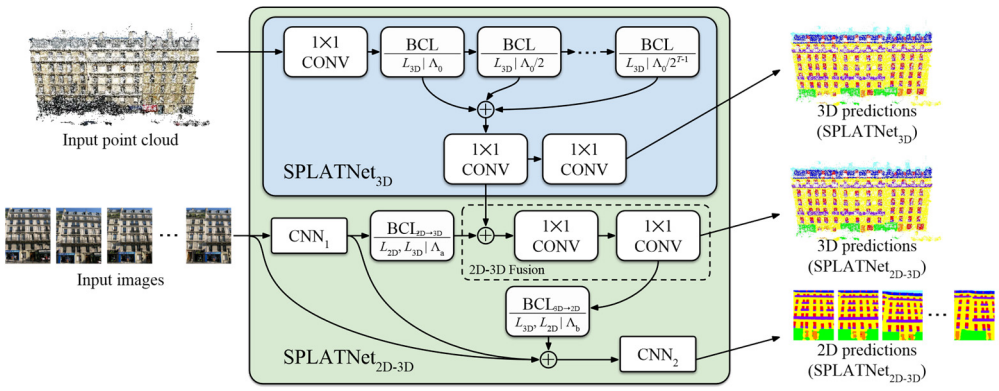


Figure 13. The SPLATNet architecture [88].

Choy, Gwak, and Savarese (2019) [104] proposed a 4D spatio-temporal convolutional neural network to interpret 3D video scan data, i.e., 3D scenes spanning over different timestamps. In general, 3D data like point clouds or voxels included empty areas that should be avoided during the calculations because they did not contribute to the performance of the networks. The authors constructed two N-dimensional sparse tensors, instead of the 3D data representation, to avoid the empty areas of the data. To be more specific, the sparse tensors were a more useful and homogeneous representation of the data, especially for high-dimensional spaces. Additionally, the authors included a detailed explanation of the generalized sparse convolution, which had multiple advantages like it was efficient, it could be applied to high-dimensional spaces, and it could be used to reproduce the milestone 2D techniques in high-dimensional space. The aforementioned units, e.g., sparse tensors and the generalized sparse convolution, were included in the open-sourced Minkowski Engine. More concretely, the Minkowski Engine included Sparse Tensor Quantization, Generalized

Sparse Convolution, max pooling, global/average pooling, Sum Pooling, and non-spatial function algorithms, which were presented in detail, including a pseudo-code explanation for each of them. Using the components of the Minkowski Engine, the authors proposed the Minkowski convolutional neural network, using ResNet [75] or UNet [99] as the backbone architecture. In general, spatio-temporal convolutions had two problems. The first one was that the computational cost was exponentially increased along with the dimensions. The second problem was that the predictions were not consistent between the different timestamps. Hence, the authors investigated the exploitation of non-conventional kernel shapes to overcome the computational cost problem and the use of trilateral stationary conditional random fields (CRFs) for the second problem. To be more specific, the authors investigated the use of different kernels between the spatial and temporal dimensions, resulting in a hybrid-shaped kernel that outperformed its counterparts, i.e., tesseract kernels. Furthermore, the proposed CRF contained a stationary 7D kernel (3D space, 1D time, and 3D color space), and thus, it was called Trilateral Stationary CRF. To conclude, the Minkowski Network was evaluated in many downstream tasks, including 3DSS, using different metrics and benchmark datasets. Finally, the performance of the presented discretization-based methods in different benchmark datasets is presented in Table 5.

Table 5. Mean Intersection over Union (mIoU) and Overall Accuracy (OA) of different benchmark datasets for discretization-based methods based on the papers examined in Section 5.3.

Algorithm	Year	Semantic3D		ScanNet		SemanticKITTI		S3DIS	
		mIoU	OA	mIoU	OA	mIoU	OA	mIoU	OA
OctNet	2017	50.7	80.7	18.1	76.6	-	-	26.3	68.9
SPLATNet	2018	-	-	39.3	-	22.8	-	-	-
MinkowskiNet	2019	-	-	73.6	-	54.3	-	65.4	-

-: No Data.

5.4. Graph-Based Methods

Wang et al. (2019) [105] stated that convolutional neural networks (CNNs) achieved high-end results in many 2D downstream tasks. However, using the convolution operation in 3D space was not a straightforward process. Hence, they proposed the EdgeConv, an operation similar to the traditional convolution, for 3D downstream applications like classification and semantic segmentation. Additionally, they proposed the DGCNN architecture (Figure 14), which was based on the PointNet architecture but without feature transformations and included the EdgeConv operation. Specifically, the proposed architecture was based on the Point cloud Transform Block and the EdgeConv Block. The former defined a 3×3 point cloud transformation by concatenating their global and local coordinates to align it in a canonical space. The local coordinates were defined by subtracting the kNN neighboring point coordinates from the center point coordinates. The latter, i.e., the EdgeConv operation, applied a convolution-based operation on graph edges, similar to the graph neural networks. To be more specific, the authors built a local neighborhood graph for each point, defining their neighborhood, i.e., connecting the neighboring points with edges. More concretely, the EdgeConv operation was defined using (i) a nonlinear and (ii) an aggregation function. Firstly, the local structure of each point in the given point cloud was defined by computing a directed graph, e.g., a kNN graph. In fact, the computed graph had 3D points as nodes connected to their neighboring nodes with edges. Secondly, the calculated edges were enriched with features using the predefined nonlinear function, i.e., edge features. Finally, the output of each EdgeConv layer was calculated by applying the aggregation function to the previously calculated features. In general, the output point cloud had the same number of 3D points but with more edge features than the input point cloud. Furthermore, the authors described in detail the choice of different nonlinear and

aggregation functions and how some SoTA methods, like PointNet, can be considered as a subset of the EdgeConv operation, i.e., by defining the appropriate set of the nonlinear and the aggregation functions. Finally, they introduced the asymmetric edge function, which was selected for the DGCNN architecture because they stated that it was thoroughly combined the global and local point cloud features and also could be defined as a shared MLP. Moreover, the authors found that finding the graph kNN in a feature space other than the Euclidean space was beneficial for their network. Thus, they proposed updating the graph kNN dynamically into each layer. To conclude, the authors presented an evaluation of the DGCNN architecture in a series of high-level tasks like 3D classification, 3D part segmentation, and indoor 3DSS, using several SoTA methods as benchmark.

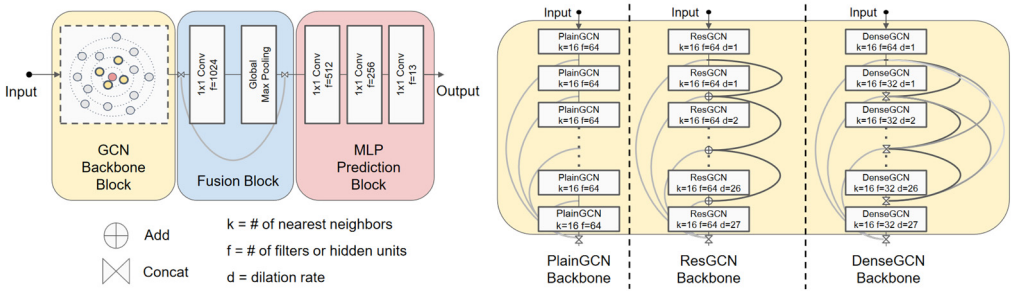


Figure 14. The GCN architecture (**Left**) and different backbone networks created to serve as the GCN backbone block (**Right**) [105].

G. Li et al. (2021) [106] observed that SoTA GCN models were shallow, basically due to the vanishing gradients problem and the high complexity in the computation of backpropagation. Hence, they investigated the transfer of several techniques like residual/dense connections and dilated convolutions from CNNs to GCNs to create deep GCNs for different high-level tasks like 3DSS. First and foremost, the authors defined the meaning of graphs. Furthermore, they defined the general idea of GCNs, i.e., the definition and application of the update function (MLPs, Gated Networks, etc.) or the nonlinear function [105] and the aggregate (mean, max, attention, etc.) function. In their framework, the authors chose the max aggregation function and an MLP with batch normalization and ReLU as the update function. Moreover, they described the differences between the fixed graph representation, commonly used in GCNs, with the Dynamic Graph representation, which was presented by Wang et al. (2019), concluding that the later representation was more beneficial for the networks. Hence, they included a Dilated kNN function to dynamically change the points neighbors in each layer to increase the GCNs receptive field. Afterward, the authors introduced in detail the *Residual Connections for GCNs* and the *Dilated Aggregation for GCNs* operations. Then, they analyzed different *Deep GCN variants*, e.g., they included the introduced operations into the EdgeConv one, to train deeper GCNs, etc. Most importantly, the authors included a detailed experimental process using a plethora of Graph Learning techniques. Also, they proposed three variations of the GCNs backbones: the *PlainGCN*, the *ResGCN*, and the *DenseGCN*. The selected backbone was the only different part among the proposed architectures during the experiments, i.e., the under-investigation part of the evaluation process was the backbone architecture. More concretely, the *PlainGCN* architecture was similar to DGCNN. In addition to the PlainGCN architecture, the dynamic dilated kNN graph and the residual graph connections were added to form the *ResGCN* architecture. Finally, the DenseGCN was defined by changing the residual with dense graph connections. To conclude, the authors presented an in-depth comparison of the different architectures and modules using many ablation studies. Finally, the performance of the presented graph-based methods in different benchmark datasets is presented in Table 6.

Table 6. Mean Intersection over Union (mIoU) and Overall Accuracy (OA) of the S3DIS benchmark datasets for graph-based methods based on the papers examined in Section 5.4.

Algorithm	Year	S3DIS	
		mIoU	OA
DGCNN	2019	58.2	84.1
Deep GCN	2021	60.0	85.9

5.5. Hybrid Methods

5.5.1. Discretization-, Point-, and Dimensionality Reduction-Based Methods

Zhang et al. (2020) [107] introduced the PolarNet architecture, aiming to cope with the LiDAR point cloud irregularity property and the use of many detailed semantic classes while retaining real-time perception performance. The authors described the very important role played by the size of the receptive field in 2DSS performance. However, they concluded that in 3DSS, the shape of the receptive field was also important in addition to the size. Firstly, they proposed a bird's-eye view (BEV) representation, observing that this view of LiDAR scans organizes the points in rings with different radii. Moreover, they described that a partition of the BEV representation using a Cartesian grid would result in an uneven distribution of points into each grid cell. To be more specific, the grid cells that were closer to a LiDAR sensor would contain more points, while those that were further away would contain fewer points. Additionally, there were many empty Cartesian grid cells. Hence, they proposed a polar grid feature learning approach, instead of the Cartesian grid, to exploit the rings with different radii described earlier. More concretely, the authors first calculated the azimuth and radius of each point onto XY-plane using the LiDAR sensor as the origin. Then, the points were assigned to a grid cell using the calculated azimuth and radius, values resulting in the BEV grid cell representation of the input 3D point cloud. In fact, each grid cell was similar to the point pillar representation [108], i.e., XY-plane coordinates and unlimited spatial extent of the Z-direction. Afterward, the points inside each cell were passed into a kNN-free PointNet followed by a max pooling operation, resulting in a set of fixed-length features (1×512). The output features were assigned to a ring matrix taking into account the spatial location of their corresponding grid cell. Moreover, the authors proposed ring convolution, aiming to predict the 3D point labels using the ring matrix as an input. Specifically, ring convolution was operated on the ring matrix along the radius axis. The authors mentioned that any 2DCNN network can process the created representation by replacing the traditional CNN operation with discrete ring convolution. Finally, they reshaped the ring predictions into a 4D matrix to exploit voxel-based segmentation loss. To conclude, PolarNet used various techniques inspired by point-, dimensionality reduction-, and discretization-based methods. Also, the authors investigated in-depth the performance of PolarNet, including several experiments with well-known benchmark datasets and comparisons with many baseline networks.

Gerdzhev et al. (2020) [109] proposed the TORNADO-Net architecture, which included techniques proposed into the PolarNet [107] and SalsaNext [91] architectures. To be more specific, they used the general idea of PolarNet, but they combined features from both BEV and range images, under the pillar projection learning scheme. Furthermore, similar to SalsaNext, they used the same range image creation process as well as a similar method to the *Context Module*, called the *Diamond Contextual Block*, which included different 2D techniques to improve the SalsaNet architecture. However, the main contribution of this architecture was the implementation of a total loss function, exploiting the combination of weighted cross-entropy loss and Lovasz–SoftMax loss, as proposed in SalsaNext but with the addition of the total variation loss along with different weights for each part. Finally, a cut-off thresholding [87] and a post-processing approach using kNN [87] were exploited

to improve the network's performance. To conclude, the authors presented several quantitative and qualitative analyses, achieving high-end results, evaluated measuring the mIoU on the SemanticKITTI benchmark.

Liu et al. (2023) [110] stated that the cross-modal and cross-view fusion approaches had not been thoroughly investigated yet, while the information gathered using multimodal data and different views of point clouds were complementary to each other. Furthermore, the authors observed that 3D point clouds carried detailed geometric information, while images carried detailed semantic information. Hence, they proposed the UniSeg architecture (Figure 15), which fused the features gathered from the point, range, and voxel representations of a 3D point cloud, in addition to those gathered from 2D images, to improve the performance of both 3DSS and 3D panoptic segmentation. Firstly, the authors extracted the point, range, voxel, and image features using conventional methods, i.e., MLPs, spherical projected images, max pooling on the voxel representation, and the ResNet architecture, respectively. Secondly, they proposed the *Learnable cross-Modal Association* (*LMA*) *module* and the *Learnable cross-View Association* (*LVA*) *module* to fuse the voxel and range features with the image features and to transform the fused features into the point space and then combine them using different views of the point cloud, respectively. In more detail, the LMA module fused the features gathered from the voxel and range representations with the image features, resulting in the set of the image-enhanced voxel and range view features. To be more specific, for each voxel, the voxel features and their corresponding image features were passed into a multi-head cross attention module, resulting in enhanced voxel features. The range-view features were processed similarly to the voxel features, resulting in enhanced range features. Before the enhanced features were fed into the LVA module, they were first transformed into the point space using trilinear and bilinear interpolation in order to alleviate the quantity mismatch problem. Afterward, the enhanced features were fed into the LVA module. Moreover, all the features, i.e., the transformed range and voxel features and the point features, were concatenated, resulting in the multi-view features. Then, the multi-view features were further processed, producing view-wised adapted features, using the original point space features. and they were projected back to the voxel and range representation, forming the final set of features for 3DSS. To conclude, the UniSeg architecture exploited the created features using different heads depending on the application, i.e., 3DSS or 3D panoptic segmentation, while it was evaluated using different benchmark datasets like SemanticKITTI, nuScenes, and Waymo Open.

