NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

DIOGO RESTOLHO MATEUS MARÇALO LAVADO

BSc in Computer Science

DETECTION OF POWER LINE SUPPORTING TOWERS VIA INTERPRETABLE SEMANTIC SEGMENTATION OF 3D POINT CLOUDS

A DEEP DIVE INTO A NOVEL POINT CLOUD LEARNING TECHNIQUE BUILT UPON A MATHEMATICAL FRAMEWORK BASED ON GROUP EQUIVARIANT NON-EXPANSIVE OPERATORS

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
February, 2022

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF
COMPUTER SCIENCE

DETECTION OF POWER LINE SUPPORTING TOWERS VIA INTERPRETABLE SEMANTIC SEGMENTATION OF 3D POINT CLOUDS

A DEEP DIVE INTO A NOVEL POINT CLOUD LEARNING TECHNIQUE BUILT UPON A MATHEMATICAL FRAMEWORK BASED ON GROUP EQUIVARIANT NON-EXPANSIVE OPERATORS

DIOGO RESTOLHO MATEUS MARÇALO LAVADO

BSc in Computer Science

Adviser: Cláudia Soares
Assistant Professor, NOVA University Lisbon

Co-adviser: Alessandra Micheletti
Associate Professor, Università degli Studi di Milano

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
February, 2022

Detection of Power Line Supporting Towers via Interpretable Semantic Segmentation of 3D Point Clouds

Acknowledgements

I would like to express my gratitude to Professor Cláudia Soares for her guidance in this dissertation, for her endless patience, for believing in me and pushing me to aim for the stars. I would also like to thank Professor Alessandra Micheletti, for her assistance and essential input, and to EDP New for this amazing opportunity, for welcoming me into your team and for always being available when I needed.

Additionally, I have to give my biggest thanks to my friends Carolina Lopes, Tiago Soares and Xavier Pacheco, for bearing with me when I lack caffeine in my system, for listening to my rants, for giving me a safe space and a second family.

Lastly, I would like to thank my mom, Marina Mateus, for insisting since I was a little boy that I have no idea of the immense power I hold inside me. Thank you for being a relentless fighter, for all the sacrifices and struggles, and above all else, for being my mom.

"Imagination is more important than knowledge. Knowledge is limited. Imagination encircles the world. " (Albert Einstein)

"Try again, fail again. Fail better. " (Samuel Beckett)

"Sometimes it is the people no one can imagine anything of who do the things no one can imagine. " (Alan Turing)

Abstract

The inspection and maintenance of energy transmission networks are demanding and crucial tasks for any transmission system operator. They rely on a combination of on-the-ground staff and costly low-flying helicopters to visually inspect the power grid structure. Recently, LiDAR-based inspections have shown the potential to accelerate and increase inspection precision. These high-resolution sensors allow one to scan an environment and store it in a 3D point cloud format for further processing and analysis by maintenance specialists to prevent fires and damage to the electrical system. However, this task is especially demanding to handle on time when we consider the extensive area that the transmission network covers. Nonetheless, the transition to point cloud data allows us to take advantage of Deep Learning to automate these inspections, by detecting collisions between the grid and the revolving scene.

Deep Learning is a recent and powerful tool that has been successfully applied to a myriad of real-life problems, such as image recognition and speech generation. With the introduction of affordable LiDAR sensors, the application of Deep Learning on 3D data emerged, with numerous methods being proposed every day to address difficult problems, from 3D object detection to 3D point cloud segmentation. Alas, state-of-the-art methods are remarkably complex, composed of millions of trainable parameters, and take several weeks, if not months, to train on specific hardware, which makes it difficult for traditional companies, like utilities, to employ them.

Therefore, we explore a novel mathematical framework that allows us to define tailored operators that incorporate prior knowledge regarding our problem. These operators are then integrated into a learning agent, called SCENE-Net, that detects power line supporting towers in 3D point clouds. SCENE-Net allows for the interpretability of its results, which is not possible in conventional models, it shows an efficient training and inference time of 85 mn and 20 ms on a regular laptop. Our model is composed of 11 trainable geometrical parameters, like the height of a cylinder, and has a Precision gain of 24% against a comparable CNN with 2190 parameters.

Keywords: Deep Learning, GENEO, LiDAR point clouds, 3D semantic segmentation

Resumo

A inspeção e manutenção de redes de transmissão de energia são tarefas cruciais para operadores de rede. Recentemente, foram adotadas inspeções utilizando sensores LiDAR de forma a acelerar este processo e aumentar a sua precisão. Estes sensores são objetos de alta precisão que conseguem inspecionar ambientes e guarda-los no formato de nuvens de pontos 3D, para serem posteriormente analisadas por specialistas que procuram prevenir fogos florestais e danos à estruta eléctrica. No entanto, esta tarefa torna-se bastante difícil de concluir em tempo útil pois a rede de transmissão é bastasnte vasta. Por isso, podemos tirar partido da transição para dados LiDAR e utilizar aprendizagem profunda para automatizar as inspeções à rede.

Aprendizagem profunda é um campo recente e em grande desenvolvimento, sendo aplicado a vários problemas do nosso quotidiano e facilmente atinge um desempenho superior ao do ser humano, como em reconhecimento de imagens, geração de voz, entre outros. Com o desenvolvimento de sensores LiDAR acessíveis, o uso de aprendizagem profunda em dados 3D rapidamente se desenvolveu, apresentando várias metodologias novas todos os dias que respondem a problemas complexos, como deteção de objetos 3D. No entanto, modelos do estado da arte são incrivelmente complexos e compostos por milhões de parâmetros e demoram várias semanas, senão meses, a treinar em GPU potentes, o que dificulta a sua utilização em empresas tradicionais, como a EDP.

Portanto, nós exploramos uma nova teoria matemática que nos permite definir operadores específicos que incorporaram conhecimento sobre o nosso problema. Estes operadores são integrados num modelo de aprendizagem prounda, designado SCENE-Net, que deteta torres de suporte de linhas de transmissão em nuvens de pontos. SCENE-Net permite a interpretação dos seus resultados, aspeto que não é possível com modelos convencionais, demonstra um treino eficiente de 85 minutos e tempo de inferência de 20 milissegundos num computador tradicional. O nosso modelo contém apenas 11 parâmetros geométricos, como a altura de um cilindro, e demonstra um ganho de Precisão de 24% quando comparado com uma CNN com 2190 parâmetros.

Palavras-chave: Aprendizagem profunda, GENEO, nuvens de pontos em LiDAR, segmentação semântica 3D

Contents

List of Figures

List of Tables

Glossary

convolution Operation between two functions (f and ψ) that outputs a third one ($f * \psi$) expressing how the shape of f reacts to ψ. Formally $(f * \psi)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\,d\tau$ 3, 10, 38, 39

equivariance Form of symmetry between two function spaces with respect to an action group, meaning that a function space is transformed into another function space while preserving a group of symmetries. 31, 32, 33, 34, 35, 36, 38, 39

GENEO Group Equivariant Non-Expansive Operator: Function that transforms a space of measurements Φ to Ψ, while preserving an associated homeomorphism 4, 24, 31, 32, 34, 35, 36, 38, 39, 40, 53, 54, 62, 65

homeomorphism A continuous function between topological spaces that has a continuous inverse function, which means that it preserves all topological properties of a given space. 32, 33, 34

IoU Intersection over union quantifies the degree of overlap between the Ground Truth (GT) and prediction region (P). Formally, $IoU = \frac{|P \cap GT|}{|P \cup GT|}$ 14, 21, 56, 57

TDA Topological Data Analysis: the evaluation and study of topological spaces, which in turn can be generally defined as a space with structure. 32

Acronyms

BEV Bird's-eye view 2, 19, 49, 50

CNN Convolutional Neural Network 4, 5, 8, 9, 12, 19, 22, 24, 25, 29, 31, 35, 38, 39, 52, 56, 58, 61, 65

DL Deep Learning 1, 2, 6, 23, 24, 25, 59, 61

LiDAR Light Detection And Ranging 2, 7, 17, 20, 25, 26, 27, 50

ML Machine Learning 1, 2, 4, 9, 24, 28, 30, 31, 36, 56, 61, 65, 66

MLP MultiLayer Perceptron 11, 12, 17, 20

MSE Mean Squared Error 47, 60

RMSProp Root Mean Squared Propagation 54, 61

TSOs Transmission System Operators 1, 2

UAVs Unmanned Aerial Vehicles 2, 25, 26

1 Introduction

1.1 Problem Description and Motivation

Nowadays, the world is embracing data-driven approaches to most real-life problems, information is collected in large quantities on a variety of subjects, from shopping habits to head CT scans, in order to unravel hidden patterns using statistical methods and gain valuable insight into a group of interest. For example, by examining the shopping habits of a large demographic, they can be correlated to other characteristics, such as age and gender, to improve the recommendation of items for people in similar social groups. Alternatively, numerous labeled head CT scans can power a statistical model to find a set of characteristics that describe the existence of tumors. The majority of traditional Machine Learning (ML) methods are black boxes. That is, they cannot provide human-understandable reasoning for their predictions. However, the widespread application of Machine Learning to real-world problems, such as the ones above described, entails responsible and comprehensible decision-making so that practitioners in other disciplines can clearly understand these models and safely apply them to their data. For example, doctors require a trustworthy explanation as to why an ML model detects a tumor in a CT scan, otherwise these statistical methods cannot be safely employed in critical situations.

In general, state-of-the-art Deep Learning (DL) models are problem-specific and have particular hardware requirements to deploy them. The computing power and hardware available in traditional companies, like utilities, are not designed for large Deep Learning pipelines. That is, a resource-efficient application of DL models in traditional companies is limited because of low expertise in Machine Learning and computational availability, despite the deep in-house knowledge in their specific fields.

Transmission System Operators (TSOs) guarantee the transportation of electrical energy from generating sites, such as power plants, to electrical substations, which are in charge of transforming between voltage levels to then supply energy to the end-customer [1]. TSOs facilitate and maintain an electrical grid in order to interconnect both ends of the transmission system. The main purpose of the grid is to efficiently transmit large amounts of energy through long distances by the means of overhead power lines,

which consist of electrical cables suspended by towers or poles. TSOs have the mission to inspect and maintain the electrical grid for security purposes, namely, they need to assess the risk of contact between the power grid and the environment in order to prevent defects, power outages, and even forest fires. Electrical grids spread over countries and even continents, thus making careful inspection an important and challenging problem.

In Portugal, such tasks are carried out by EDP partnered with Labelec. They heavily rely on a combination of on-the-ground staff and manned low-flying helicopters that examine the power grid with hand-ported devices or the naked eye. These methods are expensive, inefficient, and demanding for the staff. Recently, they adopted inspections based on Light Detection And Ranging (LiDAR) to expedite this process and increase its precision [2]. Specifically, LiDAR sensors are equipped on Unmanned Aerial Vehicles (UAVs) to scan the power grid from a Bird's-eye view (BEV) perspective and capture a 3D point cloud representation of the environment. Even though this approach is faster and cheaper since it offers a higher degree of automation and less frequent on-site surveys, the captured point clouds are quite extensive and mostly composed of arboreal areas, with the electrical grid making up a small percentage of the overall data. As a result, the point cloud data must be carefully processed and analyzed by maintenance personnel. First, the 3D environments are sectioned into strips of land that encompass the electrical grid. Then, the resulting 3D scenes are manually labeled to distinguish relevant elements, such as the ground, trees, power lines, and towers, from those that are not, i.e., noise, water, dust particles, among others. This is an extremely repetitive task with low added-value from the specialists that must perform it.

Conveniently, the transition to high-precision point cloud data using LiDAR allows us to take advantage of DL methods to assist power grid inspections. In this work, we propose a novel Deep Learning framework that transparently detects power line supporting towers. These metal structures sustain the energy distribution system and serve as a point of reference for the location of power lines. This way, the electrical grid can be automatically extracted from large-scale 3D point clouds, which considerably speeds us the inspection time for TSOs. In addition, our proposal allows us to measure the risk of contact between towers and the revolving environment, setting up a strong basis for trustworthy ML-based power grid inspections. To bootstrap this work, EDP New and Labelec provided us with a labeled dataset of 40 000 Km of rural and forest terrain, and the **T**ransmission **S**ystem, named **TS40K**.

1.2 Deep Learning on 3D Point Clouds

With the rapid growth in precision and affordability of 3D mapping technologies, 3D data can be easily produced with great detail using 3D sensors, such as LiDAR [3] and RGB-D cameras [4, 5]. By representing data in three dimensions, environments scanned by these sensors are more accurate than 2D images in terms of geometry, scale, and shape [6, 7]. The addition of the third dimension allows us to fully represent objects as they exist in

real life, that is, in terms of width, height, and depth, whereas 2D representations render objects to flat figures without depth perception. The loss of information between the true form of an object and its 2D projection makes it challenging to measure relations with the environment, such as the distance to other objects and their true sizes. In 3D space, these relations occur naturally since objects are represented with their original dimensions. Point clouds are the preferred medium to represent 3D environments for most applications, such as autonomous driving and robotics. They are sets of data points in space, each defined by at least a set of Cartesian coordinates (x, y, z). This way, they offer a compact and fine-detailed depiction of 3D data. The affordable access to high-precision 3D sensors fuelled the rise of Deep Learning applications to 3D point clouds. By integrating these sensors into day-to-day devices, such as vehicles, high-quality datasets were quickly developed and made available to the public, such as ScanObjectNN [8], Semantic3D [9] and the KITTI Vision Benchmark Suite [10, 11]. Consequently, the research on Deep Learning for 3D point clouds expanded into a diverse field, having numerous proposals addressing different problems involving 3D data.

A plausible approach to detecting supporting towers is the use of 3D semantic segmentation or 3D object detection methodologies. State-of-the-art literature for both alternatives can be divided into three groups: multi-view-based, voxel-based, and point-based methods.

Multi-view-based methods project point clouds onto 2D planes [12, 13] in order to employ classic Deep Learning strategies for 2D data, such as 2D convolutions. These approaches usually introduce information loss and decrease in accuracy in tasks such as segmentation, because of botched reconstructions of 3D data from 2D maps.

Voxel-based methods [14–16] endow point clouds with structure in order to use global feature descriptors, such as 3D convolutions. That is, certain operations widely used in Deep Learning, like the convolution operator, requires structure from the input data to be applied. Since the lack of structure of raw point clouds impedes these operators, a grid of voxels is used to discretize point clouds. A voxel grid can be thought of as a 3D image frame and a voxel is analogous to a 3D pixel that contains and represents a subset of points from the input point cloud. However, these methods are restricted in resolution due to the cubic growth of computational complexity and memory footprint. In other words, higher resolution voxel grids (i.e., the number of voxels used to discretize the point cloud) provide a more fine-grained representation of 3D scenes, which ultimately leads to more accurate results in real-life scenarios. But it also causes a cubic increase in memory consumption and computational overhead. Conversely, low-resolution voxel grids entail a loss of information when discretizing scene elements, their form is disrupted and heavily pixelized.

Point-based methods take raw point clouds directly as input [17–24]. These strategies have to resort to costly neighbor searching techniques to extract local information, which sometimes leads to global information loss. Moreover, these methods have progressively increased their complexity to a remarkable degree in order to, simultaneously, address

the challenges incurred from the use of raw point clouds (e.g., permutation invariance and heterogeneous density) and yield good performance in real-life scenarios, such as autonomous driving.

In general, state-of-the-art methods for 3D semantic segmentation and object detection are black-box models with millions of trainable parameters, non-trivial amounts of training times, and specific hardware requirements. Additionally, most proposals are tailored to boost performance in urban settings, such as Semantic3D [9] and the KITTI dataset [11], where data are sparse, objects are often occluded and may demonstrate anisotropy w.r.t. density.

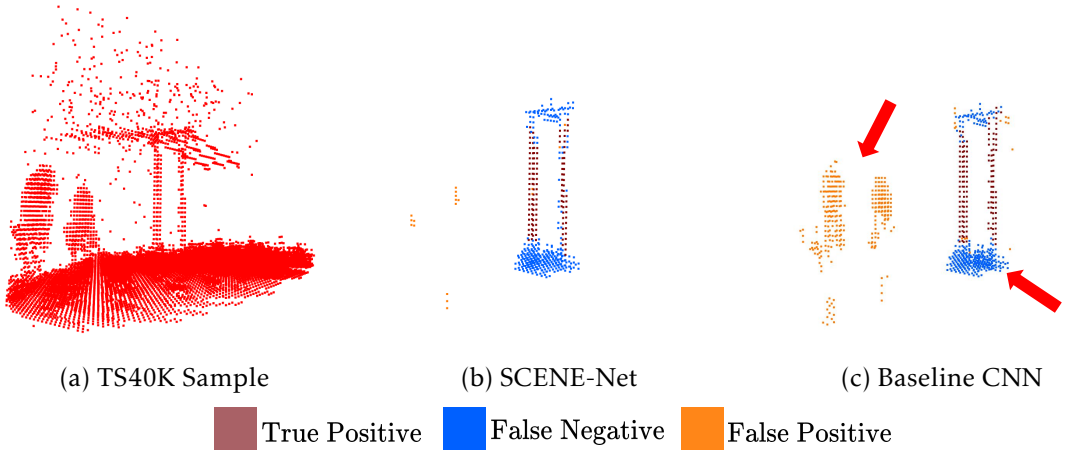
1.3 Deep Learning on Power Grid Inspections

State-of-the-art methods for power grid inspection automation usually focus on high-voltage power line segmentation. That is, most proposals center their approach in detecting power lines of a transmission grid and extracting them from the 3D environment [25–27]. Additionally, these methods try to circumvent the difficulties of working with 3D data by using 2D projections, which are known to introduce information loss and decrease accuracy. Instead of detecting power lines, our proposal focuses on the towers that support them. These structures are also subject to careful inspections and provide a reliable reference location for the power lines they support. It is arguably more difficult to derive an accurate segmentation of supporting towers from the location of power lines. In addition, we take into account raw 3D scenes without the need for 2D projections, this allows us to accurately detect supporting towers without information loss.

1.4 SCENE-Net: Signature geometriC Equivariant Non-Expansive operator Network

Our approach to supporting tower detection is built upon a novel Machine Learning paradigm based on group equivariant non-expansive operators (GENEO) [28, 29]. These operators provide us a measure of the world, just as Convolutional Neural Network (CNN) kernels learn essential features to, for instance, recognize objects. With GENEO, ML agents are formally described as a set of operators acting on the input data, we can think of them as observers that analyze data. They transform it into higher-level representations while respecting some set of properties. An appropriate observer transforms data while preserving a pre-defined set of meaningful features. For instance, the geometry of supporting towers can be fully described as a set of properties. This technique effectively let us define prior knowledge and embed it into a machine learning model as, for example, convolutional kernels.