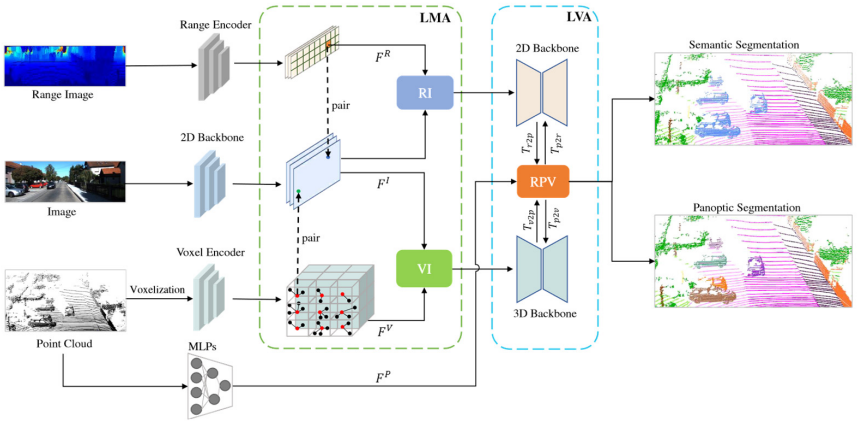


Figure 15. The UniSeg framework [110].

5.5.2. Graph- and Discretization-Based Methods

Yan et al. (2020) [111] observed that LiDAR 3D point clouds were characterized by sparsity, resulting in underperformance on the 3DSS task. Additionally, they stated

that combining multi-temporal scans could be used to create a dense representation of a scene and thus to improve the performance of 3DSS. However, SoTA multi-temporal methods were commonly used only previously to the current scans, and thus, they cannot exploit the upcoming frames. Also, they introduced time-consuming feature aggregation techniques like kNN, which harden their application on the self-driving task. Hence, Yan et al. (2020) [111] proposed the JS3C-Net architecture, which exploited both 3DSS and 3D Semantic Scene Completion (3DSSC) modules to equally improve them along the execution. To be more specific, the 3DSS module used a U-Net architecture implemented based on the SparseConv operation introduced by Graham, Engelcke. and Maaten (2018) [112] to create the voxel-based 3DSS output. Afterward, the voxel-based predictions were transferred to the original 3D point cloud by nearest neighbor interpolation. Then, the transformed features were fed into three MLPs. The first MLP transformed the point-wise features into a shape embedding (SE) that was then fed into the *Point–Voxel Interaction*, which was a submodule of the 3DSSC module. The created SE was then used diversely. Firstly, the point-wise features were fed into the second MLP and then were fused with the SE using an element-wise summing operation. The fused features were fed into the third MLP, resulting in the 3D point cloud semantic segmentation prediction of the 3DSS module, which was the input for the 3DSSC module. Moreover, the proposed 3DSSC module first densely voxelized the produced 3DSS map, creating a 3D high-resolution volume, which was further processed using convolution, pooling, concatenation, and upsampling layers along with skip connections to use multi-scale features. Finally, the 3DSSC module output was a set of voxelized coarse completion features, which was also fed into the *Point–Voxel Interaction* submodule. Furthermore, the *Point–Voxel Interaction* included the SE from the 3DSS module and the coarse-voxel-based completion from the 3DSSC module and aimed to exploit them to transform the coarse completion into a fine one. To achieve that, the coarse completion non-empty voxel centers were first collected to create a new point cloud. Then, the k-nearest neighbors between the SE and the new point cloud were gathered. Finally, several GCN layers, inspired by Wang et al. (2019), were stacked together to obtain the refined completion exploiting the kNNs. To conclude, the authors presented an evaluation of their hybrid architecture by comparing its performance with SoTA methods using data from the SemanticKITTI benchmark.

5.5.3. Point- and Discretization-Based Methods

Liu et al. (2019) [113] observed that the limitations of point-based methods and voxel-based methods were usually complementary to each other, and thus, a combination of techniques from both could result in memory-efficient 3D deep learning models. On the one hand, voxel-based methods quantized the given point cloud, resulting in the loss of information while creating a regular representation with good memory locality, i.e., mimicking the 2D grid representation. On the other hand, point-based methods had a significant latency in order to overcome irregular memory access as well as to find the relative distances among the point neighbors and the center point. Basically, the irregular property of point clouds but with the advantage of a small memory footprint. To this end, the authors proposed the PVCNN architecture as a hybrid approach, which exploited the advantages of voxel- and point-based methods using the proposed point–voxel convolution (PVConv) operation. To be more specific, the PVConv operation was decomposed into two branches, *Voxel-Based Feature Aggregation* (*VBFA*) and *Point-Based Feature Transformation* (*PBFT*), to capture coarse-grained features and fine-grained features, respectively. More concretely, VBFA first normalized the given point cloud before voxelization. To achieve that, the 3D points were expressed with respect to the gravity center of the point cloud and divided by the largest distance. Finally, the points were scaled and translated in order

to span between 0 and 1. The normalized point cloud was then fed into the voxelization step to create the 3D volumetric representation. In this regard, the authors presented an informative figure comparing the GPU memory requirements, the information loss, and the voxel resolution. Moreover, multiple 3D convolutions were applied using the 3D voxel grid in the feature aggregation step, followed by batch normalization and the nonlinear activation function. Finally, the authors exploited trilinear interpolation to map the voxel-based features to the point cloud domain. Furthermore, PBFT was applied to an MLP directly on the original point cloud, resulting in a set of point-wise features. Finally, the coarse-grained and the fine-grained features, created from the VBFA and the PBFT, respectively, were fused using the addition operation. To conclude, the authors included thorough experimentation and evaluation of the proposed architecture with respect to different SoTA methods, stating that PVCNN and PVConv were efficient and effective.

Tang et al. (2020) [114] observed that PVConv [113] and sparse convolution [104] operations struggled to capture the small instances of a 3D scene, especially using hardware with limited memory, due to point cloud coarse voxelization and aggressive downsampling, respectively. To overcome these limitations, the authors proposed the *Sparse Point–Voxel Convolution* (*SPVConv*) module along with a *3D Neural Architecture Search* (*3D-NAS*) process. Specifically, the SPVConv operation had two branches, the point-based and the voxel-based branch, which communicated through sparse voxelization and devoxelization operations. The former preserved high-resolution details, while the latter operated on a multi-receptive field manner. Firstly, the original point cloud was sparsely voxelized by exploiting a GPU-based hash table representation of the data. Furthermore, the hash table was exploited by both the devoxelization and feature aggregation procedures. More concretely, feature aggregation was implemented using sparse convolution residual layers. The information gathered through the voxel-based branch was transformed back into the 3D point cloud through the devoxelization operation. In parallel, an MLP was applied on the original point cloud, resulting in a fine detailed set of features. Both features, i.e., from the voxel-based and point-based branches, were finally fused to be used for point labeling. Moreover, the authors proposed the 3D-NAS process, in which multiple models were automatically assessed in order to automatically find the most efficient one for each application. To this end, the authors defined the searching space by incorporating the channel number and the network depth and finally using an evolutionary architecture search regarding a set of predefined hardware constraints to find the most efficient model. To conclude, the best model was evaluated on 3DSS, among other downstream tasks, using different SoTA models and the SemanticKITTI dataset.

Rosu et al. (2020) [115] presented the LatticeNet (Figure 16) a hybrid network, which exploited both the PointNet architecture and lattices for 3DSS. Firstly, the authors described in detail the notation used in the paper, the permutohedral lattice structure, and the existing lattice operations, i.e., *Splatting*, Convolving, and *Slicing*, similar to the SplatNet architecture. However, they observed that the weights used in the *Splatting* and *Slicing* operations, from SoTA methods, were defined using barycentric interpolation and stated that a better interpolation could be defined by changing the weights into a set of learnable parameters. Furthermore, they proposed four new lattice operations: *Distribute*, *Downsampling*, *Upsampling*, and *DeformSlicing*. Firstly, the *Distribute* operation aimed to enrich the lattice vertices with a list of features. To achieve that, the coordinates and the features of the contributing points were concatenated and processed using PointNet, resulting in the final lattice vertices values. The coordinates were defined with respect to the mean value of the contributing points, before the distribution operation, to include the local information of the semantic class. Additionally, the authors stated that the proposed operation aimed to avoid the naïve summation of the *Splatting* operation and thus to preserve the

information through the network. The Downsampling and Upsampling operations followed the same idea. Firstly, a coarse lattice was produced by repeatedly dividing the 3D point cloud coordinates by two. In each repetition, a coarse lattice was produced, while the previous lattice was referred to as the finer lattice. On the one hand, during the *Downsampling* operation, the coarse lattice was embedded into the corresponding finer lattice by scaling it up by two, resulting in a set of coarse vertices in the finer vertices space. Then, the convolution operation was applied on the finer lattice vertices, using a step equal to one, to obtain the coarse vertices' values, an idea similar to strided convolution. On the other hand, the *Upsampling* operation embedded the finer vertices into the coarse vertices' space by dividing them by two, and then the convolution operation was applied using a step equal to minus zero point five, similar to transposed convolution. Moreover, the *DeformSlicing* operation aimed to improve the Slicing operation by learning to shift the position of the barycentric coordinates, i.e., allowing a data-driven interpolation instead of a simple barycentric interpolation. Finally, the authors proposed a U-Net structure network, including, firstly, a *Distribute* operation then a series of ResNet blocks, *Downsampling* and *Upsampling* operations, and, finally, a *DeformSlicing* operation. To conclude, the authors presented several experiments and ablation studies to evaluate the performance of the LatticeNet architecture.

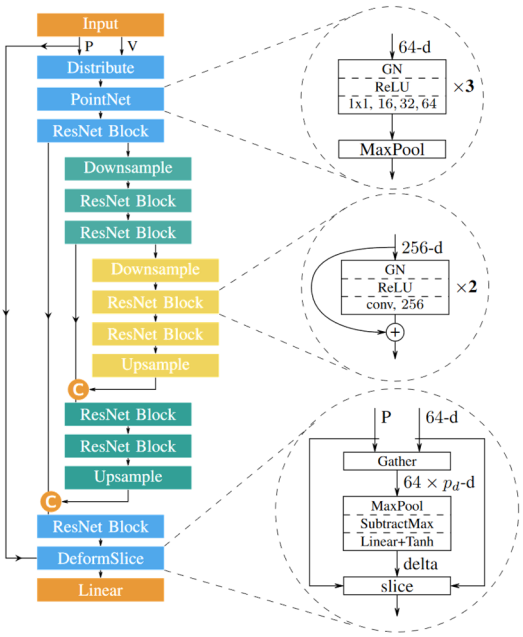


Figure 16. The LatticeNet architecture [115].

Cheng et al. (2021) [116] stated that SoTA methods for 3DSS were characterized by high computational complexity and an inability to gather the fine details of small objects. To overcome these limitations, they proposed the $(AF)^2$-S3Net architecture, exploiting techniques from both discretization- and point-based methods. To be more specific, the proposed network used the MinkNet42 [104] as backbone, enriched with the *Attentive Feature Fusion Module* (*AF2M*) and the *Adaptive Feature Selection Module* (*AFSM*). First and foremost, the input point cloud was transformed into a sparse tensor, similar to the MinkowskiNet approach, containing the 3D Cartesian coordinates along with per-point normals and intensity as features. Afterward, the sparse tensor was fed into the hybrid AF2 module, in which multi-scale point-wise and voxel-based features were extracted using different branches. In total, three branches were used: the first one was focused on the fine details of smaller objects, while the other two were focused on global features using attention maps. Finally, the multi-branch features were fused using summation. Afterward,

the fused features were further processed by convolutional layers, in an encoder–decoder fashion, while each branch features were fed independently into the AFS module. In the AFS module, each branch features were first processed using convolutional layers, resulting in a new set of features for each branch. Then, the new features were fused and fed into a shared squeeze re-weighting network [117], resulting in the output of the module, which was passed to the last transposed convolution for learning stability purposes. Finally, the predicted labels, first for the sparse tensor and then for the original point cloud, were exported from the network decoder. To conclude, the authors presented an evaluation of the proposed approach on two benchmark datasets for 3DSS and a qualitative analysis using several SoTA methods.

Zhu et al. (2021) [118] stated that dimensionality reduction-based methods for the 3DSS of outdoor scenes inevitably lose a significant amount of information due to the projection operation. Furthermore, they stated that discretization-based methods slightly improved the dimensionality reduction methods. Moreover, they observed that outdoor 3D point clouds suffer from sparsity and varying density. To overcome these limitations, the authors proposed a new framework that incorporated two components: the *Cylindrical Partition* (*CP*) and the *Asymmetrical 3D Convolution Network* (*A3DCN*). To be more specific, the uniform cube voxelization process did not take into account the varying density of LiDAR outdoor point clouds, i.e., the cell size was independent of the distance. The general idea of the authors was to use the cylindrical partition to follow the point density, i.e., the cell size to be increased following the distance. Hence, the farther cells contained more points than their counterparts on uniform cube voxelization, i.e., the cells had a more balanced point distribution. To prove that, the authors explained the pros and cons of uniform and cylindrical voxelization. More concretely, the CP component first transformed the 3D points from a Cartesian to a Cylinder coordinate system. Then, 3D partitioning was performed. Meanwhile, the original point cloud was fed into multiple MLPs, resulting in a set of features which were assigned to their corresponding cylindrical cells, gathering the cylindrical features sets. Afterward, the cylinder was unrolled, resulting in the representation that was fed into the A3DCN component. The authors observed that the objects typically occupied the crisscross area of the cylindrical partition, and hence, those cells should be strengthened compared to the others, resulting in an asymmetrical cell operation. First and foremost, the authors defined the asymmetrical upsample and downsample operations. Then, the Asymmetrical Residual Block enhanced the kernels that operated on the crisscross area and applied a series of asymmetrical downsample and upsample operations. Afterward, the *Dimension Decomposition-based Context Modeling* (*DDCM*) component was applied to gather the global context of the point clouds by stacking several low-rank features, resulting in a discretization-based 3DSS, i.e., a label for each cell, which suffered from information loss. To this end, the authors proposed the *Point-wise Refinement Module* (*PRM*) to enhance the 3DSS output with fine grained details. To be more specific, RPM first projects the 3D convolution features gathered from the aforementioned process to the point-wise features. Finally, a series of MLPSs, which communicated with the first set of MLPs, was used to fuse point-wise and voxel-wise features, resulting in the refined output. To conclude, the authors evaluated their framework on the SemanticKITTI and nuScenes datasets and among different SoTA methods.