In this work, we introduce **SCENE-Net**, an intrinsically interpretable 3D point cloud semantic segmentation framework identifying signature shapes with GENEOs that allow



(a) TS40K Sample (b) SCENE-Net (c) Baseline CNN

Figure 1.1: Learning signature shapes for power line supporting tower detection. For the TS40K sample shown in (a), SCENE-Net accurately detects the body of the power grid tower (b), while a comparable CNN has a large false positive area in the vegetation (c). Our model is interpretable with 11 trainable geometric parameters whereas the CNN has a total of 2190 parameters. The ground around the towers and the lines above are mislabeled as towers.

for fast training even with small data, and robustness to labeling noise and strong imbalance. GENEOs act as meaningful observers of properties of the semantic classes we aim to identify in the data. Here, we identify geometric properties of electric grid supporting towers, such as verticality. Our model is a sweet spot between fully data-driven (CNNs) and fully model-driven (e.g., template matching) solutions. To summarize, our contributions are:

- We present TS40K, a new 3D point cloud dataset covering 40 000 Km of non-urban terrain, with more than 9000 million 3D points (Section 5.1);
- SCENE-Net is the first intrinsically interpretable model for 3D semantic segmentation on large-scale landscapes, including non-urban environments (Section 4.3);
- SCENE-Net is intrinsically interpretable and robust under noisy labels (Section 5.2.2);
- The architecture of SCENE-Net has less trainable parameters than traditional methods, and, thus, fast training times while achieving effective results (Section 5.2.6);
- As GENEOs are continuous observer functions, SCENE-Net is independent of input and kernel voxelized shapes. I.e., it can change the kernel shape after training. The kernel size used in SCENE-Net is fine-tuned after training to boost its performance (Section 5.2.7).

1.5 Document Structure

First, we present the related work to our dissertation in Chapter 2, including an in-depth explanation of 3D data representation techniques and existing DL models, we also discuss existing power grid inspection strategies, the concept of explainability and interpretability, and how our proposal goes a step beyond these notions. In Chapter 3, we introduce group equivariant non-expansive operators along with the published work around it. In Chapter 4, we describe in detail our proposed power line supporting tower detection framework. Then, we detail the TS40K dataset used in our experiments, from its properties to its pre-processing stage in Section 5.1. Chapter 5 discusses all the performed experiments during this dissertation along with key decision-making that culminated in our final proposal. Lastly, we present the conclusions of our work in Chapter 6, with an objective discussion and interesting ideas for future work in this field.

2

Related Work

In this chapter, we examine literature pertinent to our dissertation. First, we examine research work regarding Deep Learning models on 3D point clouds, from the properties of 3D data to state-of-the-art methods in 3D semantic segmentation and 3D object detection. We present a detailed analysis of these methodologies concerning the detection of supporting towers. Then, we describe the current techniques for power grid inspections and compare them to our proposal. Lastly, we introduce the concepts of explainability and interpretability, along with state-of-the-art models that follow these notions, and compare them to our proposed SCENE-Net.

2.1 Deep Learning on 3D Point Clouds

In this section, we examine the state-of-the-art literature regarding 3D point cloud learning. First, we introduce the characteristics and challenges of working with 3D data. Next, we examine several state-of-the-art methods that respond to 3D segmentation and 3D object detection. Seeing as traditional companies, such as EDP, do not have the resources to employ these methods, we do not benchmark our proposal against them. Then, we present useful datasets used as benchmarks by researchers, pertinent evaluation metrics, and show the performance of the introduced models on these benchmarks. Lastly, we discuss the benefits and drawbacks of 3D semantic segmentation and 3D object detection with respect to the detection of supporting towers.

2.1.1 Working with 3D Point Clouds

Points clouds are data structures used to represent 3D spaces, they are sets of 3D points represented by (x, y, z) coordinates and may include additional information in their definition, such as RGB values, normal vectors, among others. Thus, 3D point clouds are generally denoted as $\mathcal{P} \in \mathbb{R}^{N \times (3+d)}$, where N is the number of points and $3 + d$ is the cardinality of spatial coordinates plus any point-wise features, such as color.

High-precision sensors capable of scanning environments and storing them in a point cloud format, such as LiDAR and mobile phones with depth cameras [4, 5], became more

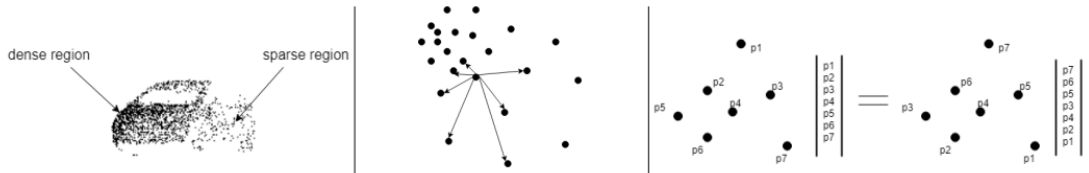
affordable, available, and easily integrated into day-to-day devices, such as vehicles [3]. This allowed for the quick acquisition of 3D data from different environments, fueling the development of several publicly available datasets that today are used as important benchmarks in the area, namely ScanObjectNN [8], SensatUrban [30], the KITTI Vision Benchmark Suite [10, 11], among others. These are composed of numerous point clouds with millions of points that represent 3D scenes where urban objects of interest are labeled, such as cars and pedestrians. With well-founded datasets, industries such as autonomous driving, robotics, and augmented reality, quickly began to investigate the application of Deep Learning to 3D point clouds.

Although point clouds provide a compact and fine-detailed depiction of 3D data, processing raw point clouds proved to be a challenging task for classic Machine Learning strategies due to the properties of point sets in $\mathbb{R}^{3+d}$:

Heterogeneous Density: Point clouds may not be evenly sampled across different regions of a scene, meaning that an object can have both dense and sparse point sections. Figure 2.1 (a) shows this property, the car is composed of both dense and sparse regions. Such an effect is especially common in datasets captured from a vehicle point-of-view. Deep Learning models based on neighbor searching techniques, such as k-Nearest Neighbors, fail to accurately capture local features due to this trait.

Unstructured Nature: Contrary to images, where a 2D grid structure relates each pixel to its neighbors, point clouds are not organized in a regular grid. Each point is scanned independently of the other and there is no fixed distance between points as there is for pixels. Adjacent to this, points are not isolated structures, they gain meaning in a 3D scene when they are part of a subset of neighboring points. So, it is important to capture nearby structures and their interactions. This property is illustrated in Figure 2.1 (b), most mathematical operators used in Machine Learning require structured data, this is the case for the convolution operator for example. Without structure, state-of-the-art methods are forced to achieve a global feature descriptor by the means of local aggregation techniques, which may lead to information loss due to heterogeneous density and object occlusion.

Permutation Invariance: Point clouds are independent of the order in which 3D points are stored, as it does not change the represented scene. In addition, point clouds are invariant under certain planar transformations. For instance, rotating and translating point clouds does not change the elements in 3D scenes. Figure 2.1 (c) explains this property, the two presented orders are considered equivalent since they represent the same points, and therefore, the same 3D setting. Deep Learning methods often correlate the order of the input to the intended output. For example, the order in which image pixels are presented to a CNN is essential for it to detect, for instance, the written digit in the provided image due to the use of convolutions. However, the invariance to permutations of 3D point clouds clashes with this behavior. Thus, state-of-the-art models often need to employ additional strategies to mitigate this problem.



(a) Irregular. Sparse and dense regions

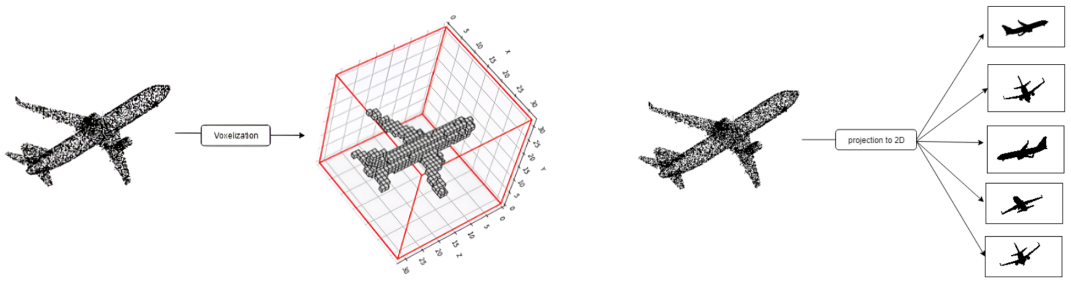
(b) Unstructured. No grid, each point is independent and distance between neighboring points is not fixed

(c) Unordered. As a set, point cloud is invariant to permutation

Figure 2.1: Properties of 3D Point Clouds [31].

2.1.2 Structured-Based Learning

Early Deep Learning proposals try to overcome the challenges associated with 3D point clouds by converting them into structured representations. These can be broadly divided into two categories, voxel-based and multi-view-based approaches.



(a) Point cloud voxelized with a 32^3 voxel grid

(b) Multi-view projection of a point cloud to 2D images

Figure 2.2: Structured-based learning techniques. [31].

2.1.2.1 Multi-View Based Methods

Convolution is a well-studied operation in Machine Learning and signal processing. It is the nucleus of numerous ML models, such as CNNs for image classification. With the recent wide-spread use of 3D data in numerous fields, one of the first introduced methodologies was projecting data into multiple two-dimensional views, as shown in Figure 2.2b. This way, traditional 2D convolutions can be used to extract view-wise features, which are then merged into a representation of the original 3D shape.

When compared to volumetric methods, multi-view approaches [12, 13, 32–34] achieve dominating performance on shape classification and retrieval tasks. However, it is difficult to extend them to scene understanding tasks, such as 3D semantic segmentation and 3D object detection [35]. By disregarding the depth dimension, multi-view-based methods cannot accurately reconstruct the original 3D space, which ultimately leads to a poor performance in such tasks.