Yan et al. (2022) [119] stated that modality-specific 3DSS suffers from the limitations inherited by each sensor capability. Additionally, they observed that LiDAR and images complement each other and thus overcome some of the limitations of each sensor; hence, fusion-base methods seem to be beneficial for 3DSS. However, different fusion techniques like point–pixel correspondence construction, along with the different Field-of-View (FOV) of the sensors and the more computational resources required, downgrade the SoTA

fusion-based methods. To this end, the authors proposed the 2D Priors-Assisted Semantic Segmentation (2DPASS) framework, aiming to overcome the aforementioned issues. To be more specific, each modality was processed using a modal-specific architecture. Concretely, the 2D encoder–decoder architecture was based on the ResNet34 architecture and the fully convolutional layer (FCN), while the 3D encoder–decoder architecture was based on sparse convolution, similar to the Tang et al. (2020) [114] implementation, resulting in a 2D and a 3D set of multi-scale features. Moreover, the authors aimed to use complementary information and thus to create point–pixel correspondences. To achieve that, the authors exploited the perspective projection along with a timestamp calibration between the two modalities. In fact, the authors used the 2D branch only during training using the projected ground truth 3D labels as 2D ground truths. Afterward, the 2D-3D features were fused using the key component called *Multi-Scale Fusion-to-Single Knowledge Distillation* (*MSFSKD*). More specifically, the 3D features were fed into an MLP, resulting in a new set of features that was further concatenated with the 2D features through another MLP gathering the fused features. However, similar to the ResNet idea, the authors enhanced the original 3D features with the new set of features, preserving the modality-specific information through the process. Finally, both the fused and the 3D features were fed into independent classifiers, gathering the semantic scores. To conclude, the authors evaluated their method on the SemanticKITTI and nuScenes benchmarks among a wide range of SoTA methods.

5.5.4. Point- and Graph-Based Methods

Landrieu and Simonovsky (2018) [120] observed that SoTA methods struggled to process large-scale point clouds, especially for 3DSS. Hence, they proposed a hybrid architecture that combined the point cloud graph representation with the PointNet architecture. Specifically, the main contribution was the SPGraph representation, which was proposed to handle large-scale point clouds. To be more specific, the proposed approach divided 3DSS into four steps: *Geometrically Homogeneous Partition* (*GHP*), *Superpoint Graph Construction* (*SGC*), *Superpoint Embedding* (*SE*), and *Contextual Segmentation* (*CS*). The first step, i.e., GHP, took as the input the entire point cloud and aimed to decomposed it into different geometrically homogeneous parts. This process was similar to simple segmentation. To achieve that, the neighborhood of each point was defined. Furthermore, several neighborhood characteristics were calculated, e.g., linearity, planarity, scattering, verticality, elevation, etc., based on the covariance matrix of the points' neighborhood, which constituted the features of the points in addition to the observations, i.e., color, intensity, etc. Finally, an approximate solution of the generalized minimal partition problem was found using an adjacency graph technique and the l_0-cut pursuit algorithm [121], resulting in several point components, each of which constituted the homogenous simple shapes called superpoints. Moreover, the SGC was aimed at creating the oriented attributed graph representation called SPG. More concretely, the SPG was a graph with nodes representing the superpoints, i.e., a set of 3D points define simple shapes, which were created in the previous step, and edges representing the adjacency between them, called super edges. The adjacent superpoints were defined using a symmetric Voronoi adjacency graph of the original 3D point cloud, while the edge features were defined by calculating the covariance matrix of each set of points and finding different features like the surface ratio, volume ratio, length ratio, etc. Thus, the SPG representation was created. Next, the SE step aimed to find a descriptor for each superpoint. To this end, the PointNet architecture was applied to each superpoint, which had a reliable number of points after a rescaling process. However, the original superpoint was included as a feature after the max pooling operation to preserve the original shape of the partition. Finally, each point in the superpoints obtained its label using a Filter Generating Network (FGN). To be more specific, the FGN was based on the ideas of

Gated Graph Neural Networks, *Edge-Conditioned Convolutions*, and *Gated Recurrent Units* (*GRUs*), resulting in the point labels. To conclude, the authors included an in-depth analysis for each terminology included in the paper along with an evaluation of the proposed algorithm in comparison to SoTA methods and using benchmark datasets.

5.5.5. Dimensionality Reduction- and Point-Based Methods

Kochanov, Nejadasl, and Booij, (2020) [122] proposed the KPRNet architecture, which exploited a dimensionality reduction- and point-based method. Firstly, a 2DSS was performed, resulting in 2D labels. Then, the 2D labels were projected back into 3D space. The authors observed that a wide variety of methods explored a post-processing step to refine the re-projected labels using either a kNN- or CRF-based approach. Instead, they proposed using a KPConv layer to predict the final 3D labels. Additionally, the authors stated that most of the dimensionality reduction methods exploited the spherical projection to create range images. However, they proposed unfolding the scans in a similar way to that which LiDAR acquired data, resulting in smoother range images than spherical-projected range images. Finally, the authors evaluated the KPRNet architecture on the SemanticKITTI dataset and compared the metrics with those of some of the SoTA methods.

Alonso et al. (2021) [123] stated that point-based methods were computationally expensive due to the 3D point neighborhood search and that 3D space operations were more complex than those in 2D space. Thus, they proposed the 3D-MiniNet architecture (Figure 17) to overcome the aforementioned limitations. Specifically, the 3D-MiniNet architecture was decomposed into three submodules: the *Fast 3D Point Neighbor Search* (*F3PNS*), *3D-MiniNet*, and *post-processing*. Firstly, the F3PNS module received the original 3D point cloud and projected it using the spherical projection. Then, the neighborhood of each point was gathered using a sliding window on the spherical image. Afterward, the features of each point in a neighborhood were augmented by computing the relative features to the mean point of them, resulting in a set of eleven features for each point. Furthermore, the 3D-MiniNet module had two submodules: the *Projection Learning Module* (*PLM*) and the *2D Segmentation Module* (*2DSM*). Firstly, the PLM extracted three types of features, i.e., local, context, ands spatial, and finally fused them. To be more specific, the input group of points, defined in the F3PNS, were fed into the *Local Feature Extractor* (*LFE*) (Figure 17 (a)), extracting PointNet like features. Meanwhile, after the second layer of the LFE, the points and their features were fed into the *Context Feature Extractor* (*CFE*) (Figure 17 (b)), in which the neighbor search implementation was applied using different window sizes, resulting in an informative set of features. Both features, i.e., from LFE and CFE, were concatenated and fed into the max pooling operation. Additionally, the input groups were also fed into the *Spatial Feature Extractor* (*SFE*) (Figure 17 (c)), in which a convolution-based implementation was applied. Then, the fused features were further concatenated with the SFE features and fed into the feature fusion (FF) (Figure 17 (d)), submodule, which fused the concatenated features using a three step processes. Firstly, the concatenated features were fed into a reshaping process. Meanwhile, the concatenated features were processed using average pooling, a 1×1 convolution, and a sigmoid function. Afterward, self-attention was applied by multiplying the reshaped features and the computed features. Finally, the features were reduced and fed into the 2DSM module. More concretely, the 2DSM contained two branches. The first branch took as the input the output of the PLM and processed it using the MiniNet architecture. The second branch took as the input the spherical image and processed it to extract high-resolution features. Finally, the 2DSS labels were extracted and re-projected back into 3D space. Afterward, a kNN-based post-processing approach similar to Milioto et al. (2019) was applied. To conclude, the authors

evaluated the 3DMiniNet architecture using the SemanticKITTI dataset and compared it with different SoTA methods.

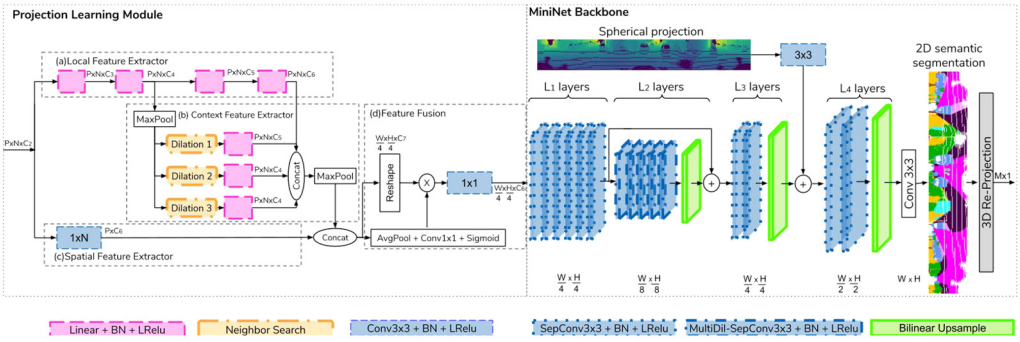


Figure 17. The 3D-MiniNet architecture [123]. The Local Feature Extractor (a), extracts PointNet like features, using the given point cloud. The Context Feature Extractor (b) exploits the points and their LFE features to extract a new informative set of features. The Spatial Feature Extractor (c) extracts convolution like features using the given point cloud. The Feature Fusion (d) module combines the calculated features.

Robert, Vallet, and Landrieu (2022) [124] proposed a hybrid 3DSS approach that exploited 3D point clouds and a set of images along with their poses. The authors stated that, in general, the images better captured the textural and contextual information than 3D point clouds. Hence, the proposed method aimed to assess the given images based on their viewing conditions and finally use them in combination with a 3D point cloud for 3DSS. First and foremost, a *Point–Pixel Mapping* (*PPM*) process was constructed to link each 3D point with a set of images in which it was visible. To achieve that, a method similar to Z-buffering was created, resulting in a set of images for each 3D point. Afterward, its image–point pair was assessed to extract the viewing conditions using a vector that included the normalized depth, some local geometric descriptors, the local density, and the viewing angle with respect to the normal vector, among other metrics. Furthermore, the authors exploited the input data along with the PPM outputs to gather diverse features for each point. The authors stated that based on the viewing conditions found earlier, each image could be used diversely, contributing to different types of information, like detailed textural information, important contextual cues, etc. To this end, the authors exploited a deep-set architecture to predict a vector that represented the quality of the images corresponding to a 3D point. Moreover, the predicted quality vector was fed into the SoftMax function to obtain the attention scores. In the case that the images' quality was poor, the authors blocked the extraction of relevant features from them and relied only on the geometric information, using a gating parameter. Finally, the features gathered for each image in the image set of a 3D point were fused and assigned to it. The aforementioned method was called the *Multi-View Feature Aggregation Method* by the authors and was exploited under a *Bimodal Point–Image Network* for 3DSS purposes. To be more specific, the proposed bimodal network was composed of a 2D FCN, a 3D encoder–decoder network, and a fusion of the 2D-3D features strategy. To conclude, the authors investigated different fusion techniques, i.e., early, intermediate, and late, and evaluated their network on three benchmarks (S3DIS, ScanNet, and KITTI-360) against several SoTA methods.

Wang, Zhu, and Zhang (2022) [125] observed that autonomous vehicles captured sequences of LiDAR data, while SoTA 3DSS methods usually processed them using a single frame at a time. Hence, they proposed a range residual spherical image representation, aiming to capture both the spatial and temporal information of LiDAR sequential data for 3DSS. More specifically, the proposed approach, called Meta-RangeSeg, was composed of three steps: Range residual image generation, feature extraction, and post-processing. The

former step created a nine-channel spherical residual image representation of the sequential 3D point clouds. To be more specific, the authors called the proposed representation residual because they included three residual channels in the created spherical images, in addition to the remission, the 3D coordinates, and an indicator m, which defined if a pixel position was a projected point or not. More concretely, the residual channel number was equal to the sequential frames that were used in addition to the current frame. The authors included three previous LiDAR scans; hence, there were three residual channels: d1, d2, and d3. Specifically, the residual channels were created by applying three steps. Firstly, the 3D point clouds of the previous frames were transformed into the coordinate system of the current frame. Then, the transformed point clouds were projected, creating range images. Finally, the residual images were created by calculating the absolute difference among the range images of the previous frames and the range image of the current frame. Moreover, the created spherical images were fed into the feature extraction step, which included the MetaKernel and U-Net blocks. The MetaKernel block included a sliding 5×5 window on the spherical image, which was used to indicate the point neighbors. Then, the relative 3D coordinates and range of the neighborhood with respect to the center point were calculated and fed into an MLP. Finally, an element-wise product between the learned weights and the MLP features was calculated, followed by concatenation and 1×1 convolution operations, resulting in a set of meta-features. Afterward, the meta-features were fed into a U-Net encoder–decoder network with four downsampling layers and upsampling layers with skip connections. Hence, two types of features were produced: the meta-features, using the MetaKernel, and the multi-scale features, using the meta-features fed into a UNet network. Then, the *post-processing* step was applied, taking as the input only the range channel, which was the most valuable one according to the authors, the multi-scale features, and the meta-features under the *Feature Aggregation Module* (*FAM*). Firstly, the range channel was fed into the *Context Module* presented to the SalsaNext [91] architecture, followed by several concatenation, convolution, batch normalization, ReLU, and element-wise product operations. The resulting features were fused with the multi-scale features, creating range-guided features, which were concatenated with the meta-features. After the concatenation, several layers with the same operations, i.e., convolution, batch normalization, etc., were applied, resulting in a set of 2D labels. Finally, the 2D labels were converted into a 3D prediction following the kNN post-processing approach presented by Milioto et al. (2019). To conclude, the authors evaluated the Meta-RangeSeg algorithm using the SemanticKITTI [67] and SemanticPOSS [126] benchmarks along with the mIoU metric.

5.5.6. Dimensionality Reduction- and Discretization-Based Methods

Wu et al. (2024) [127] observed that the advancements in 2D deep learning methods were based on scale principals, e.g., dataset size, the number of model parameters, the size of the receptive field, etc. However, the investigation of the scale principals in the 3D domain was not a straightforward process due to the limited size and diversity of the available 3D point cloud datasets. Additionally, by observing the advantages of sparse convolution [104], especially using large 3D point clouds, they hypothesized that the 3D deep learning algorithm's performance was linked to the scale principals more than the complex architecture design's. Hence, Wu et al. (2024) [127], motivated by the introduction of scale principals into transformers, proposed the PTv3 architecture. In fact, SoTA methods commonly investigate a tradeoff between efficiency and accuracy. To be more specific, the authors emphasized simplicity and efficiency over accuracy in order to leverage the scale principals. Firstly, they propose a method different to the kNN neighborhood extraction process by using point cloud serialization along with specific patterns using space-filling curves. Specifically, the 3D point cloud was transformed into a structured format using

the space-filling curves and transforming the position of every 3D point to an integer that represented its order regarding the space-filling curve. During the point cloud serialization, the neighboring points were not changed, i.e., the points' localities were preserved while a structured format was created. Next, for the created representation, the authors proposed the *Patch Attention* mechanism, which was decomposed into *Patch Grouping* and *Patch Interaction* designs. In more detail, the former used *reordering* and *padding* operations based on the serialization pattern and neighboring patches, respectively. The latter described the interaction among points belonging to different patches using different techniques like *Shift Patch*, *Shuffle Order*, etc. Finally, the overall architecture was designed using a U-Net structure. To conclude, the PTv3 architecture was evaluated on different tasks and benchmarks with remarkable results. Finally, the performance of the presented hybrid-based methods in different benchmark datasets is presented in Table 7.

Table 7. Mean Intersection over Union (mIoU) and Overall Accuracy (OA) of different benchmark datasets for hybrid-based methods based on the papers examined in Section 5.5.

Algorithm	Year	Semantic3D		NuScenes		SemanticKITTI		S3DIS		Sensat Urban	
		mIoU	OA	mIoU	OA	mIoU	OA	mIoU	OA	mIoU	OA
PolarNet	2020	-	-	69.4		54.3	90.0	-	-	-	-
TORNADO-Net	2020	-	-	-	-	63.1	90.7	-	-	-	-
UniSeg	2023	-	-	83.5	-	75.2	-	-	-	-	-
JS3C-Net	2020	-	-	73.6	-	66.0	-	-	-	-	-
PVCNN	2019	-	-	-	-	-	-	58.98	-	-	-
SPVConv	2020	-	-	77.4	-	67.0	-	-	-	-	-
LatticeNet	2020	-	-	-	-	52.9	-	-	-	-	-
$(AF)^2$-S3Net	2021	-	-	78.3	-	69.7	-	-	-	-	-
Cylinder3D	2021	-	-	-	-	67.8	-	-	-	-	-
2DPASS	2022	-	-	80.8	-	-	-	-	-	-	-
SPGraph	2018	76.2	94.0	-	-	20.0	-	63.2	86.4	37.29	85.27
KPRNet	2020	-	-	-	-	63.1	-	-	-	-	-
3D-MiniNet	2021	-	-	-	-	55.8	89.7	-	-	-	-
DeepViewAgg	2022	-	-	-	-	-	-	69.5	-	-	-
Meta-RangeSeg	2022	-	-	-	-	61.0	-	-	-	-	-
PTv3	2024	-	-	83.0	-	75.5	-	80.81	-	-	-

-: No Data.

6. Loss Functions in 3D Semantic Segmentation

The loss or objective function is an integral part of deep learning algorithms. During the research conducted on deep learning algorithms in 3DSS, several loss functions have been encountered. For instance, the Boundary loss [125], Consistency loss [128], Contextual loss [129], Contrastive loss [130,131], Dice loss [101], Focal loss [82,97,132–134], and Total Variation loss [109] functions. However, a detailed analysis of the 3DSS loss functions is out of the scope of this effort. Thus, a small part of the loss functions that are mostly used and seem to guarantee their future usage is presented in this section.

Categorical Cross-Entropy Loss: One of the main losses used in 3DSS is the categorical cross-entropy loss, which measures the difference between the predicted and the ground truth value formulated as:

$$L_{ce}(y,\ \hat{y}) = -\frac{1}{N}\sum_{i}^{N}\sum_{j}^{C} y_{ij}\log \hat{y}_{ij} \tag{9}$$

where

- N: Number of samples;
- C: Number of classes
- y : Ground truth label;

- $\hat{y}$: Predicted label.

Weighted Cross-Entropy Loss: In fact, the cross-entropy loss does not take into account the frequency of each class, while most real-world data suffer from the class imbalance problem. Hence, the weighted cross-entropy loss is frequently adopted in 3DSS as a component of the total loss to deal with the class imbalance problem using the frequency of each class.

$$L_{wce}(y,\ \hat{y}) = -\frac{1}{N}\sum_{i}^{N}\sum_{j}^{C} w_n y_{ij} \log \hat{y}_{ij} \tag{10}$$

where

- N: Number of samples;
- C: Number of classes;
- w_n: Class weight;
- y : Ground truth label;
- $\hat{y}$: Predicted label.

Several methods [93,94,109,125,135] mention that the weight for each class is commonly calculated by using the $\frac{1}{\sqrt{f_j}}$ formula based on the frequency of each class, avoiding the manual definition of the weights.

Geo-Aware Anisotropic Loss: Liu et al. (2020) presented a metric that takes into account the variability between the semantic classes of the current voxel and its neighboring voxels. To be more specific, the authors referred to this metric as Local Geometric Anisotropy, which takes higher values when a voxel at the edge of the semantic category is under investigation, i.e., when the neighboring voxels have a different category than the under-investigation voxel. Overall, LGA is a factor that quantifies the significance of the voxel position and was included in the PA loss proposed by the authors. In 3DSS, LGA is used to recover the fine details of a 3D point cloud and, in combination with other losses, to form a total loss [116,136]. Commonly, LGA is included in the geo-aware anisotropic loss, formulated as follows:

$$L_{geo}(y,\ \hat{y}) = -\frac{1}{N}\sum_{i,j,k}\sum_{c=1}^{C} \frac{M_{LGA}}{\Phi} y_{ijk,c} \log \hat{y}_{ijk,c} \tag{11}$$

where

- N: voxel neighborhood located at I, j, and k;
- c: Current class of classes C;
- M_{LGA} : Local Geometric Anisotropy metric;
- y : Ground truth label;
- $\hat{y}$: Predicted label;
- Φ: Sliding window.

Lovasz–SoftMax loss: In general, the most common evaluation metric in 2D and 3D semantic segmentation is the mean Intersection over Union score (mIoU) or the Jaccard Index. Berman, Triki, and Blaschko (2018) [137] proposed the Lovasz–SoftMax loss to maximize the IoU score. Commonly, the Lovasz–SoftMax loss is included in 3DSS methods to take into account the fine details presented in the data [91,93,94,109,125]. However, the Lovasz–SoftMax loss does not consider the points' neighborhood during the computation, which can lead to noise predictions [109]. The Lovasz–SoftMax loss is formulated as follows:

$$L_{ls} = \frac{1}{|C|}\sum_{c\ \in C} \overline{\Delta_{Jc}}(m(c)),\ \ and\ \ m_i = \begin{cases} 1 - x_i(c) & if \quad c = y_i(c) \\ x_{(i)}(c) & otherwise \end{cases} \tag{12}$$

where

- $|C|$: Number of classes;
- $\overline{\Delta_{Jc}}$: Lovasz extension of the Jaccard Index;
- $x_{(i)}(c) \in [0,\ 1]$: Predicted probability of point I for class c;
- $y_i(c)$: The ground truth label.

7. Discussion

In this section, an informative analysis of the examined 3DSS algorithms and datasets is presented to foster new research directions and applications in the field of 3DSS. First and foremost, a timeline of the included 3DSS methods is presented in Figure 18. Then, the included methods along with their code implementation are summarized in Table 8.

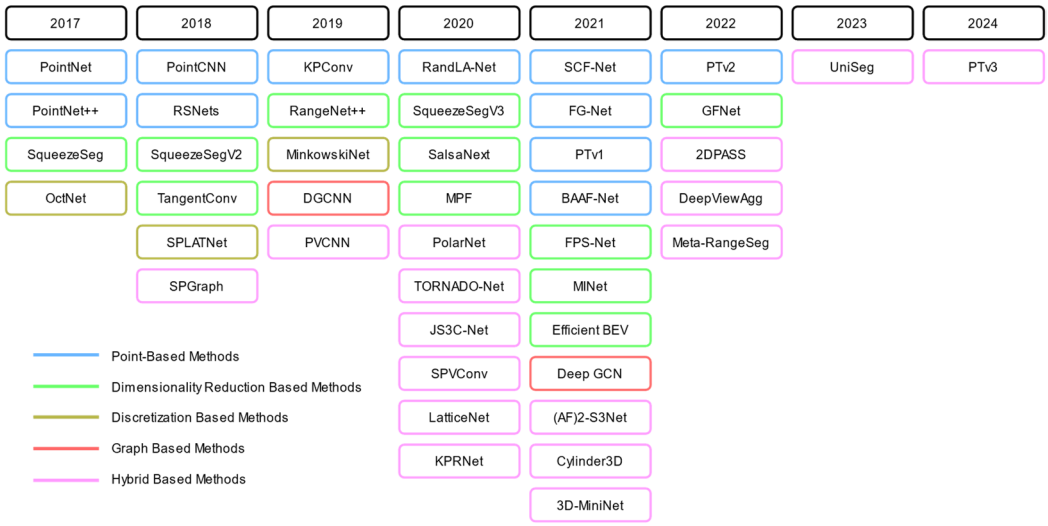


Figure 18. The timeline of the methods included. Point-based methods (blue), dimensionality reduction-based methods (green), discretization-based methods (yellow), graph-based methods (Orange), and hybrid-based methods (pink).

Table 8. The included methods for 3DSS along with their code implementations. Point-based (PB), dimensionality reduction-based (DRB), hybrid-based (HB), graph-based (GB), and discretization-based (DB).

Algorithm	Category	Year	Code
PointNet	PB	2017	https://github.com/charlesq34/pointnet
PointNet++	PB	2017	https://github.com/charlesq34/pointnet2
SqueezeSeg	DRB	2017	https://github.com/BichenWuUCB/SqueezeSeg
OctNet	DB	2017	https://github.com/griegler/octnet
PointCNN	PB	2018	https://github.com/yangyanli/PointCNN
RSNets	PB	2018	-
SqueezeSegV2	DRB	2018	-
TangenConv	DRB	2018	-
SPLATNet	DB	2018	https://suhangpro.github.io/splatnet/
SPGRaph	HB	2018	https://github.com/loicland/superpoint_graph
KPConv	PB	2019	https://github.com/HuguesTHOMAS/KPConv
RangeNet++	DRB	2019	https://github.com/PRBonn/lidar-bonnetal
MinkowskiNet	DB	2019	https://github.com/StanfordVL/MinkowskiEngine
DGCNN	GB	2019	https://github.com/WangYueFt/dgcnn
PVCNN	HB	2019	https://github.com/mit-han-lab/pvcnn
RandLA-Net	PB	2020	https://github.com/QingyongHu/RandLA-Net https://github.com/aRI0U/RandLA-Net-pytorch
SqueezeSegV3	DRB	2020	https://github.com/chenfengxu714/SqueezeSegV3

Table 8. *Cont.*

Algorithm	Category	Year	Code
SalsaNext	DRB	2020	https://github.com/TiagoCortinhal/SalsaNext
MPF	DRB	2020	-
PolarNet	HB	2020	https://github.com/edwardzhou130/PolarSeg
TORNADO-Net	HB	2020	-
JS3C-Net	HB	2020	https://github.com/yanx27/JS3C-Net
SPVConv	HB	2020	https://github.com/mit-han-lab/torchsparse https://github.com/mit-han-lab/spvnas
LatticeNet	HB	2020	https://github.com/AIS-Bonn/latticenet
KPRNet	HB	2020	https://github.com/DeyvidKochanov-TomTom/kprnet
SCF-Net	PB	2021	https://github.com/leofansq/SCF-Net
FG-Net	PB	2021	https://github.com/KangchengLiu/Feature-Geometric-Net-FG-Net
PTv1	PB	2021	https://github.com/POSTECH-CVLab/point-transformer (Unofficial) https://github.com/Pointcept/Pointcept
BAAF-Net	PB	2021	https://github.Com/ShiQiu0419/BAAF-Net
FPS-Net	DRB	2021	https://github.com/xiaoaoran/FPS-Net
MINet	DRB	2021	https://github.com/sj-li/MINet
Efficient BEV	DRB	2021	-
DeepGCN	GB	2021	https://github.com/lightaime/deep_gcns_torch https://github.com/lightaime/deep_gcns
$(AF)^2$_S3Net	HB	2021	-
Cylinder3D	HB	2021	https://github.com/xinge008/Cylinder3D
3D-MiniNet	HB	2021	https://sites.google.com/a/unizar.es/semanticseg/
PTv2	PB	2022	https://github.com/Gofinge/PointTransformerV2 https://github.com/Pointcept/Pointcept
GFNet	DRB	2022	https://github.com/haibo-qiu/GFNet
2DPASS	HB	2022	https://github.com/yanx27/2DPASS
DeepViewAgg	HB	2022	https://github.com/drprojects/DeepViewAgg
Meta-RangeSeg	HB	2022	https://github.com/songw-zju/Meta-RangeSeg
UniSeg	HB	2023	https://github.com/PJLab-ADG/PCSeg
PTv3	HB	2024	https://github.com/Pointcept/PointTransformerV3

All links last accessed: 30 December 2024.

In Section 2, an analysis of previous review papers is performed to define a unified taxonomy scheme for 3DSS methods. Regardless of the category in which the 3DSS algorithms belong, i.e., point-, dimensionality reduction-, discretization-, graph-, and hybrid-based, they aim to handle similar issues, like computational cost and memory consumption, information loss during the algorithm execution, the properties of the 3D point cloud, e.g., sparsity and irregularity, the introduction of a novel feature extraction strategy, etc. However, based on their category, the advantages and disadvantages of them as long as the new research directions are unique.

7.1. Regarding the 3DSS Method Category

Specifically, point-based methods are applied directly on a given 3D point cloud to extract meaningful 3DSS point features; thus, there is no need for extra computational time to create an intermediate representation of the data. To this end, the 3D spatial structure information of the data is preserved without losing the detailed geometric information [125]. A crucial part of 3DSS using point-based methods is to preserve the detailed geometric information of the data during the execution of the algorithm. Regarding this task, the sampling method used to define each neighborhood centroid point is one of the main parts. The most common sampling method is Farthest Point Sampling (FPS) [68,69,78,138].

However, several approaches mentioned that FPS is time-consuming [70,74,139], while other sampling methods like Random Sampling (RS) can reduce the execution time while preserving high-end results. In fact, RS has also some drawbacks due to the random picking of points, e.g., to dropout useful point features [70]. Hence, an investigation of the new strategies for point sampling to improve the efficiency and efficacy of point-based methods is recommended. In general, [70] presented an in-depth analysis of many sampling methods and, thus, it could be used as a source.

Furthermore, the definition of the points' neighborhood plays a significant role in the preservation of the local information of the point cloud as well as the execution time of the algorithm. The most common approaches to define the neighborhood of points is the k nearest neighbor (kNN) and the Ball Query algorithms. In general, point-based methods spend most of the execution time in defining the points' neighborhood, i.e., handling the random memory access or, in other words, extracting the point neighbors into the continuous space, rather than extracting features [95,104,113,114]. Frequently, point-based methods are characterized as unreliable and time-consuming, especially for large-scale point clouds due to the neighborhood definition step [1,74,90,91,94,116,125,140,141]. In fact, point-based methods are applied on large-scale point clouds using point cloud partitioning approaches. However, the partitioning operation harms the global consistency of the point cloud, resulting in weaker global features [142]. Furthermore, the kNN algorithm is not sufficient for modeling the local neighborhood of points [143]. Thus, efficient techniques on the definition of the points' neighborhood should be investigated in order to alleviate the issue of applying point-based methods on large-scale point clouds [144]. Finally, a more efficient definition of the points' neighborhood will further improve the preservation of the local information encapsulated into the points' neighborhoods and thus enable the application of point-based 3DSS methods in more detailed applications, e.g., point cloud serialization.