2.1.2.2 Voxel-Based Methods

The convolution of 2D images is possible because they have structure, they are organized in $N \times M$ matrices, where N is the number of vertical pixels, and M is the count of horizontal pixels, so the values inside each coordinate (x_i, y_j) correspond to the intensity of a pixel. Voxel-based strategies [14–16, 36, 37] encompass raw 3D point clouds with a 3D voxel grid with size $N \times M \times K$, where K is the cardinality of the depth dimension. This way, they endow structure to 3D data in order to obtain global feature descriptors, namely by using 3D convolutions with learned kernels, while preserving the three dimensions of the original data.

Figure 2.2a shows the voxelization of a 3D point cloud containing an airplane in a 32^3 voxel grid. This structure is a three-dimensional tensor where each voxel at coordinate (x_i, y_j, z_k) contains a subset of 3D points from the original 3D data. A representation function is subsequently applied to show the presence or density of 3D points in each voxel. This is analogous to assigning real values to pixels in a grayscale image, measuring the whiteness of each pixel. In voxel grids, voxels are usually given a value of one if they contain any 3D points, and zero otherwise. As depicted in the airplane voxelization, this technique reduces the sharpness of the objects discretized, which ultimately leads to information loss for low-resolution voxel grids. Thus, higher voxel grid resolutions are preferred, similarly to how higher resolution images are preferred to obtain better accuracy in, for instance, object classification. However, volumetric representations suffer from high memory consumption due to voxel sparsity, which leads to great computation overheads when performing convolutions of voxel grids [31]. Moreover, 3D convolutions are generally more computationally expensive than 2D convolutions. In conclusion, voxel-based methods have a clear trade-off between resolution and performance. Low resolution shows efficient results but leads to information loss and a decrease in accuracy. In turn, high resolution sharply discretize 3D objects but entail a large memory footprint and slow training times [31, 35].

To overcome the challenges of voxelization, references [38–40] propose octree-based representations in order to reduce the impact of unoccupied voxels and increase the feasible resolution of the voxel grid (reaching 256^3 voxels). Essentially, to construct an octree representation of a point cloud, the 3D space is first partitioned into 8 voxels of the same size, and then each non-empty voxel is recursively partitioned in the same way until the maximum depth level is reached. This way, the voxel grid represents occupied voxels in much more detail and minimizes the impact of processing empty voxels. Withal, state-of-the-art point-based methods show better performance with more efficient use of computational resources.

2.1.3 Learning from Raw Point Clouds: Point-Based Methods

Despite the challenges of working with raw point clouds, Deep Learning on 3D data began to receive a lot of attention after the pioneering work of PointNet [17] and PointNet++ [18],

which directly take point clouds as input and report state-of-the-art performance for 3D classification and 3D segmentation tasks. Consequently, point-based methods became increasingly popular as they do not introduce any information loss or additional overhead in preprocessing point clouds, unlike volumetric and multi-view-based approaches. Nowadays, these methods dominate the field of point cloud learning in terms of research and achieved performance [35].

We can further divide these approaches according to the techniques they employ, for our work we are mostly interested in pointwise MultiLayer Perceptron (MLP) methods and convolution-based methods. In this subsection, we will review state-of-the-art techniques that laid the groundwork for the vanguard research being developed today.

2.1.3.1 PointNet

PointNet [17] is a deep learning framework that directly consumes unordered point sets as input and performs 3D classification and 3D segmentation. For the former, the network will output k scores for all k candidate classes, whereas, for the latter, it will output $n \times m$ scores, for each of the n points and each of the m semantic sub-categories. Essentially, this means that each point will have a score representing how much it belongs to each of the m segments.

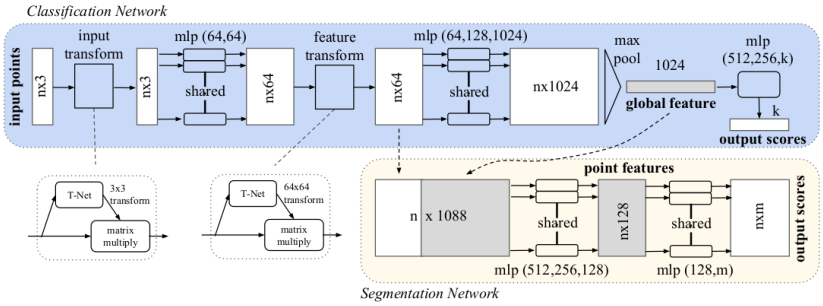


Figure 2.3: PointNet architecture [17].

Qi et al. [17] define three key modules that grant PointNet with its performance: A symmetry function that aggregates information from all n points, a local and global information combination structure, and two joint alignment networks that align the input points with the point features.

Symmetry Function for Unordered Input: As previously discussed, point sets demonstrate permutation invariance, meaning that they are independent of the order in which the points are stored. However, neural networks naturally relate the order of the input with its desired output, which poses a problem. To solve this, PointNet employs a series of shared multilayer perceptrons (MLPs) to extract s features from the 3D cloud. It then applies a symmetry function, specifically max pooling, to aggregate the results from all n points. The output of this function is a new vector of size $s \leq n$ that serves as a global descriptor of the input point cloud. This answers the permutation invariance issue because

the global descriptor will be the same for any order in which the n points are organized.

Local and Global Information Aggregation: For 3D segmentation, PointNet concatenates the result from an MLP layer (representing local information about each point) with the result of the symmetric functions, that is, the global descriptor. The authors state that this technique allows PointNet to predict per-point quantities that rely on both local geometry and global semantics, which warrants its performance in shape part segmentation and scene segmentation. However, by associating each individual point to the global geometry, this technique fails to correlate local points with each other and, by extension, capture local structure. This problem is addressed in the subsequent work of Qi et al. [18], PointNet++.

Joint Alignment Network: Besides permutation, point clouds are also invariant to geometric transformations, such as translations and rotations. This means that their intrinsic meaning does not change if they were to undergo one of these transformations. Taking this into account, PointNet employs T-Nets across its architecture, which is a simple network that predicts an affine transformation matrix and directly applies it to the coordinates of the input points. This causes the network to learn how to classify or segment point clouds regardless of orientation or size, for instance.

2.1.3.2 PointNet++

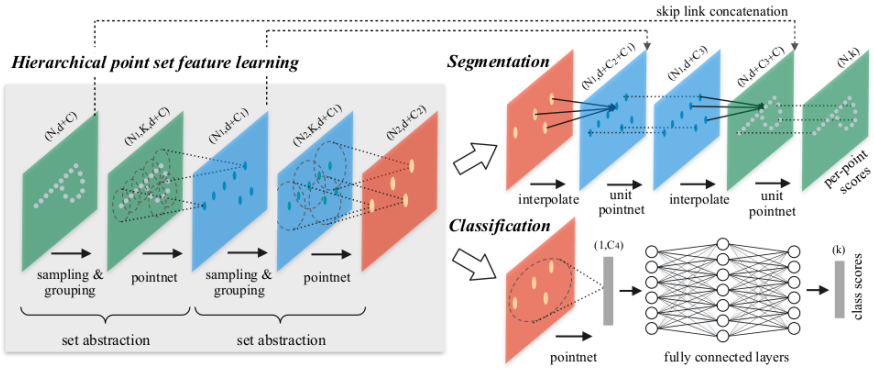
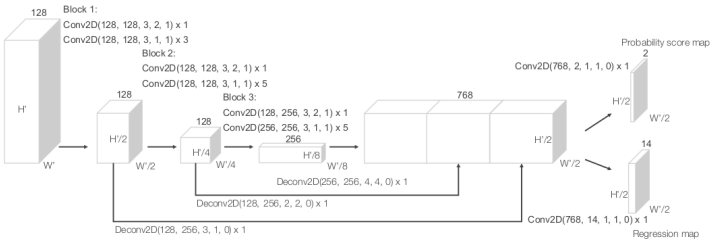


Figure 2.4: Illustration of the architecture of PointNet++ [18].

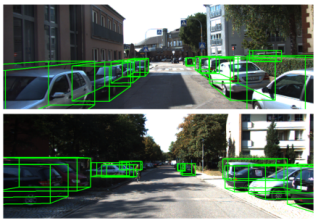
PointNet++ [18] is a neural network based on PointNet [17] with the goal of answering some of its demonstrated issues, such as the inability to capture local structure. Since point clouds are defined in a Euclidean space, the distance metric induces local neighborhoods in the point cloud that may exhibit different properties. For instance, density may not be uniform across different subsections of the point cloud. The ability to exploit the local structure has proven to be important for CNNs, these networks capture features from the input at increasingly larger scales, so the deeper layers of a CNN have large receptive fields with local patterns abstracted by previous layers. This allows for better generalizability to unseen cases.

To this end, PointNet++ processes a set of points in a hierarchical fashion: First, the point set is sampled using iterative farthest point sampling algorithm in order to increase the coverage of the entire point set (the sampled points define the centroids for the next stage). Second, points are aggregated into groups around each centroid following the Ball query algorithm, which finds all points within a certain radius of the centroid. Finally, a mini-PointNet processes each group individually in order to learn the local patterns of each group. This process, sampling, grouping, and PointNet, defines an abstraction set. PointNet++ stacks multiple of these sets in order to increasingly augment its receptive field and better extrapolate global features from the local patterns detected in previous sets.

2.1.3.3 VoxelNet



(a) VoxelNet RPN architecture



(b) VoxelNet Qualitative results projected on RGB images

Figure 2.5: VoxelNet RPN Architecture and Qualitative Results [37].

VoxelNet [37] is an end-to-end trainable deep neural network for point cloud 3D object detection that operates directly on the 3D input data. Its output corresponds to a set of bounding boxes in the original point cloud that contain the requested objects.