Moreover, the local feature aggregation method, which is applied on the points' neighborhood to transform 3DSS local features, is crucial for the preservation of the local information. In fact, the extraction of the local information from the points' neighborhood is demanding due to the point cloud properties and the lack of explicit relationship among the 3D points [145]. Furthermore, the existing 3DSS approaches do not thoroughly capture the local information [51,76,146]. In general, the local feature aggregation operations of point-based 3DSS algorithms are commonly used in maximum or summation operation. However, these operations tend to lose the information of the geometric structures [147]. Even more deeply, the way that the local aggregation is applied on each category affects the performance of the model. In general, the existing 3DSS algorithms process all categories using the same aggregation operation, which could result in confusion among similar categories [148,149]. Additionally, the complex details presented in the point clouds lead to pattern imbalances, either between the categories, resulting in the developed algorithms learning only the dominant cases [150], or inside the same category, resulting in a reduction in the model's performance [151]. Overall, an improvement in the local aggregation operators will positively affect diversely the point-based 3DSS algorithms, and thus, an investigation of this part is recommended. In general, [152] presented an in-depth analysis of local aggregation operators, in which they surprisingly concluded that sophisticated local aggregation operators performed similarly to the conventional ones using the same residual network. Hence, the analysis could be used as a source for further exploration.

Furthermore, most 3DSS algorithms investigate local feature extraction, paying less attention to global features, which are quite important for the performance of 3DSS algorithms [153–156]. Thus, an investigation of the global feature extraction process could

be performed to improve the performance of 3DSS algorithms. In addition to local and global feature extraction, multi-scale feature extraction is also investigated, aiming to find multi-resolution information and thus improve the performance of 3DSS methods [157,158]. Finally, point-based methods require less GPU memory in comparison to discretization-based methods [113], while they achieve high-end results in many 3D tasks [90,152].

In general, dimensionality reduction-based (DRB) methods project the given point cloud into a lower dimensional space, perform the semantic segmentation of the data in that space, and then re-project it into the 3D space. Using the aforementioned methodology, DRB methods try to alleviate the unstructured and irregularity properties of 3D point clouds by constructing a regular representation of the data.

However, the creation of such a representation costs extra computation time and also results in the loss of information, e.g., the geometric information and the details of the point cloud, due to the projection of the 3D data in the lower dimension space [1,70,73,74,78,87,97,116,118,136,152,159]. Despite the extra computation time required for the creation of the new representation, DRB methods overcome the expensive 3D computations presented in point-based and graph-based methods [107]. Besides, the projection process alters and abandons the 3D topology and the geometric relations of the data [118,125,136,160]. However, DRB methods could benefit from mature 2DSS algorithms and 2DSS datasets and, thus, follow their advancements [95,97,107]. Overall, DRB methods seems to be fast, but do not exploit the benefits of 3D data [91,123].

Moreover, DRB methods require post-processing steps to re-project the semantic information gathered into the lower space to the 3D space. In fact, the performance of 3DSS is hampered due to the smoothed labels that are generated in the lower dimension space and re-projected into the 3D space [87,122]; thus, a refinement technique is commonly applied to achieve high-end results. This problem was formulated as the label re-projection problem, while the first approaches tried to alleviate it using conditional random fields (CRFs) [82]. In general, the most common approach that handles the label re-projection problem is that presented in RangeNet++ using the kNN algorithm [87]. Recently, an alternative post-processing solution to avoid the kNN approach was presented by exploiting the KPConv [73] network [2,122]. In general, new post-processing techniques to alleviate the label re-projection problem could be explored.

In fact, DRB methods stack several channels to create multi-channel range images from 3D point cloud data. Commonly, the created images contain the x, y, z, range, and remission or intensity channels. However, the feature distribution of such images varies and is different from RGB images. Xu et al. (2020) [90] mentioned that the convolution operator performs poorly using spatially varying feature images. Additionally, Xiao et al. (2021) [93] stated that each channel has unique characteristics and thus should be treated differently. In general, an investigation of how the developed algorithm will fully exploit the information presented in the generated multi-channel images of DRB methods will be beneficial.

Moreover, DRB methods most commonly use the spherical projection of 3D data. However, there are other projections, like the bird's-eye view (BEV), which are exploited to perform 3DSS. Jiang et al. (2023) [161] stated that the methods that exploit the BEV projection could achieve real-time inference on 3DSS. Recently, a combination of such projections was used to perform 3DSS, mentioning that each one contains complementary information to the other [2,97]. Thus, a further exploration of projections in order to fully exploit the 3D information of point clouds could be performed, e.g., using scan unfolding [122].

In fact, point-based methods and discretization-based (DB) methods have some similarities regarding the feature extraction process, i.e., both extract features using the

points' neighborhood. However, the neighborhood definition is different, i.e., using kNN or voxels. DB methods transform an unstructured point cloud into a regular representation, e.g., voxel grid, on which the well-known convolution operation could be applied. However, the creation of a regular grid is a time-consuming operation due to the cubic growth of voxels with respect to the density of the given point cloud and the requirements of the voxel grid resolution [1,2,70,72,74,78,80,87,88,97,102,104,111,113,114,152,159,162,163]. Additionally, the quantization of the given point cloud results in a loss of information, especially when points of different categories are included in the voxels or a large-scale point cloud is used [1,74,93,105,113,114,152,159,164]. Mainly, the computation cost of DB methods is due to the large amount of empty voxels, i.e., of the empty space, involved in the computations. Thus, sparse structures like octrees, kd-trees, and hash-maps [67,73,80,88,102,104,162] and post-processing techniques like CRFs [67,80] were introduced to alleviate the aforementioned problem. However, the convolution kernels that are exploited by sparse voxel-based methods are small and thus harm the network's performance [165]. Furthermore, voxel-based methods have good memory locality, i.e., preserving the spatial relationship due to the regular grid representation [74,113]. To conclude, DB methods are commonly applied separately but also in combination with other methods, e.g., point-based, forming hybrid architectures. A further investigation of the hybrid architecture exploiting the advantages of DB methods is recommended. Additionally, high-dimensional lattices could be exploited as a base for the exploration of multi-modal 3DSS.

In general, graph-based (GB) methods aim to create a new graph representation and then to extract 3D features by exploiting the convolution operation. However, the creation of such representation is time-consuming [74]. Additionally, GB methods are commonly limited to very shallow models [106]. However, the graph representation of the data preserves the local geometry of the data while it can capture complex shapes. Moreover, the edges could be used to carry features on them [105]. Finally, the graph representation could be exploited differently than the 3D Euclidean space, e.g., feature space, [105] which is advantageous for the creation of hybrid methods.

The four main categories of 3DSS methods, i.e., point, dimensionality reduction, discretization, and graph, are commonly combined to form architectures that leverage the advantages of each category and complement the disadvantages of them. In general, the receptive field plays a significant role in the performance of both 2D and 3D architectures. However, in 3D space, the shape, apart from the size of the receptive field, is also important [104,107]. Hence, several hybrid approaches that exploit discretization-based methods tried to define voxels that follow the distribution of LiDAR points, e.g., using polar grid [107], cylindrical voxel shapes with different sizes depending on the distance of the 3D point [118], or crisscross shape voxels [104]. In fact, LiDAR-based methods dominate the research on 3DSS. Hence, different shaped voxels follow the acquisition of LiDAR point clouds. A further exploration of the voxel shape regarding other types is of interest.

Moreover, discretization-based and point-based methods are commonly combined to leverage the efficient memory locality and the extraction of fine-grained details, respectively [113,114,116,118], using two branch feature extraction approaches. Different fusion strategies, e.g., early or late fusion, could be explored using features gathered using different approaches.

Additionally, the complementarity among different data modalities is also explored, mainly using a combination of images and 3D point clouds [119,124]. The general idea is that the images contain better textural and contextual information than 3D point clouds. To conclude, the complementarity among the different categories of 3DSS methods should be further explored to advance the performance and efficiency of 3DSS algorithms.

7.2. Regarding the Datasets and the Data

In general, most 3DSS algorithms are trained using supervised learning, i.e., having available 3D ground truth information. In Section 3, several 3DSS datasets for indoor and outdoor environments are presented. In fact, most of the datasets were created using an RGB-D camera and a LiDAR sensor in indoor and outdoor environments, respectively. Additionally, even the available datasets use the same sensors, like the Matterport camera (indoor) and Velodyne LiDAR (outdoor). On the one hand, the combination of datasets is more straightforward due to the same data acquisition sensor type [166]. On the other hand, datasets with a different acquisition method, e.g., using photogrammetric approaches or different sensors, should be released. To be more specific, the 3D point clouds created using a LiDAR sensor are strongly differ from those created using photogrammetry or terrestrial laser scanners with respect to point density and point distribution. Thus, more datasets using such techniques should be released to explore the 3DSS concept. In Figure 19 an example of an annotated indoor and outdoor dataset is presented.

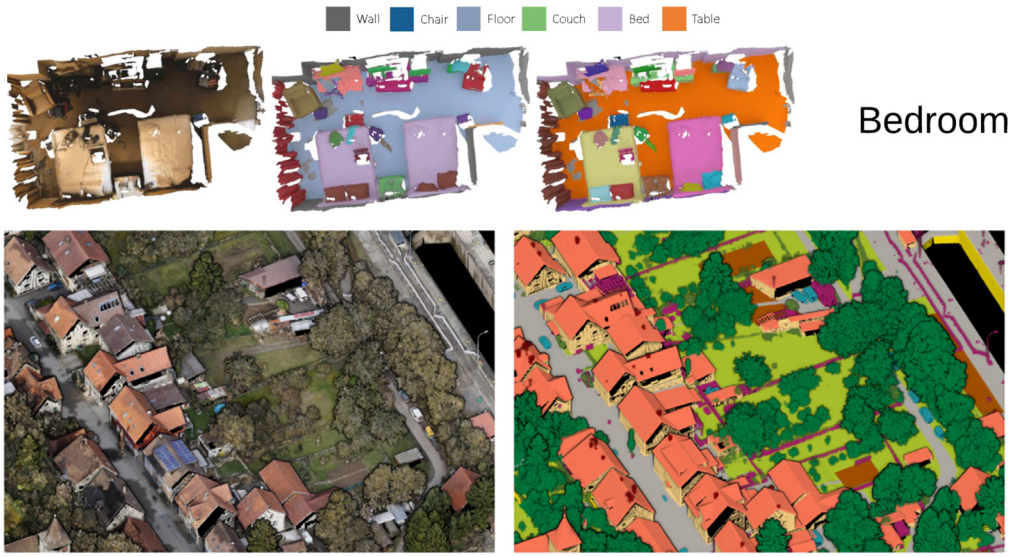


Figure 19. Labeled point clouds in indoor (ScanNet dataset [28]) and outdoor (Hessigheim dataset [46]) environments.

Additionally, there are not so many 3DSS datasets that explore multimodality. Specifically, multimodal 3DSS segmentation should be explored with regard to using complementary information with different types of data to improve 3DSS performance, e.g., using images in combination with 3D point clouds or other types of data like text. The aforementioned problem is also referred as the sensory gap problem [9]. However, the aforementioned exploration should be assisted with well-constructed benchmark datasets, which currently are lacking.

Furthermore, in order for 3DSS algorithms to achieve better generalization, data from different geographic locations should be available. In fact, the 3DSS datasets do not span different geographic locations [166]. Thus, datasets from different places on earth should be released to advance the interpretation of 3DSS algorithms.

Moreover, a common problem regarding the datasets is the class imbalance problem [91,93,95]. More concretely, the number of points in each category significantly differs from each other, e.g., the class motorcycle will have fewer points than the class street or pavement, resulting in the underperformance of 3DSS on these classes. Additionally, the intra class geometry between the instances of each class plays a significant role in the performance of 3DSS algorithms. A further exploration of the class-imbalance problem and

an investigation of the differences in the intra-class geometry is recommended to improve the performance of 3DSS algorithms on these categories.

In general, the available 3DSS datasets offer static information, i.e., specific temporal information. Hence, the creation of spatio-temporal datasets of the same scene will be beneficial for several applications like constructions inspection, etc.

Finally, application-oriented datasets, such as construction, cultural heritage, climate change, etc., are important to create to advance 3DSS performance in each application. However, this process is time-consuming and tedious and requires experts for the labeling annotation process. A counterpart of the creation of real application-oriented datasets is the creation of application-oriented synthetic datasets. However, there are limitations due to the domain shift problem, i.e., different acquisition circumstances between the real and the synthetic data.

7.3. Regarding the Application

In general, the application plays a significant role in the characteristics of using the 3DSS algorithm. In more detail, some applications need real-time 3DSS algorithms, e.g., in an autonomous driving scenario. Moreover, other applications need lightweight architectures, e.g., embedded systems, while other applications require unique implementations due to the large scale of the available data, e.g., the urban scale 3DSS. Therefore, a further exploration of methods or the advancement of existing methods, regarding a specific application, is recommended.

7.4. Regarding the Learning Approach

In fact, most of the available 3DSS methods use a supervised learning approach, i.e., they exploit available 3D ground-truth data to train the algorithms. However, the creation of such data is a time-consuming and tedious process and is commonly referred as the data-hungry problem [9], i.e., the existing algorithms require a large amount of fine annotated data to be trained on [52,86,167,168]. Additionally, the creation of manual annotated data is error-prone, especially in the 3D domain [169,170]. To this end, other learning approaches, like semi-supervised, weakly supervised, few/zero-shot learning, or even un-supervised learning, have been proposed to alleviate the aforementioned limitation [168,171]. To be more specific, in 3D space, the exploitation of these learning techniques is crucial regarding the difficulty in finding application-oriented datasets. Additionally, such techniques could be used to enrich the available datasets with new classes and unseen objects, i.e., the open-set problem. Moreover, weaker types of annotations have been released, such as scribble annotations [128], to define an easier way to create annotated datasets. Furthermore, continual learning techniques could be adopted to further improve the interpretation of 3DSS algorithms. To this end, the exploration of extracting further knowledge about a scene using the available information could also be explored, commonly referred as the semantic gap problem [9]. Therefore, the exploration of methods different from the supervised learning techniques for 3DSS is recommended to alleviate the data-hungry problem and to define open-set 3DSS approaches.

8. Conclusions

In this effort, we surveyed in detail many of the most notable deep learning 3DSS algorithms and datasets of the previous years. Firstly, based on the analysis of the existing methods, along with the investigation of previous review papers, we proposed a unified taxonomy scheme. In detail, we categorized 3DSS methods into point-, dimensionality reduction-, discretization-, graph-, and hybrid-based methods. Additionally, we presented a thorough review of the existing indoor and outdoor 3DSS benchmarks, along with

their characteristics and evaluation metrics. Moreover, a non-exhaustive but informative presentation of the loss functions used in 3DSS was included. Furthermore, a fruitful discussion of the examined 3DSS algorithms and datasets was presented to foster new research directions and applications in the field of 3DSS. Finally, a GitHub repository (Supplementary Materials: https://github.com/thobet/Deep-Learning-on-3D-Semantic-Segmentation-a-Detailed-Review, accessed on 30 December 2024), which includes an initial classification of over 400 3DSS algorithms, was included.