The proposed architecture consists of three blocks: feature learning network, convolution middle layers, and region proposal network.

Feature Learning Network: This first step transforms the point cloud into a volumetric representation and computes a feature vector for each of the voxels in order to aggregate their local information. This is performed over several stages, such as voxel partition and random sampling inside each voxel to increase both generalization and efficiency. However, one of the main contributions of this research is the design of a novel voxel feature encoding (VFE) layer: it extracts both point-wise and local features from each voxel, creating a high-level representation of the data. By chaining several VFE layers, the feature learning network is able to extract more complex and useful features for the following steps.

Convolutional Middle Layers: This block applies 3D convolutions on the produced volume to further aggregate the voxel-wise features within a progressively expanding receptive field, adding more context to the shape descriptor.

Region Proposal Network (RPN): The last block takes the volume from the previous step as input and produces several bounding boxes proposals that predict where objects of interest are located in the original 3D scene. Specifically, the output of the RPN is a probability score map and a regression map. Let us analyze the proposed loss function to better understand these elements, it is composed of two terms: (1) the regression term enables the network to learn the bounding boxes around the objects in space; (2) the classification term tells the network if, in some space, there is or not an object.

The regression term approximates a residual vector that represents anchors (bounding box candidates) and is composed of 7 regression targets (the center (3), its dimensions (3) and rotation angle w.r.t. the x-axis (1)). We divide anchors as positive or negative predictions, following that an anchor is positive if it has an Intersection over Union (IoU), also known as the Jaccard Index, with the ground truth of at least 0.6 (i.e., $IoU = \frac{|P \cap GT|}{|P \cup GT|} \geq 0.6$, where P is the anchor prediction, GT is the ground truth, and $|A|$ the cardinality of the set A). In other words, the model is adjusting the positive anchors to emulate the structure of the ground truth bounding boxes.

Regarding the classification term, the network performs a standard binary cross entropy throughout the 3D scene, representing where relevant objects are located. During training, the authors employ a different model of this architecture for each class of interest in the dataset. Therefore, in the RPN output layer, the probability score map indicates where objects are located in the original 3D space and the regression map defines bounding boxes encasing each of them.

2.1.4 3D Segmentation

The goal of 3D segmentation on point clouds is to separate the data into subsets according to some relation of the points. This relation can be semantic, instance-based, or part-whole. **Semantic segmentation** segregates the point cloud data according to the contextual meaning of the points. In our problem, this type of segmentation would tell us, for instance, which segments of the point cloud are part of the ground, the wires, and towers, among others. Whereas **instance segmentation** discriminates each individual instance of each class, following the example, the tower group would be further segmented into various groups, each representing one tower. **Part segmentation**, on the other hand, divides an object into its composing parts, for example, a chair into its arms, legs, and back. Although interesting, part-whole segmentation is not relevant to our problem.

Taking this into account, instance segmentation seems to align best with the main objective of this dissertation, segmenting each power line supporting tower in 3D point clouds. However, this kind of segmentation is more challenging than semantic segmentation since it requires more accurate inference over the points. Additionally, most proposed methods are employed on indoor point clouds, such as classrooms. On the other hand, semantic segmentation is widely used in outdoor datasets and is easier to compute. Moreover, we can take advantage of the disposition of towers in the point cloud: they do

not intersect and are relatively far from each other. Consequently, we can use a clustering algorithm, such as DBSCAN [41], to group the different towers together. This hypothesis has been tested during the exploratory data analysis stage of our work in the TS40K data, and demonstrated good results. The towers were perfectly segmented after tuning the parameters of the clustering algorithm (further details are available in Section 5.1.2).

Therefore, we believe that 3D semantic segmentation is the most suitable methodology to explore out of the three possibilities.

2.1.4.1 KPConv

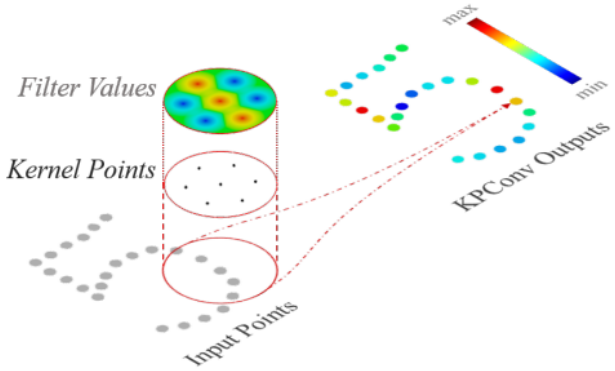


Figure 2.6: KPConv illustrated on 2D points. Input points with a constant scalar feature (in grey) are convolved through a KPConv that is defined by a set of kernel points (in black) with filter weights on each point [20].

KPConv [20] is a continuous point convolution-based architecture that is mainly used for classification or segmentation problems. It introduces a novel point convolution operator, named Kernel Point Convolution (KPConv), illustrated in Figure 2.6. This operator is defined by a set of kernel points, each containing a kernel weight, that define both a 3D filter and an area of application in the 3D space where the convolution operation will take place. Specifically, KPConv is formulated as a point convolution inspired by image convolution. In images, the indices of pixels define their location and the associated RGB values are considered the features where the kernel will be applied. In 3 dimensions, we can see the point coordinates (x, y, z) as their location and any additional features F, such as color, to be the information considered by the kernel points. The convolution of F by the kernel g at a cloud point p is defined as a sphere with radius r and center c where g is applied to the points that fall under the scope of said sphere. The function g defines the relation between the cloud points and kernel points through a correlation function h, which outputs a higher result the closer to points are.

Taking this into account, the positions of the kernel points are crucial for the performance of this architecture. Thomas et al. [20] present two ways of defining their position: through a rigid or deformable perspective. In the former, the disposition of the points is

solved by an optimization problem where points are constrained to stay away from each other, but to stay within the bounds of the sphere and one of them is forced to be at the center. The latter however is not as straightforward, the deformable KPConv architecture is more flexible than its rigid counterpart because it learns the best disposition of the kernel points. Specifically, it learns a set of shifts for every convolution location by introducing *fitting* and *repulsive* regularization terms in the loss function used to train the model.

2.1.4.2 RandLA-Net

RandLA-Net [21] introduces a computationally and memory-efficient network for large-scale point cloud segmentation.

Although state-of-the-art methods present remarkable all-around results in this field, they are often limited by computationally expensive operations, verbose pre-processing or are only able to work with small subsets of points from the point cloud, forcing the data to be divided into segments which might cause discrepancies in the final semantic segmentation of the point cloud.

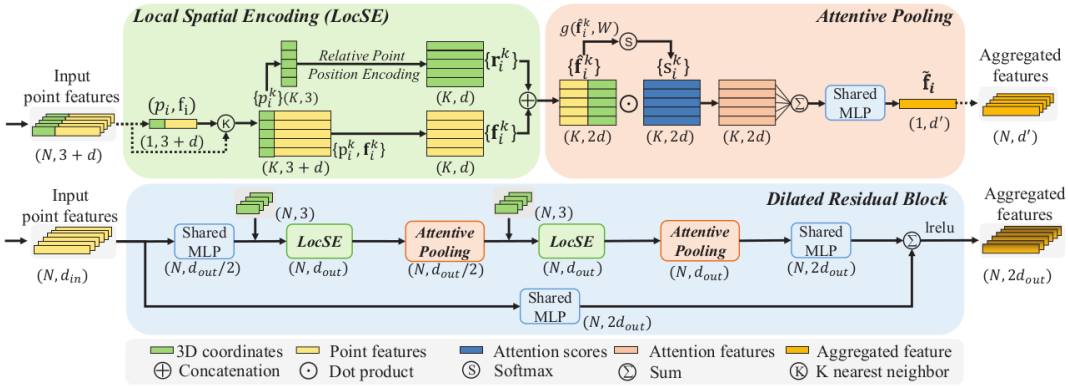


Figure 2.7: Modules used in RandLA-Net [21].

In order to overcome these challenges, the key approach of Hu et al. [21] is to use random point sampling instead of more complex point selection heuristics. However, random sampling might discard important features of the point cloud. To answer this problem, the authors introduce a novel local feature aggregation module to preserve the geometric details for each 3D point.

Local Feature Aggregator (LFA) takes advantage of 3 modules:

1. *Local Spatial Encoding (LocSE)* takes the k-nearest neighbors (k-NNs) of each point x in the provided point cloud and concatenates its features with an encoding of its relative position on the point cloud. This allows for the definition of the local geometric structure of the neighboring points around x.

2. *Attentive Pooling*: this neural unit is used to aggregate the set of neighboring features. The use of max or mean pooling results in information loss, so it is discouraged. Therefore, the authors take advantage of attentive pooling in order to automatically learn the important local features. This is performed by computing attention scores through a shared MLP for each feature of each neighboring point. Then the most important features are selected by a weighted summation, culminating in an informative vector that describes the geometric patterns and features where x is located.

3. *Dilated Residual Block* is a stack of multiple LocSE and Attentive Pooling units with skip connections. A LocSE/Attentive Pooling operation results in x observing K neighboring points, then, by repeating this operation, its neighborhood increases to K^2 points. Consequently, the receptive field of each point is significantly dilated, which helps to preserve most of the geometric details of the 3D scene, compensating for the information loss during the random sampling stage.

The architecture of RandLa-Net follows a 3D encoder-decoder style with skip connections. The encoding is performed by LFA layers followed by random sampling, while the decoder is done by up-sampling the data through shared MLPs.

RandLA-Net was applied to the main benchmarks in point cloud learning, achieving performance that surpasses the state-of-the-art approaches. In terms of efficiency, the network can process 1 million points in 185 seconds with 1.24 million parameters. For comparison, KPConv [20] takes almost 4 times the time that RandLA-Net does to process 0.5 million points with 14.9 million parameters. When compared to SPG [42], this approach is 200x faster. Further details on this are available in Section 2.1.8.

2.1.5 3D Object Detection

3D object detection is quickly growing as an important field of study with numerous real-life applications, such as autonomous navigation and virtual reality. These methods take a 3D scene as input and output 3D bounding boxes encasing objects of interest. A bounding box, as the name suggests, delineates the limits of the space occupied by an object as illustrated in Figure 2.5b. Traditionally, they are defined by a center and a minimum and maximum 3D point that define, respectively, the lowest and highest corner of the box. For instance, 3D object detection is useful in autonomous driving scenarios. By enclosing scene elements of interest, such as pedestrians and cars, in bounding boxes, the vehicle can better assert the risk of collision with these objects and avoid accidents.

Objects are more accurately described in three dimensions than with images, where they are subject to perspectives, occlusion, color, and lighting. However, unlike images, LiDAR point clouds are usually sparse, lack structure, and point density can vary depending on the range and relative position of the 3D sensors. In order to address these challenges, existing 3D detection methods can be classified into two categories in terms of

point cloud representation: grid-based and point-based methods. The former gives structure to point clouds in order to take advantage of 2D/3D convolutions, while the latter directly extracts discriminative features from the raw point cloud. Generally, the discretization step of grid-based methods entails information loss that inevitably degrades their accuracy. Point-based methods, on the other hand, achieve more precise object localization, but their complex architecture usually implies higher inference and training times.

In the following subsections, we present two proposals that lead the 3D object detection vanguard for point clouds along with their results on important datasets, such as the KITTI 3D detection benchmark [11].

2.1.5.1 PointRCNN

PointRCNN [19] is a neural network for 3D object detection from raw point clouds. The proposed architecture is divided into two stages:

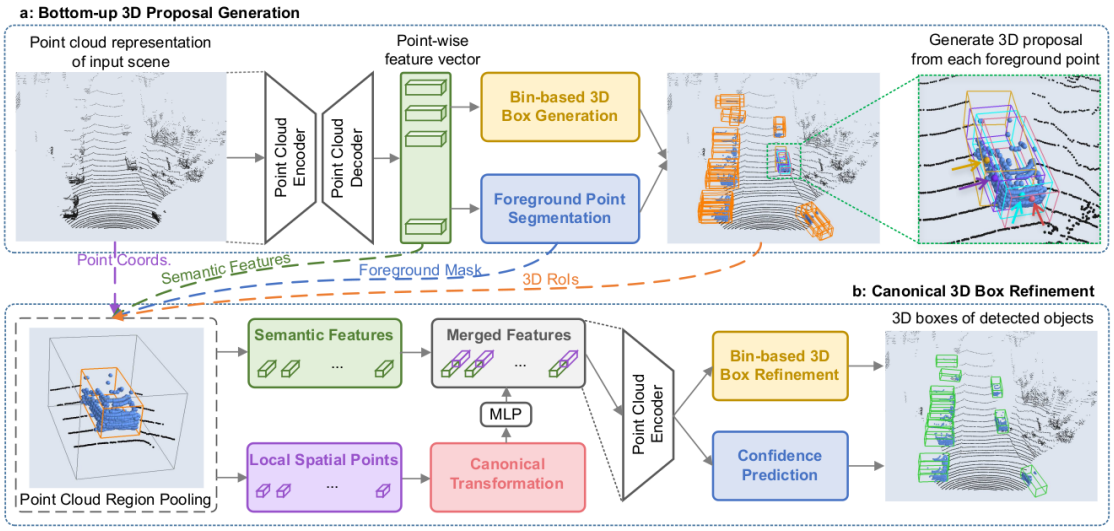


Figure 2.8: PointRCNN architecture [19].

The first stage is dedicated to generating 3D bounding boxes proposals in a bottom-up scheme. It starts by taking the raw point cloud and learning a point-wise feature vector by the means of a backbone network: PointNet++ [18]. Then, the network learns a foreground point segmentation of the point cloud and, at the same time, generates 3D region proposals for each of the foreground points.

In the second stage, the proposals from stage-1 are combined with point cloud data in order to refine the proposed 3D bounding boxes. Specifically, the raw data points are used to compute local spatial feature vectors, which then suffer a canonical transformation and are combined with the semantic features and foreground masks computed in stage-1. This allows the network to build a robust representation of the point cloud to then refine the 3D bounding boxes from stage-1.

PointRCNN achieves state-of-the-art performance on 3D object detection on the KITTI dataset, one of the main benchmarks of the area. Surpassing VoxelNet [37] on detecting cars and cyclists, whereas the latter still maintains the best performance in detecting pedestrians. From ablation tests, the authors confirmed that the performance of the model increases when using data augmentation. This implies scaling and rotating the point cloud (around the z-axis) by some factor. They also augmented the data set by randomly selecting non-overlapping ground truth boxes and insert them on the training scenes.

2.1.5.2 PV-RCNN

Shi et al. [43] present a novel 3D object detection framework named PointVoxel-RCNN (PV-RCNN) for accurate object detection from point clouds. Their proposal integrates both 3D voxel-based convolutions to efficiently summarize the 3D scene and PointNet-based abstraction sets to learn more discriminative point cloud features in order to refine the region proposals. Therefore, we can divide the architecture of PV-RCNN in two main stages:

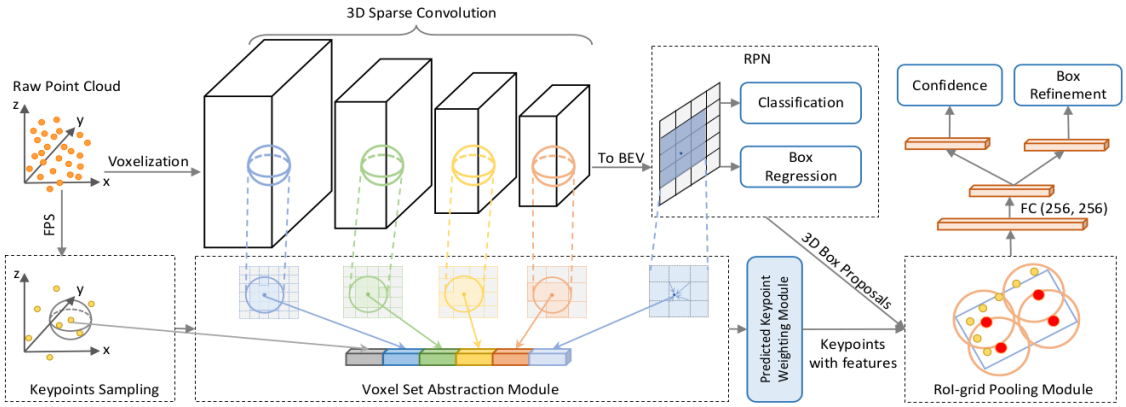


Figure 2.9: PV-RCNN architecture [43].

Voxel-to-keypoint Scene Encoding via Voxel Set Abstraction: The point cloud is first divided into small voxels in order to utilize a series of 3D sparse convolutions to gradually convert the point clouds into feature volumes. Then, such volumes are converted into 2D bird-view feature maps in order to generate the first 3D box proposals.

As it is, the second stage of the network cannot take place because the PointNet-based abstraction sets expect the 3D input data to have the structure of a point cloud. To solve this, PV-RCNN represents the entire 3D scene in a small number of keypoints, which serve as a bridge between the 3D voxel CNN feature encoder and the proposal refinement network. Then, the previously learned voxel-wise features volumes and the BEV feature maps are summarized and embedded into such keypoints. This way, the produced 3D scene preserves its structural information and is enriched by the features extracted from 3D CNNs.

Keypoint-to-grid RoI Feature Abstraction for Proposal Refinement:

In this stage, the 3D proposals are refined by the means of RoI-grid pooling via set abstraction. Essentially, the 3D proposals previously computed may have very different shapes, but these need to be presented with a standard form in order to feed them to the last section of the network (a set of fully connected layers followed by the output layers).

To this end, object detection algorithms use RoI (region of interest) pooling to transform the region proposals to a standard shape. Many state-of-art methods simply average all point-wise features within the region proposal. Shi et al. [43] on the other hand, employ a novel method that captures much richer contextual information with flexible receptive fields for each proposed region. Specifically, they first sample a number of grid points within each 3D proposal and identify the neighboring keypoints for each of them. Then, a PointNet block [17] is adopted to obtain a feature vector for each grid point by aggregating the features of their respective keypoints neighbors.

Finally, all grid-point features from the same region are transformed into a fixed-size vector using a two-layer MLP, resulting in a representation of the overall region proposal.

PV-RCNN performed very well when tested in the KITTI dataset for object detection [11], outperforming all previous state-of-the-art methods with remarkable margins. The network is further tested on the Waymo Open Dataset [44], achieving once again excellent results and outperforming previous methods.

2.1.6 Benchmark Datasets

In this section, we present a number of popular benchmark datasets. They provide a means to fairly compare diverse approaches to 3D learning tasks, such as 3D object detection and 3D semantic segmentation.

- *ScanNet* [45]: is an indoor dataset developed by Stanford University. It contains 1513 scanned scenes, including 2.5 million RGB-D images from diverse indoor environments. In the semantic segmentation task, the dataset provides ground truth segments with 20 categories.

- *Waymo Open Dataset* [44]: contains high-resolution sensor data in a variety of conditions. It focuses on 4 object classes: vehicles, pedestrians, cyclists and signs. It contains almost 400 000 frames of high-quality LiDAR data and over 12 million 3D bounding boxes for 3D object detection.