Supplementary Materials: A GitHub repository (https://github.com/thobet/Deep-Learning-on-3D-Semantic-Segmentation-a-Detailed-Review, accessed on 30 December 2024) was created to store an initial classification of over 400 3DSS papers using the proposed taxonomy scheme, which are not all referenced here.

Author Contributions: Conceptualization, T.B. and A.G.; methodology, T.B. and A.G.; investigation, T.B.; writing—original draft preparation, T.B.; writing—review and editing, T.B, A.G, A.D. and P.G.; supervision, A.G. and A.D.; funding acquisition, A.G. All authors have read and agreed to the published version of the manuscript.

Funding: The publication of this paper was partially funded by NTUA Research Committee project number 950368 entitled "INVESTIGATION OF 3D MODELING PROCESSES FOR THE RECORDING, RESTORATION AND PRESERVATION OF MONUMENTS".

Acknowledgments: This manuscript was awarded the Laura Bassi Scholarship—Spring 2024, by the Bassi Foundation and Editing Press, as part of the research of the first author towards his PhD at NTUA.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Fan, S.; Dong, Q.; Zhu, F.; Lv, Y.; Ye, P.; Wang, F.-Y. SCF-Net: Learning Spatial Contextual Features for Large-Scale Point Cloud Segmentation. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 14499–14508.
2. Qiu, H.; Yu, B.; Tao, D. GFNet: Geometric Flow Network for 3D Point Cloud Semantic Segmentation. *arXiv* **2022**, arXiv:2207.02605.
3. Weinmann, M.; Schmidt, A.; Mallet, C.; Hinz, S.; Rottensteiner, F.; Jutzi, B. Contextual Classification of Point Cloud Data by Exploiting Individual 3d Neigbourhoods. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2015**, *II-3/W4*, 271–278. [CrossRef]
4. Poux, F.; Neuville, R.; Billen, R. Point Cloud Classification of Tesserae from Terrestrial Laser Data Combined with Dense Image Matching for Archaeological Information Extraction. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2017**, *IV-2/W2*, 203–211. [CrossRef]
5. Weinmann, M.; Jutzi, B.; Hinz, S.; Mallet, C. Semantic Point Cloud Interpretation Based on Optimal Neighborhoods, Relevant Features and Efficient Classifiers. *ISPRS J. Photogramm. Remote Sens.* **2015**, *105*, 286–304. [CrossRef]
6. Sevgen, E.; Abdikan, S. Classification of Large-Scale Mobile Laser Scanning Data in Urban Area with LightGBM. *Remote Sens.* **2023**, *15*, 3787. [CrossRef]
7. Bello, S.A.; Yu, S.; Wang, C. Review: Deep Learning on 3D Point Clouds. *Remote Sens.* **2020**, *12*, 1729. [CrossRef]
8. Camuffo, E.; Mari, D.; Milani, S. Recent Advancements in Learning Algorithms for Point Clouds: An Updated Overview. *Sensors* **2022**, *22*, 1357. [CrossRef]
9. Gao, B.; Pan, Y.; Li, C.; Geng, S.; Zhao, H. Are We Hungry for 3D LiDAR Data for Semantic Segmentation? A Survey of Datasets and Methods. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 6063–6081. [CrossRef]
10. Griffiths, D.; Boehm, J. A Review on Deep Learning Techniques for 3D Sensed Data Classification. *Remote Sens.* **2019**, *11*, 1499. [CrossRef]
11. Grilli, E.; Menna, F.; Remondino, F. A Review of point Clouds Segmentation and Classification Algorithms. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2017**, *XLII-2/W3*, 339–344. [CrossRef]
12. Guo, Y.; Wang, H.; Hu, Q.; Liu, H.; Liu, L.; Bennamoun, M. Deep Learning for 3D Point Clouds: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *43*, 4338–4364. [CrossRef] [PubMed]
13. Jhaldiyal, A.; Chaudhary, N. Semantic Segmentation of 3D LiDAR Data Using Deep Learning: A Review of Projection-Based Methods. *Appl. Intell.* **2023**, *53*, 6844–6855. [CrossRef]
14. Rauch, L.; Braml, T. Semantic Point Cloud Segmentation with Deep-Learning-Based Approaches for the Construction Industry: A Survey. *Appl. Sci.* **2023**, *13*, 9146. [CrossRef]

15. Rizzoli, G.; Barbato, F.; Zanuttigh, P. Multimodal Semantic Segmentation in Autonomous Driving: A Review of Current Approaches and Future Perspectives. *Technologies* **2022**, *10*, 90. [CrossRef]
16. Xiao, A.; Zhang, X.; Shao, L.; Lu, S. A Survey of Label-Efficient Deep Learning for 3D Point Clouds 2023. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*; IEEE: Piscataway, NJ, USA, 2024.
17. Xiao, A.; Huang, J.; Guan, D.; Zhang, X.; Lu, S.; Shao, L. Unsupervised Point Cloud Representation Learning with Deep Neural Networks: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 11321–11339. [CrossRef]
18. Xie, Y.; Tian, J.; Zhu, X.X. Linking Points With Labels in 3D: A Review of Point Cloud Semantic Segmentation. *IEEE Geosci. Remote Sens. Mag.* **2020**, *8*, 38–59. [CrossRef]
19. Yang, S.; Hou, M.; Li, S. Three-Dimensional Point Cloud Semantic Segmentation for Cultural Heritage: A Comprehensive Review. *Remote Sens.* **2023**, *15*, 548. [CrossRef]
20. Zhang, A.; Li, S.; Wu, J.; Li, S.; Zhang, B. Exploring Semantic Information Extraction From Different Data Forms in 3D Point Cloud Semantic Segmentation. *IEEE Access* **2023**, *11*, 61929–61949. [CrossRef]
21. Zhang, H.; Wang, C.; Tian, S.; Lu, B.; Zhang, L.; Ning, X.; Bai, X. Deep Learning-Based 3D Point Cloud Classification: A Systematic Survey and Outlook. *Displays* **2023**, *79*, 102456. [CrossRef]
22. Zhang, J.; Zhao, X.; Chen, Z.; Lu, Z. A Review of Deep Learning-Based Semantic Segmentation for Point Cloud. *IEEE Access* **2019**, *7*, 179118–179133. [CrossRef]
23. Zhang, R.; Wu, Y.; Jin, W.; Meng, X. Deep-Learning-Based Point Cloud Semantic Segmentation: A Survey. *Electronics* **2023**, *12*, 3642. [CrossRef]
24. Armeni, I.; Sener, O.; Zamir, A.R.; Jiang, H.; Brilakis, I.; Fischer, M.; Savarese, S. 3D Semantic Parsing of Large-Scale Indoor Spaces. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 1534–1543.
25. Armeni, I.; Sax, S.; Zamir, A.R.; Savarese, S. Joint 2D-3D-Semantic Data for Indoor Scene Understanding. *arXiv* **2017**, arXiv:1702.01105.
26. Freiburg Campus 360 Degree 3D Scans—Arbeitsgruppe: Autonome Intelligente Systeme. Available online: http://ais.informatik.uni-freiburg.de/projects/datasets/fr360/ (accessed on 17 March 2023).
27. Chang, A.; Dai, A.; Funkhouser, T.; Halber, M.; Nießner, M.; Savva, M.; Song, S.; Zeng, A.; Zhang, Y. Matterport3D: Learning from RGB-D Data in Indoor Environments. *arXiv* **2017**, arXiv:1709.06158.
28. Dai, A.; Chang, A.X.; Savva, M.; Halber, M.; Funkhouser, T.; Nießner, M. Scannet: Richly-Annotated 3d Reconstructions of Indoor Scenes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 5828–5839.
29. RGB-D Scenes Dataset v.2. Available online: http://rgbd-dataset.cs.washington.edu/dataset/rgbd-scenes-v2/ (accessed on 4 October 2023).
30. McCormac, J.; Handa, A.; Leutenegger, S.; Davison, A.J. SceneNet RGB-D: 5M Photorealistic Images of Synthetic Indoor Trajectories with Ground Truth. *arXiv* **2016**, arXiv:1612.05079.
31. Hua, B.-S.; Pham, Q.-H.; Nguyen, D.T.; Tran, M.-K.; Yu, L.-F.; Yeung, S.-K. Scenenn: A Scene Meshes Dataset with Annotations. In Proceedings of the 2016 Fourth International Conference on 3D Vision (3DV), Stanford, CA, USA, 25–28 October 2016; pp. 92–101.
32. Song, S.; Lichtenberg, S.P.; Xiao, J. SUN RGB-D: A RGB-D Scene Understanding Benchmark Suite. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 567–576.
33. Xiao, J.; Owens, A.; Torralba, A. SUN3D: A Database of Big Spaces Reconstructed Using SfM and Object Labels. In Proceedings of the 2013 IEEE International Conference on Computer Vision, Sydney, Australia, 1–8 December 2013; pp. 1625–1632.
34. Straub, J.; Whelan, T.; Ma, L.; Chen, Y.; Wijmans, E.; Green, S.; Engel, J.J.; Mur-Artal, R.; Ren, C.; Verma, S.; et al. The Replica Dataset: A Digital Replica of Indoor Spaces. *arXiv* **2019**, arXiv:1906.05797.
35. Martínez-Gómez, J.; García-Varea, I.; Cazorla, M.; Morell, V. ViDRILO: The Visual and Depth Robot Indoor Localization with Objects Information Dataset. *Int. J. Robot. Res.* **2015**, *34*, 1681–1687. [CrossRef]
36. Behley, J.; Steinhage, V.; Cremers, A.B. Performance of Histogram Descriptors for the Classification of 3D Laser Range Data in Urban Environments. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation, St Paul, MN, USA, 14–18 May 2012; pp. 4391–4398.
37. Can, G.; Mantegazza, D.; Abbate, G.; Chappuis, S.; Giusti, A. Semantic segmentation on Swiss3DCities: A benchmark study on aerial photogrammetric 3D pointcloud dataset. *Pattern Recognit. Lett.* **2021**, *150*, 108–114. [CrossRef]
38. Deschaud, J.-E.; Duque, D.; Richa, J.P.; Velasco-Forero, S.; Marcotegui, B.; Goulette, F. Paris-CARLA-3D: A Real and Synthetic Outdoor Point Cloud Dataset for Challenging Tasks in 3D Mapping. *Remote Sens.* **2021**, *13*, 4713. [CrossRef]
39. Fang, J.; Zhou, D.; Zhao, J.; Tang, C.; Xu, C.-Z.; Zhang, L. LiDAR-CS Dataset: LiDAR Point Cloud Dataset with Cross-Sensors for 3D Object Detection. *arXiv* **2023**, arXiv:2301.12515.

40. Fuentes Reyes, M.; Xie, Y.; Yuan, X.; d'Angelo, P.; Kurz, F.; Cerra, D.; Tian, J. A 2D/3D Multimodal Data Simulation Approach with Applications on Urban Semantic Segmentation, Building Extraction and Change Detection. *ISPRS J. Photogramm. Remote Sens.* **2023**, *205*, 74–97. [CrossRef]
41. Hackel, T.; Savinov, N.; Ladicky, L.; Wegner, J.D.; Schindler, K.; Pollefeys, M. Semantic3D.Net: A New Large-Scale Point Cloud Classification Benchmark. *arXiv* **2017**, arXiv:1704.03847. [CrossRef]
42. Hu, Q.; Yang, B.; Khalid, S.; Xiao, W.; Trigoni, N.; Markham, A. Towards Semantic Segmentation of Urban-Scale 3D Point Clouds: A Dataset, Benchmarks and Challenges. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021.
43. Ibrahim, M.; Akhtar, N.; Anwar, S.; Mian, A. SAT3D: Slot Attention Transformer for 3D Point Cloud Semantic Segmentation. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 5456–5466. [CrossRef]
44. Jiang, P.; Osteen, P.; Wigness, M.; Saripalli, S. RELLIS-3D Dataset: Data, Benchmarks and Analysis. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021.
45. Klokov, A.A.; Pak, D.U.; Khorin, A.; Yudin, D.A.; Kochiev, L.; Luchinskiy, V.D.; Bezuglyj, V.D. DAPS3D: Domain Adaptive Projective Segmentation of 3D LiDAR Point Clouds. *IEEE Access* **2023**, *11*, 79341–79356. [CrossRef]
46. Kölle, M.; Laupheimer, D.; Schmohl, S.; Haala, N.; Rottensteiner, F.; Wegner, J.D.; Ledoux, H. The Hessigheim 3D (H3D) Benchmark on Semantic Segmentation of High-Resolution 3D Point Clouds and Textured Meshes from UAV LiDAR and Multi-View-Stereo. *ISPRS Open J. Photogramm. Remote Sens.* **2021**, *1*, 100001. [CrossRef]
47. Li, X.; Li, C.; Tong, Z.; Lim, A.; Yuan, J.; Wu, Y.; Tang, J.; Huang, R. Campus3D: A Photogrammetry Point Cloud Benchmark for Hierarchical Understanding of Outdoor Scene. In Proceedings of the 28th ACM International Conference on Multimedia, Seattle, WA, USA, 12–16 October 2020; pp. 238–246.
48. Loiseau, R.; Vincent, E.; Aubry, M.; Landrieu, L. Learnable Earth Parser: Discovering 3D Prototypes in Aerial Scans. *arXiv* **2023**, arXiv:2304.09704.
49. Qin, N.; Tan, W.; Ma, L.; Zhang, D.; Li, J. OpenGF: An Ultra-Large-Scale Ground Filtering Dataset Built Upon Open ALS Point Clouds Around the World. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021.
50. Roynard, X.; Deschaud, J.-E.; Goulette, F. Paris-Lille-3D: A Large and High-Quality Ground-Truth Urban Point Cloud Dataset for Automatic Segmentation and Classification. *Int. J. Robot. Res.* **2018**, *37*, 545–557. [CrossRef]
51. Su, Y.; Liu, W.; Yuan, Z.; Cheng, M.; Zhang, Z.; Shen, X.; Wang, C. DLA-Net: Learning Dual Local Attention Features for Semantic Segmentation of Large-Scale Building Facade Point Clouds. *Pattern Recognit.* **2021**, *123*, 108372. [CrossRef]
52. Tan, W.; Qin, N.; Ma, L.; Li, Y.; Du, J.; Cai, G.; Yang, K.; Li, J. Toronto-3D: A Large-Scale Mobile LiDAR Dataset for Semantic Segmentation of Urban Roadways. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Seattle, WA, USA, 14–19 June 2020; pp. 797–806.
53. Trybała, P.; Szrek, J.; Remondino, F.; Kujawa, P.; Wodecki, J.; Blachowski, J.; Zimroz, R. MIN3D Dataset: MultI-seNsor 3D Mapping with an Unmanned Ground Vehicle. *PFG–J. Photogramm. Remote Sens. Geoinf. Sci.* **2023**, *91*, 425–442. [CrossRef]
54. Wang, L.; Huang, Y.; Shan, J.; Liu, H. MSNet: Multi-Scale Convolutional Network for Point Cloud Classification. *Remote Sens.* **2018**, *10*, 612. [CrossRef]
55. Wang, Y.; Wan, Y.; Zhang, Y.; Zhang, B.; Gao, Z. Imbalance Knowledge-Driven Multi-Modal Network for Land-Cover Semantic Segmentation Using Aerial Images and LiDAR Point Clouds. *ISPRS J. Photogramm. Remote Sens.* **2023**, *202*, 385–404. [CrossRef]
56. Xu, Y.; Du, B.; Zhang, L.; Cerra, D.; Pato, M.; Carmona, E.; Prasad, S.; Yokoya, N.; Hänsch, R.; Le Saux, B. Advanced Multi-Sensor Optical Remote Sensing for Urban Land Use and Land Cover Classification: Outcome of the 2018 IEEE GRSS Data Fusion Contest. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2019**, *12*, 1709–1724. [CrossRef]
57. Zolanvari, S.M.I.; Ruano, S.; Rana, A.; Cummins, A.; da Silva, R.E.; Rahbar, M.; Smolic, A. DublinCity: Annotated LiDAR Point Cloud and Its Applications. *arXiv* **2019**, arXiv:1909.03613.
58. Matrone, F.; Lingua, A.; Pierdicca, R.; Malinverni, E.S.; Paolanti, M.; Grilli, E.; Remondino, F.; Murtiyoso, A.; Landes, T. A Benchmark for Large-Scale Heritage Point Cloud Semantic Segmentation. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2020**, *XLIII-B2-2020*, 1419–1426. [CrossRef]
59. Vallet, B.; Brédif, M.; Serna, A.; Marcotegui, B.; Paparoditis, N. TerraMobilita/iQmulus Urban Point Cloud Analysis Benchmark. *Comput. Graph.* **2015**, *49*, 126–133. [CrossRef]
60. Liao, Y.; Xie, J.; Geiger, A. KITTI-360: A Novel Dataset and Benchmarks for Urban Scene Understanding in 2D and 3D. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 3292–3310. [CrossRef]
61. Caesar, H.; Bankiti, V.; Lang, A.H.; Vora, S.; Liong, V.E.; Xu, Q.; Krishnan, A.; Pan, Y.; Baldan, G.; Beijbom, O. nuScenes: A Multimodal Dataset for Autonomous Driving. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 11618–11628.
62. Niemeyer, J.; Rottensteiner, F.; Soergel, U. Contextual Classification of Lidar Data and Building Object Detection in Urban Areas. *ISPRS J. Photogramm. Remote Sens.* **2014**, *87*, 152–165. [CrossRef]

63. Oakland 3D Point Cloud Dataset—CVPR 2009 Subset. Available online: https://www.cs.cmu.edu/~vmr/datasets/oakland_3d/cvpr09/doc/ (accessed on 17 March 2023).
64. Serna, A.; Marcotegui, B.; Goulette, F.; Deschaud, J.E. Paris-Rue-Madame Database—A 3D Mobile Laser Scanner Dataset for Benchmarking Urban Detection, Segmentation and Classification Methods. In Proceedings of the 4th International Conference on Pattern Recognition, Applications and Methods ICPRAM 2014, Anger, France, 6–8 March 2014; Science and and Technology Publications—ESEO: Angers, France, 2014; pp. 819–824.
65. Open Dataset—Waymo. Available online: https://waymo.com/open/ (accessed on 18 February 2023).
66. Zhu, J.; Gehrung, J.; Huang, R.; Borgmann, B.; Sun, Z.; Hoegner, L.; Hebel, M.; Xu, Y.; Stilla, U. TUM-MLS-2016: An Annotated Mobile LiDAR Dataset of the TUM City Campus for Semantic Point Cloud Interpretation in Urban Areas. *Remote Sens.* **2020**, *12*, 1875. [CrossRef]
67. Behley, J.; Garbade, M.; Milioto, A.; Quenzel, J.; Behnke, S.; Stachniss, C.; Gall, J. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 9296–9306.
68. Charles, R.Q.; Su, H.; Kaichun, M.; Guibas, L.J. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 21–26 July 2017; pp. 77–85.
69. Qi, C.R.; Yi, L.; Su, H.; Guibas, L.J. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; p. 10.
70. Hu, Q.; Yang, B.; Xie, L.; Rosa, S.; Guo, Y.; Wang, Z.; Trigoni, N.; Markham, A. RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 11105–11114.
71. Qiu, S.; Anwar, S.; Barnes, N. Semantic Segmentation for Real Point Cloud Scenes via Bilateral Augmentation and Adaptive Fusion. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 1757–1767.
72. Li, Y.; Bu, R.; Sun, M.; Wu, W.; Di, X.; Chen, B. PointCNN: Convolution On X-Transformed Points. In Proceedings of the Advances in Neural Information Processing Systems 31 (NeurIPS 2018), Montréal, QC, Canada, 3–8 December 2018.
73. Thomas, H.; Qi, C.R.; Deschaud, J.-E.; Marcotegui, B.; Goulette, F.; Guibas, L.J. KPConv: Flexible and Deformable Convolution for Point Clouds. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019.
74. Liu, K.; Gao, Z.; Lin, F.; Chen, B.M. FG-Net: Fast Large-Scale LiDAR Point Clouds Understanding Network Leveraging Correlated Feature Mining and Geometric-Aware Modelling. *arXiv* **2021**, arXiv:2012.09439.
75. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
76. Huang, Q.; Wang, W.; Neumann, U. Recurrent Slice Networks for 3d Segmentation of Point Clouds. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 2626–2635.
77. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. In Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017.
78. Zhao, H.; Jiang, L.; Jia, J.; Torr, P.H.S.; Koltun, V. Point Transformer. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021.
79. Wu, X.; Lao, Y.; Jiang, L.; Liu, X.; Zhao, H. Point Transformer V2: Grouped Vector Attention and Partition-Based Pooling. In Proceedings of the Advances in Neural Information Processing Systems 35 (NeurIPS 2022), New Orleans, LA, USA, 28 November–9 December 2022.
80. Tatarchenko, M.; Park, J.; Koltun, V.; Zhou, Q.-Y. Tangent Convolutions for Dense Prediction in 3D. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 3887–3896.
81. Su, H.; Maji, S.; Kalogerakis, E.; Learned-Miller, E. Multi-View Convolutional Neural Networks for 3D Shape Recognition. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015.
82. Wu, B.; Wan, A.; Yue, X.; Keutzer, K. SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud. *arXiv* **2017**, arXiv:1710.07368.
83. Fischler, M.A.; Bolles, R.C. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM* **1981**, *24*, 381–395. [CrossRef]
84. Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-Level Accuracy with 50x Fewer Parameters and <0.5 MB Model Size. *arXiv* **2016**, arXiv:1602.07360.
85. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]

86. Wu, B.; Zhou, X.; Zhao, S.; Yue, X.; Keutzer, K. SqueezeSegV2: Improved Model Structure and Unsupervised Domain Adaptation for Road-Object Segmentation from a LiDAR Point Cloud. *arXiv* **2018**, arXiv:1809.08495.
87. Milioto, A.; Vizzo, I.; Behley, J.; Stachniss, C. RangeNet ++: Fast and Accurate LiDAR Semantic Segmentation. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 4–8 November 2019; pp. 4213–4220.
88. Su, H.; Jampani, V.; Sun, D.; Maji, S.; Kalogerakis, E.; Yang, M.-H.; Kautz, J. SPLATNet: Sparse Lattice Networks for Point Cloud Processing. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 2530–2539.
89. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767.
90. Xu, C.; Wu, B.; Wang, Z.; Zhan, W.; Vajda, P.; Keutzer, K.; Tomizuka, M. SqueezeSegV3: Spatially-Adaptive Convolution for Efficient Point-Cloud Segmentation. In *Computer Vision—ECCV 2020, Proceedings of the 16th European Conference, Glasgow, UK, 23–28 August 2020, Proceedings, Part XXVIII 16*; Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2020; Volume 12373, pp. 1–19, ISBN 978-3-030-58603-4.
91. Cortinhal, T.; Tzelepis, G.; Aksoy, E.E. SalsaNext: Fast, Uncertainty-Aware Semantic Segmentation of LiDAR Point Clouds for Autonomous Driving. In *Advances in Visual Computing, Proceedings of the 15th International Symposium, ISVC 2020, San Diego, CA, USA,* 5–7 October 2020, Proceedings, Part II 15; Springer International Publishing: Berlin/Heidelberg, Germany, 2020.
92. Aksoy, E.E.; Baci, S.; Cavdar, S. SalsaNet: Fast Road and Vehicle Segmentation in LiDAR Point Clouds for Autonomous Driving. In Proceedings of the 2020 IEEE Intelligent Vehicles Symposium (IV), Las Vegas, NV, USA, 19 October–13 November 2020; pp. 926–932.
93. Xiao, A.; Yang, X.; Lu, S.; Guan, D.; Huang, J. FPS-Net: A Convolutional Fusion Network for Large-Scale LiDAR Point Cloud Segmentation. *ISPRS J. Photogramm. Remote Sens.* **2021**, *176*, 237–249. [CrossRef]
94. Li, S.; Chen, X.; Liu, Y.; Dai, D.; Stachniss, C.; Gall, J. Multi-Scale Interaction for Real-Time LiDAR Data Segmentation on an Embedded Platform. *IEEE Robot. Autom. Lett.* **2021**, *7*, 738–745. [CrossRef]
95. Zou, Z.; Li, Y. Efficient Urban-Scale Point Clouds Segmentation with BEV Projection. *arXiv* **2021**, arXiv:2109.09074.
96. Hu, Q.; Yang, B.; Khalid, S.; Xiao, W.; Trigoni, N.; Markham, A. SensatUrban: Learning Semantics from Urban-Scale Photogrammetric Point Clouds. *Int. J. Comput. Vis.* **2022**, *130*, 316–343. [CrossRef]
97. Alnaggar, Y.A.; Afifi, M.; Amer, K.; Elhelw, M. Multi Projection Fusion for Real-Time Semantic Segmentation of 3D LiDAR Point Clouds. *arXiv* **2020**, arXiv:2011.01974.
98. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4510–4520.
99. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015, Proceedings of the 18th International Conference, Munich, Germany, 5–9 October 2015, Proceedings, Part III 18*; Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2015; Volume 9351, pp. 234–241, ISBN 978-3-319-24573-7.
100. Chen, L.-C.; Papandreou, G.; Schroff, F.; Adam, H. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv* **2017**, arXiv:1706.05587.
101. Triess, L.T.; Peter, D.; Rist, C.B.; Zöllner, J.M. Scan-Based Semantic Segmentation of Lidar Point Clouds: An Experimental Study. In Proceedings of the 2020 IEEE Intelligent Vehicles Symposium (IV), Las Vegas, NV, USA, 19 October–13 November 2020; pp. 1116–1121.
102. Riegler, G.; Ulusoy, A.O.; Geiger, A. OctNet: Learning Deep 3D Representations at High Resolutions. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 6620–6629.
103. Miller, A.; Jain, V.; Mundy, J.L. Real-Time Rendering and Dynamic Updating of 3-d Volumetric Data. In Proceedings of the Fourth Workshop on General Purpose Processing on Graphics Processing Units; ACM: Newport Beach, CA, USA, 5 March 2011; pp. 1–8.
104. Choy, C.; Gwak, J.; Savarese, S. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 3070–3079.
105. Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S.E.; Bronstein, M.M.; Solomon, J.M. Dynamic Graph CNN for Learning on Point Clouds. *ACM Trans. Graph. (TOG)* **2019**, *38*, 1–12. [CrossRef]
106. Li, G.; Müller, M.; Qian, G.; Delgadillo, I.C.; Abualshour, A.; Thabet, A.; Ghanem, B. DeepGCNs: Making GCNs Go as Deep as CNNs. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; p. 1. [CrossRef]
107. Zhang, Y.; Zhou, Z.; David, P.; Yue, X.; Xi, Z.; Gong, B.; Foroosh, H. PolarNet: An Improved Grid Representation for Online LiDAR Point Clouds Semantic Segmentation. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 9598–9607.