- *KITTI* [11]: is one of the most popular and regarded datasets for use in robotics and autonomous navigation. It consists of numerous traffic scenarios recorded with a variety of sensors, such as high-resolution cameras and 3D laser scanners. For 3D object detection, the dataset is usually divided according to object category and difficulty of identifying it. The most used categories are cars, pedestrians and, cyclists, each with easy, moderate, and hard difficulty subsets. In total, it contains

7481 training and 7518 testing point clouds. When it comes to semantic segmentation, the *SemanticKITTI* [10] dataset is based on the object detection dataset with semantic labeling included. It provides 23201 point clouds for training and 20351 for testing, with a total of 28 categories.

- *Semantic3D* [9] is a point cloud dataset of outdoor scenes with over 3 billion points. It contains 15 training and 15 test scenes annotated with 8 class labels. This dataset includes numerous and diverse urban scenes, such as churches, streets, railroad tracks, villages, soccer fields,and castles. The provided point clouds are scanned statically with state-of-the-art equipment and contain very fine details.

- *SensatUrban* [30] consists of large areas from three UK cities, covering more than 7km^2 of landscape, in the form of 3D point clouds. It is composed of 2847 million 3D points, each one labeled with one out of 13 classes, such as ground, vegetation, building, parking, car, and water, among others.

2.1.7 Evaluation metrics

Depending on the point cloud tasks at hand, different evaluation metrics have been proposed to evaluate and compare proposed methods.

For 3D point cloud segmentation, *Overall Accuracy* (OA) and *mean Intersection over Union* (mIoU) are the most used criterion for performance evaluation [9, 10]. OA is simply the average of correctly segmented objects; IoU (intersection over union) represents how much the predicted segmentation intersects with the ground truth, mIoU is an average over the classes in the dataset. IoU may also be employed in some approaches to distinguish the performance of the model in the different classes of the dataset.

For 3D object detection, *Average Precision* (AP) is the most frequently used criterion along with mIoU. AP is defined as the area between the precision-recall curve, which translates to the number of correctly identified objects, averaged across the classes in the dataset.

2.1.8 Results

Method	Speed (fps)	Cars			Pedestrians			Cyclists		
		E	M	H	E	M	H	E	M	H
PointRCNN [19]	10.0	86.96	75.64	70.70	**47.98**	**39.37**	**36.01**	74.96	58.82	52.53
PV-RCNN [43]	12.5	90.25	81.43	76.82	-	-	-	**78.60**	**63.71**	**57.65**
Voxel R-CNN [46]	-	90.9	81.62	77.06	-	-	-	-	-	-
STRL + PV-RCNN [47]	-	91.08	81.63	**79.39**	-	-	-	-	-	-
SE-SSD [48]	25.0	**91.49**	**82.54**	77.15	-	-	-	-	-	-
VoxelNet [37]	2.0	77.47	65.11	57.73	39.48	33.69	31.51	61.22	48.36	44.37

Table 2.1: Comparative 3D Object Detection Results on the KITTI Test 3D Detection Benchmark. *3D bounding box IoU threshold is 0.7 for cars and 0.5 for pedestrians and cyclists. The presented results refer to AP, with % omitted for simplicity. 'E', 'M', and 'H' represent easy, moderate and hard classes of objects, respectively. The symbol '-' means the results are unavailable.* Values compiled from papers [31, 35]

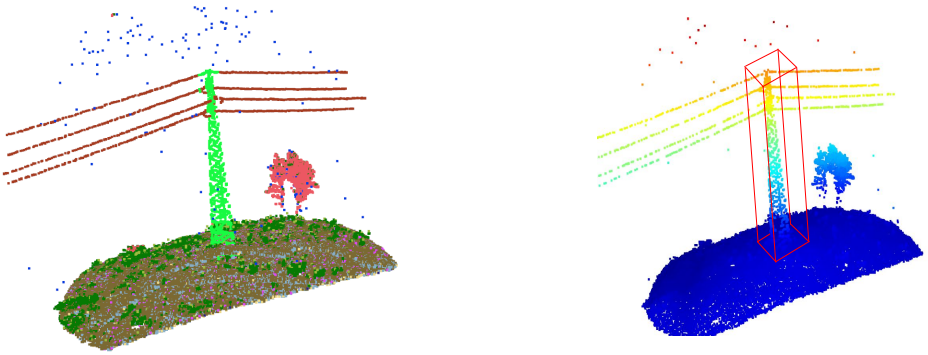
Method	Semantic3D		ScanNet	SensatUrban		SemanticKITTI
	OA	mIoU	mIoU	OA	mIoU	mIoU
(AF)2-S3Net [49]	-	-	-	-	-	**70.8**
Cylinder3D [22]	-	-	-	-	-	68.9
KPConv[20]	74.6	-	68.6	**93.20**	**64.5**	58.8
RandLA-Net [21]	77.4	**94.8**	-	89.78	62.8	53.9
PointNet++ [18]	-	-	33.9	80.78	58.13	20.1
PointNet [17]	-	-	-	84.30	52.53	14.6
O-CNN [40]	-	-	76.2	-	-	-
Mix3D [50]	-	-	**78.1**	-	-	-
FG-Net [51]	**78.2**	93.6	69.0	-	-	63.1

Table 2.2: Comparative Semantic Segmentation Results on the Semantic3D [9], ScanNet [45], SensatUrban [30] and SemanticKITTI [10] datasets. *The presented results refer to OA and mIoU, with % omitted for simplicity. The symbol '-' means the results are unavailable.* Values compiled from papers [31, 35]

2.1.9 3D Semantic Segmentation vs. 3D Object Detection

The goal of this dissertation is two-fold. First, we want to determine the coordinate location of power line supporting towers in a given point cloud. This way, the extensive 3D scenes retrieved by the utilities company can be automatically sectioned, and the maintenance personnel can focus on inspecting the power grid and other tasks with higher added-value than manually sectioning point clouds. On a second note, we wish to take the first step towards automatic risk assessment of collisions between a transmission grid and its environment by detecting power line supporting towers. Thereafter, other 3D scene elements can be detected and, therefore, the distance between different objects can be calculated.

Upon analyzing the state-of-the-art methodologies for learning from 3D point clouds, 3D object detection is the most direct approach. Methods receive a 3D scene as input and produce multiple 3D bounding boxes that estimate where pertinent objects are located



(a) 3D semantic segmentation. (b) 3D object detection.

Figure 2.10: Intended output of both Deep Learning methodologies on the TS40K dataset.

within the scene. In our case, this translates to developing a Deep Learning model that predicts 3D bounding boxes encasing the numerous towers in the 3D input as illustrated in Figure 2.10b.

Alternatively, 3D semantic segmentation models provide a different solution: they divide the input data points into a number of groups according to their contextual meaning within the 3D scene. For instance, all power line supporting towers in the point cloud may form a single group representing tall vertical objects with several lines orthogonal to them (i.e., the power lines). This idea can then be extended to other concepts in the 3D scene such as the ground, the power lines, the vegetation, and even noise can be aggregated into their own semantic groups as shown in Figure 2.10a.

Taking this into account, semantic segmentation models offer two advantages when compared to object detection:

First, they allow us to embed meaning into 3D scenes because, during training, models learn how to aggregate points according to the defining attributes of each semantic group. Following the previous example, points that compose a tower are segmented into a group consisting of tall vertical objects, whereas ground points are aggregated in a different semantic group seeing as they are not tall nor vertical. In contrast, object detection models do not add any interpretation to the point cloud, they solely adjust their bounding box predictions to the ground truth during training.

Secondly, segmentation methodologies are better suited for the following stages of the project. Although object detection can precisely determine the coordinate location of the supporting towers, it is more challenging to build a DL model that detects multiple objects, such as power lines and the revolving flora. Additionally, the predicted bounding boxes normally encase plenty of unoccupied space, which would hinder the distance estimation between relevant objects. Conversely, 3D semantic segmentation methods divide point clouds into several semantic groups by definition. Estimating the distance between 3D scene elements is more accurate because segmentation models effectively classify every single 3D point in a scene. Thus, this approach precisely outlines the

structure of objects, instead of computing bounding boxes containing unoccupied space, which leads to a better distance estimation between relevant objects.

Therefore, our model follows a 3D semantic segmentation framework, as well as the state-of-the-art methods we focused on.

2.1.10 Analysis of SCENE-Net against State-of-the-Art Methods

State-of-the-art methods [18, 20, 22, 23, 42, 49] have progressively increased their complexity to a remarkable degree in order to yield good performance in real-life scenarios, such as autonomous driving [35]. These models have millions of trainable parameters, a non-trivial amount of training time on powerful GPUs, and specific hardware requirements [21, 35, 42]. This is a consequence of having to fine-tune so many parameters combined with the massive amounts of data that describe 3D point clouds, which need to be sectioned, sampled, or even preprocessed by voxelization or multi-view methods. Additionally, most proposals are tailored to boost performance in urban settings (e.g., Semantic3D [9], SensatUrban [30] and SemanticKITTI [10]), where data are sparse, objects are often occluded and may demonstrate anisotropy w.r.t. density.

Despite their deep in-house knowledge in their specific fields, traditional companies, such as utilities, are limited when applying ML models to real-world point clouds because of low expertise in data science, and low data and computational availability to deploy state-of-the-art models.