108. Lang, A.H.; Vora, S.; Caesar, H.; Zhou, L.; Yang, J.; Beijbom, O. PointPillars: Fast Encoders for Object Detection From Point Clouds. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 12697–12705.
109. Gerdzhev, M.; Razani, R.; Taghavi, E.; Liu, B. TORNADO-Net: mulTiview tOtal vaRiatioN semAntic Segmentation with Diamond inceptiOn Module. *arXiv* **2020**, arXiv:2008.10544.
110. Liu, Y.; Chen, R.; Li, X.; Kong, L.; Yang, Y.; Xia, Z.; Bai, Y.; Zhu, X.; Ma, Y.; Li, Y.; et al. UniSeg: A Unified Multi-Modal LiDAR Segmentation Network and the OpenPCSeg Codebase. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Paris, France, 2–6 October 2023.
111. Yan, X.; Gao, J.; Li, J.; Zhang, R.; Li, Z.; Huang, R.; Cui, S. Sparse Single Sweep LiDAR Point Cloud Segmentation via Learning Contextual Shape Priors from Scene Completion. *arXiv* **2020**, arXiv:2012.03762. [CrossRef]
112. Graham, B.; Engelcke, M.; Maaten, L. van der 3D Semantic Segmentation with Submanifold Sparse Convolutional Networks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 9224–9232.
113. Liu, Z.; Tang, H.; Lin, Y.; Han, S. Point-Voxel CNN for Efficient 3D Deep Learning. In Proceedings of the Advances in Neural Information Processing Systems 32 (NeurIPS 2019), Vancouver, BC, Canada, 8–14 December 2019.
114. Tang, H.; Liu, Z.; Zhao, S.; Lin, Y.; Lin, J.; Wang, H.; Han, S. Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution. In *Computer Vision—ECCV 2020, Proceedings of the 16th European Conference, Glasgow, UK, 23–28 August 2020, Proceedings, Part XXVIII*; Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2020; Volume 12373, pp. 685–702, ISBN 978-3-030-58603-4.
115. Rosu, R.A.; Schütt, P.; Quenzel, J.; Behnke, S. LatticeNet: Fast Point Cloud Segmentation Using Permutohedral Lattices. In Proceedings of the Robotics: Science and Systems 2020, Corvalis, Oregon, USA, 12–16 July 2020.
116. Cheng, R.; Razani, R.; Taghavi, E.; Li, E.; Liu, B. (AF)2-S3Net: Attentive Feature Fusion with Adaptive Feature Selection for Sparse Semantic Segmentation Network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021.
117. Hu, J.; Shen, L.; Sun, G. Squeeze-and-Excitation Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 7132–7141.
118. Zhu, X.; Zhou, H.; Wang, T.; Hong, F.; Ma, Y.; Li, W.; Li, H.; Lin, D. Cylindrical and Asymmetrical 3D Convolution Networks for LiDAR Segmentation. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 9934–9943.
119. Yan, X.; Gao, J.; Zheng, C.; Zheng, C.; Zhang, R.; Cui, S.; Li, Z. 2DPASS: 2D Priors Assisted Semantic Segmentation on LiDAR Point Clouds. In *Computer Vision—ECCV 2022, Proceedings of the 17th European Conference, Tel Aviv, Israel, 23–27 October 2022, Proceedings, Part XXVIII*; Springer Nature: Cham, Switzerland, 2022.
120. Landrieu, L.; Simonovsky, M. Large-Scale Point Cloud Semantic Segmentation with Superpoint Graphs. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4558–4567.
121. Landrieu, L.; Obozinski, G. Cut Pursuit: Fast Algorithms to Learn Piecewise Constant Functions on General Weighted Graphs. *SIAM J. Imaging Sci.* **2017**, *10*, 1724–1766. [CrossRef]
122. Kochanov, D.; Nejadasl, F.K.; Booij, O. KPRNet: Improving Projection-Based LiDAR Semantic Segmentation. *arXiv* **2020**, arXiv:2007.12668.
123. Alonso, I.; Riazuelo, L.; Montesano, L.; Murillo, A.C. 3D-MiniNet: Learning a 2D Representation from Point Clouds for Fast and Efficient 3D LIDAR Semantic Segmentation. *IEEE Robot. Autom. Lett.* **2020**, *5*, 5432–5439. [CrossRef]
124. Robert, D.; Vallet, B.; Landrieu, L. Learning Multi-View Aggregation In the Wild for Large-Scale 3D Semantic Segmentation. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 5565–5574.
125. Wang, S.; Zhu, J.; Zhang, R. Meta-RangeSeg: LiDAR Sequence Semantic Segmentation Using Multiple Feature Aggregation. *IEEE Robot. Autom. Lett.* **2022**, *7*, 9739–9746. [CrossRef]
126. Pan, Y.; Gao, B.; Mei, J.; Geng, S.; Li, C.; Zhao, H. SemanticPOSS: A Point Cloud Dataset with Large Quantity of Dynamic Instances. In Proceedings of the 2020 IEEE Intelligent Vehicles Symposium (IV), Las Vegas, NV, USA, 19 October–13 November 2020.
127. Wu, X.; Jiang, L.; Wang, P.-S.; Liu, Z.; Liu, X.; Qiao, Y.; Ouyang, W.; He, T.; Zhao, H. Point Transformer V3: Simpler, Faster, Stronger. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 16–22 June 2024.
128. Unal, O.; Dai, D.; Van Gool, L. Scribble-Supervised LiDAR Semantic Segmentation. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 2687–2697.
129. Liu, K.; Gao, Z.; Lin, F.; Chen, B.M. FG-Net: A Fast and Accurate Framework for Large-Scale LiDAR Point Cloud Understanding. *IEEE Trans. Cybern.* **2023**, *53*, 553–564. [CrossRef]

130. Jiang, L.; Shi, S.; Tian, Z.; Lai, X.; Liu, S.; Fu, C.-W.; Jia, J. Guided Point Contrastive Learning for Semi-Supervised Point Cloud Semantic Segmentation. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 11–17 October 2021; pp. 6403–6412.
131. Rozenberszki, D.; Litany, O.; Dai, A. Language-Grounded Indoor 3D Semantic Segmentation in the Wild. In *Computer Vision—ECCV 2022, Proceedings of the 17th European Conference, Tel Aviv, Israel, 23–27 October 2022, Proceedings, Part XXXIII*; Springer Nature: Cham, Switzerland, 2022.
132. Kuras, A.; Jenul, A.; Brell, M.; Burud, I. Comparison of 2D and 3D Semantic Segmentation in Urban Areas Using Fused Hyperspectral and Lidar Data. *J. Spectr. Imaging* **2022**, *11*, a11. [CrossRef]
133. Pan, J.; Cao, K.; Zhao, B.; Li, W.; Zhang, T. Semantic Segmentation of Large-Scale Point Clouds by Encoder-Decoder Shared MLPs with Weighted Focal Loss. In Proceedings of the 2022 IEEE 21st International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC), Toronto, ON, Canada, 8–10 December 2022; pp. 153–159.
134. Yunxiang, Z.; Ankang, J.; Limao, Z.; Xiaolong, X. Sampling-Attention Deep Learning Network with Transfer Learning for Large-Scale Urban Point Cloud Semantic Segmentation. Available online: https://reader.elsevier.com/reader/sd/pii/S0952197622005449?token=0833CEF881FED5ED093428D5EA678689AF0A9DE084A74D1624F22803FFEC668ED661E8AB56D0F5BA36853717D48A451D&originRegion=eu-west-1&originCreation=20221209104154 (accessed on 9 December 2022).
135. Liu, Y.; Li, J.; Yuan, X.; Zhao, C.; Siegwart, R.; Reid, I.; Cadena, C. Depth Based Semantic Scene Completion with Position Importance Aware Loss. *arXiv* **2020**, arXiv:2001.10709. [CrossRef]
136. Cheng, R.; Razani, R.; Ren, Y.; Bingbing, L. S3net: 3d Lidar Sparse Semantic Segmentation Network. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; pp. 14040–14046.
137. Berman, M.; Triki, A.R.; Blaschko, M.B. The Lovasz-Softmax Loss: A Tractable Surrogate for the Optimization of the Intersection-Over-Union Measure in Neural Networks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4413–4421.
138. Yan, X.; Zheng, C.; Li, Z.; Wang, S.; Cui, S. Pointasnl: Robust Point Clouds Processing Using Nonlocal Neural Networks with Adaptive Sampling. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 5589–5598.
139. Stearns, C.; Fu, A.; Liu, J.; Park, J.J.; Rempe, D.; Paschalidou, D.; Guibas, L.J. Curvecloudnet: Processing point clouds with 1d structure. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; IEEE: Piscataway, NJ, USA, 2024; pp. 27981–27991.
140. Truong, G.; Gilani, S.Z.; Islam, S.M.S.; Suter, D. Fast Point Cloud Registration Using Semantic Segmentation. In Proceedings of the 2019 Digital Image Computing: Techniques and Applications (DICTA), Perth, Australia, 2–4 December 2019; pp. 1–8.
141. Zhang, F.; Fang, J.; Wah, B.; Torr, P. Deep Fusionnet for Point Cloud Semantic Segmentation. In *Computer Vision–ECCV 2020, Proceedings of the 16th European Conference, Glasgow, UK, 23–28 August 2020, Proceedings, Part XXIV 16*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 644–663.
142. Huang, W.; Zhu, L.; Wang, W. Semantic Segmentation for Point Cloud Based on Distance Weighted and Adaptive Augmentation. In Proceedings of the 2022 34th Chinese Control and Decision Conference (CCDC), Hefei, China, 15–17 August 2022; pp. 6106–6111.
143. Xu, Y.; Tang, W.; Zeng, Z.; Wu, W.; Wan, J.; Guo, H.; Xie, Z. NeiEA-NET: Semantic Segmentation of Large-Scale Point Cloud Scene via Neighbor Enhancement and Aggregation. *Int. J. Appl. Earth Obs. Geoinf.* **2023**, *119*, 103285. [CrossRef]
144. Balado, J.; Fernández, A.; González, E.; Díaz-Vilariño, L. Semantic Point Cloud Segmentation Based on Hexagonal Klemperer Rosette and Machine Learning. In *Advances in Design Engineering III, Proceedings of the XXXI INGEGRAF International Conference 29–30 June, Málaga, Spain, 1 July 2022*; Cavas-Martínez, F., Marín Granados, M.D., Mirálbes Buil, R., de-Cózar-Macías, O.D., Eds.; Springer International Publishing: Cham, Switzerland, 2023; pp. 617–629.
145. Hassan, R.; Fraz, M.M.; Rajput, A.; Shahzad, M. Residual Learning with Annularly Convolutional Neural Networks for Classification and Segmentation of 3D Point Clouds. *Neurocomputing* **2023**, *526*, 96–108. [CrossRef]
146. Wang, C.; Samari, B.; Siddiqi, K. Local Spectral Graph Convolution for Point Set Feature Learning. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 52–66.
147. Fang, Y.; Xu, C.; Cui, Z.; Zong, Y.; Yang, J. Spatial Transformer Point Convolution. *arXiv* **2020**, arXiv:2009.01427.
148. Lu, T.; Wang, L.; Wu, G. CGA-Net: Category Guided Aggregation for Point Cloud Semantic Segmentation. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 11688–11697.
149. Wang, Z.; Rao, Y.; Yu, X.; Zhou, J.; Lu, J. SemAffiNet: Semantic-Affine Transformation for Point Cloud Segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022.
150. He, T.; Gong, D.; Tian, Z.; Shen, C. Learning and Memorizing Representative Prototypes for 3d Point Cloud Semantic and Instance Segmentation. In *Computer Vision–ECCV 2020, Proceedings of the 16th European Conference, Glasgow, UK, 23–28 August 2020, Proceedings, Part XVIII 16*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 564–580.

151. Zhao, Y.; Wang, J.; Li, X.; Hu, Y.; Zhang, C.; Wang, Y.; Chen, S. Number-Adaptive Prototype Learning for 3D Point Cloud Semantic Segmentation. In *Computer Vision, Proceedings of the ECCV 2022 Workshops, Tel Aviv, Israel, 23–27 October 2022, Proceedings, Part III*; Karlinsky, L., Michaeli, T., Nishino, K., Eds.; Springer Nature: Cham, Switzerland, 2023; pp. 695–703.
152. Liu, Z.; Hu, H.; Cao, Y.; Zhang, Z.; Tong, X. A Closer Look at Local Aggregation Operators in Point Cloud Analysis. In *Computer Vision—ECCV 2020, Proceedings of the 16th European Conference, Glasgow, UK, 23–28 August 2020, Proceedings, Part XXIII*; Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2020; Volume 12368, pp. 326–342, ISBN 978-3-030-58591-4.
153. Ma, Y.; Guo, Y.; Liu, H.; Lei, Y.; Wen, G. Global Context Reasoning for Semantic Segmentation of 3D Point Clouds. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Website, 2–5 March 2020; pp. 2931–2940.
154. Chen, Q.; Zhang, Z.; Chen, S.; Wen, S.; Ma, H.; Xu, Z. A Self-Attention Based Global Feature Enhancing Network for Semantic Segmentation of Large-Scale Urban Street-Level Point Clouds. *Int. J. Appl. Earth Obs. Geoinf.* **2022**, *113*, 102974. [CrossRef]
155. Lai, X.; Liu, J.; Jiang, L.; Wang, L.; Zhao, H.; Liu, S.; Qi, X.; Jia, J. Stratified Transformer for 3D Point Cloud Segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022.
156. Huang, Z.; Wu, X.; Zhao, H.; Zhu, L.; Wang, S.; Hadjidemetriou, G.; Brilakis, I. GeoSpark: Sparking up Point Cloud Segmentation with Geometry Clue. *arXiv* **2023**, arXiv:2303.08274.
157. Zheng, Y.; Xu, X.; Zhou, J.; Lu, J. PointRas: Uncertainty-Aware Multi-Resolution Learning for Point Cloud Segmentation. *IEEE Trans. Image Process.* **2022**, *31*, 6002–6016. [CrossRef]
158. Wan, J.; Zeng, Z.; Qiu, Q.; Xie, Z.; Xu, Y. PointNest: Learning Deep Multi-Scale Nested Feature Propagation for Semantic Segmentation of 3D Point Clouds. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2023**, *16*, 9051–9066. [CrossRef]
159. Xu, J.; Zhang, R.; Dou, J.; Zhu, Y.; Sun, J.; Pu, S. RPVNet: A Deep and Efficient Range-Point-Voxel Fusion Network for LiDAR Point Cloud Segmentation. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 11–17 October 2021; pp. 16004–16013.
160. Zhou, H.; Zhu, X.; Song, X.; Ma, Y.; Wang, Z.; Li, H.; Lin, D. Cylinder3D: An Effective 3D Framework for Driving-Scene LiDAR Semantic Segmentation. *arXiv* **2020**, arXiv:2008.01550.
161. Jiang, F.; Gao, H.; Qiu, S.; Zhang, H.; Wan, R.; Pu, J. Knowledge Distillation from 3D to Bird's-Eye-View for LiDAR Semantic Segmentation. In Proceedings of the 2023 IEEE International Conference on Multimedia and Expo (ICME), Brisbane, Australia, 10–14 July 2023.
162. Hua, B.-S.; Tran, M.-K.; Yeung, S.-K. Pointwise Convolutional Neural Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 984–993.
163. Wang, Z.; Lu, F. VoxSegNet: Volumetric CNNs for Semantic Part Segmentation of 3D Shapes. *arXiv* **2018**, arXiv:1809.00226. [CrossRef]
164. Zhou, W.; Cao, X.; Zhang, X.; Hao, X.; Wang, D.; He, Y. Multi Point-Voxel Convolution (MPVConv) for Deep Learning on Point Clouds. *arXiv* **2021**, arXiv:2107.13152. [CrossRef]
165. Wang, P.-S. OctFormer: Octree-Based Transformers for 3D Point Clouds. *ACM Trans. Graph.* **2023**, *42*, 1–11. [CrossRef]
166. Genova, K.; Yin, X.; Kundu, A.; Pantofaru, C.; Cole, F.; Sud, A.; Brewington, B.; Shucker, B.; Funkhouser, T. Learning 3D Semantic Segmentation with Only 2D Image Supervision. In Proceedings of the 2021 International Conference on 3D Vision (3DV), Online, 1–3 December 2021; pp. 361–372.
167. Wei, J.; Lin, G.; Yap, K.-H.; Hung, T.-Y.; Xie, L. Multi-Path Region Mining For Weakly Supervised 3D Semantic Segmentation on Point Clouds. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020.
168. Xu, X.; Lee, G.H. Weakly Supervised Semantic Point Cloud Segmentation: Towards 10x Fewer Labels. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 13706–13715.
169. Zhao, J.; Huang, W.; Wu, H.; Wen, C.; Yang, B.; Guo, Y.; Wang, C. SemanticFlow: Semantic Segmentation of Sequential LiDAR Point Clouds from Sparse Frame Annotations. *IEEE Trans. Geosci. Remote Sens.* **2023**, *61*, 5701611. [CrossRef]
170. Peters, T.; Brenner, C.; Schindler, K. Semantic Segmentation of Mobile Mapping Point Clouds via Multi-View Label Transfer. *ISPRS J. Photogramm. Remote Sens.* **2023**, *202*, 30–39. [CrossRef]
171. Zhao, N.; Chua, T.-S.; Lee, G.H. Few-Shot 3D Point Cloud Semantic Segmentation. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 8869–8878.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.