With this in mind, we propose a novel DL method that is specifically designed to detect supporting towers in 3D point clouds. SCENE-Net achieves effective results with fast training times (85 minutes) on a regular laptop and shows an inference time of 20ms. To do this, we take advantage of group equivariant non-expansive operators (GENEOs) [28], a mathematical theory that allows us to greatly decrease the number of trainable parameters, as well as reduce the complexity of a DL model by embedding it with interesting prior knowledge. Furthermore, GENEOs allow for the interpretability of the model, in the sense that the contributions of the different parameters are comprehensible to the human mind. Following the definition of interpretable ML provided by Lipton [52] described in 2.3.2, our model is simulatable (i.e., understandable in a reasonable amount of time), decomposable (i.e., the input, parameters, and calculations of SCENE-Net provide an understanding of its inner workings), and its learning process is transparent (explained in detail in Section 4.5).

Typical DL models tune their trainable parameters in order to recognize patterns and extract them from the input data. Conversely, we define tailor-made kernels that encode the relevant patterns that characterize objects of interest for the problem. For example, let us consider that a conventional CNN is employed to detect towers in the point cloud data. The CNN would be composed of multiple 3D convolutional layers that could ideally learn to extract patterns such as verticality, invariance in regard to rotations along the z-axis, cylindrical patterns, and orthogonal segments on higher z coordinates, among

others. Alternatively, by devising a model that focuses purely on detecting towers, we can encode the mentioned patterns as kernels directly into the model, instead of having the network learn them. Additionally, it is not the kernel weights that are trained in our model, since this would disrupt the prior knowledge they define, instead, the error is backpropagated to the shape parameters of each operator, and the kernels are recomputed after each forward pass. This results in a drastic decrease in trainable parameters when compared to conventional DL models. In our case, SCENE-net exhibits a total of 11 trainable parameters, whereas a CNN with an identical design has 2190 parameters (more details on this in Section 5.2.2).

Since there is no free lunch, the applicability of our model is restrained to a smaller subset of problems (i.e., ones that imply detecting pole-like structures in 3D point clouds) than traditional DL models. Nonetheless, the development of state-of-the-art methods nowadays is mostly problem-oriented. For example, most advancements in 3D semantic segmentation were tailored for autonomous navigation [21, 24, 35, 49]. On the other hand, our approach guarantees the interpretability of its decisions, whereas conventional methods do not. That is, the results given by SCENE-Net can be understood by humans since we are the ones providing the patterns for the network to extract from the data. This contrasts with the behavior of conventional models, which are often regarded as black boxes because their decisions cannot be interpreted or explained. However, the interpretability of SCENE-Net entails a knowledge engineering phase to develop the necessary GENEO-kernels that describe properties of interest. In addition, the alteration of the problem statement is not solved by simply retraining the model, GENEO-based models may require more operators to describe different properties in the new problem. For example, imagine that SCENE-Net also needed to detect the power lines, this requires the development of more GENEO-kernels to encode the properties of these elements. Contrastingly, black box models would only have to be retrained with a dataset that encompasses the detection of power lines in the ground truth.

2.2 Power Line Segmentation from 3D Point Clouds

Generally, transmission line inspection is performed by on-site maintenance personnel and manned helicopters that examine the power grid with portable devices or the naked eye. These methods are very expensive, inefficient, extremely demanding and with poor working conditions for the staff. Often, power grids are built on highly irregular terrains in order to avoid urban areas, this handicaps on-site inspections, even with land vehicles. Therefore, advancing to safe and efficient inspections is crucial for transmission system operators. To this end, UAVs carrying high-precision LiDAR sensors are deployed to scan the power grid and capture a 3D point cloud representation of the environment.

In this section, we will present three state-of-the-art methods that take advantage of Machine Learning strategies to aid power line inspection, and compare them to our proposal.

2.2.1 Electric power line patrol operation based on vision and laser SLAM fusion perception

With affordable high-level sensors, such as depth cameras and laser range finders, simultaneous localization and mapping (SLAM) is now used in multiple practical applications in various fields. For example, SLAM is integrated in home robot vacuums so that they can understand the layout of a room and clean it efficiently.

The work of Ding, Wang, and Wu [27] proposes the fusion of vision and laser SLAM strategies in order to extract a semantic segmentation of power lines. A UAV is equipped with a semantic SLAM algorithm that allows it to track the power line and avoid obstacles while scanning the area. During this process, the UAV captures a large-scale LiDAR point cloud of the scene (laser SLAM) and image data collected by a camera (vision SLAM). The camera images are fed to a deep neural network, Deeplabv3+ [53], for 2D semantic segmentation. Then, the point cloud data is projected to the image, so that the semantic information can be transferred to the 3D scene.

The main goal of this pipeline is to improve the flight control design of UAVs. The images with the power line semantic segmentation information are still subject to manual inspection by the staff. Since the 2D point clouds are projected into two dimensions, this method can be considered a multi-view-based approach to 3D semantic segmentation. However, the authors do not reconstruct the 3D scene from the 2D raster maps, which means that distance estimation is performed with two-dimensional data. Therefore, the accuracy of this prediction may be hindered by the loss of depth information when compared to other methods.

Our proposal differs from this model, we do not focus on the automation of the flight control of UAVs, instead, our goal is to provide a trustworthy semantic segmentation framework of supporting towers to power grid inspectors. In addition, we do not use 2D projections, so our distance estimations are based on 3D data and, therefore, more accurate.

2.2.2 Research on Point Cloud Power Line Segmentation and fitting algorithm

Guo et al. [26] propose an algorithm for power line segmentation by combining xy-plane projections and clustering algorithms on power grid point clouds. A power grid point cloud only contains points pertaining to the transmission system. The authors first project the data onto xy-plane with two models, one that assumes that power lines coincide in the projection (i.e., power lines organized in a vertical arrangement), and another that does not work on this assumption. The k-means clustering algorithm is applied to determine the number of power lines and segment them. This power line segmentation is then completed by following a straight line algorithm that fills the missing spots in the power line 3D representation. Then, the model that assumes a coincident power line projection on the xy-plane is used to determine the segmentation of power lines in a

vertical arrangement. This is achieved by measuring the relationship of the z coordinate values of each cluster.

The accuracy of this model is measured according to the number of power line 3D points correctly extracted from the power grid point clouds. Our model measures its performance according to Precision and Recall (see Section 5.2.4.1 for more details). In comparison, our proposal works with complete 3D scenes, seeing as this is the type of data that transmission system operators usually retrieve from LiDAR scans. Point clouds containing just the power grid are not as useful for utility companies, since this type of data does not allow them to assess the risk of contact with other scene elements. So, the scope of application of this model is very limited.

2.2.3 Study on segmentation algorithm with missing point cloud in power line

The work of Tao et al. [25] proposes a power line reconstruction algorithm that also achieves power line point cloud segmentation. First, it performs a rough classification of an input point cloud by creating a voxel grid representation of the 3D data and then uses reasonable elevation thresholds to quickly segment non-power line points. From this step, the ground and vegetation are practically segmented from the power line, intersection line, and part of the supporting towers. Secondly, the resulting point cloud containing the power grid is projected into the xy-plane. The power lines and cross line points are then extracted using the random sampling consistency algorithm (RANSAC) [54], which calculates the mathematical model parameters of these two scene elements in order to segment them. Lastly, the authors reconstruct power lines with missing sections using polynomial equations projected in xz-plane or yz-plane, which are transformed into an extreme value problem.

Similar Guo et al. [26], Tao et al. measure the accuracy of their model according to the number of power line 3D points correctly extracted from the total number of power line points. Seeing as the main focus of our dissertation differs from this paper, their performances are not comparable.

Nonetheless, the rough classification stage of this strategy does not take into account irregular terrains that may not be subject to the same reasonable height thresholds used in this model. So, this approach is only effective with somewhat flat terrains. For instance, the developed TS40K dataset contains sections with very irregular terrains where height thresholds would not produce an accurate rough classification.

Additionally, this work was tested exclusively with high-voltage power grid 3D data, where the height thresholds are more applicable due to the height difference between high voltage towers and the rest of the environment. This difference is not as prominent with other voltage power grid areas, where the vegetation may easily exceed the transmission system in terms of height.

2.3 Interpretability and Explainability of ML Methods

In this Section, we discuss the importance of intrinsically interpretable Machine Learning strategies. First, we explain why black box models are not suitable for high-risk domains, such as healthcare or power grid maintenance. Then, we introduce the notion of explainable and interpretable ML, along with their advantages and drawbacks. Lastly, we justify why the transparency provided by interpretable ML trumps the reasoning of explainable methods for high-stakes scenarios and traditional companies, like utilities.

2.3.1 Why are black box Models Not Enough?

The widespread application of ML models to real-world problems with high-risk decision-making, such as healthcare [55], autonomous navigation, or power grid maintenance, has the potential to greatly affect human lives. As such, interpretable and explainable Machine Learning strategies stem from the need for trustworthy and intelligible ML models. That is, models whose inner workings are comprehensible by humans, and techniques that can explain the predictions of black box models. Essentially, a black box model is too complicated for a human to understand, so they are usually described by their input and output. For example, traditional neural networks are notably opaque, because they do not provide any reasoning behind their predictions. A network may classify images of dogs and cats with extreme accuracy, but does not inherently explain why or how it does so.

This need for an explanation or an understandable causal relation between the input and output is not only present in high-risk scenarios, but also in traditional companies, where practitioners of other disciplines need to clearly understand and apply trustworthy models to their own data. For instance, maintenance personnel of a utility company may want to understand why an ML model detects certain objects in 3D point clouds. This not only provides transparency but also enables better troubleshooting strategies.

2.3.2 Interpretability and Explainability

The concept of interpretability and explainability are closely related, and often literature does not differentiate the two, seeing as there is no generally agreed definition. Therefore, in this dissertation we adhere to the distinction between interpretable and explainable ML provided by Rudin [56]. Specifically, Rudin defines interpretable ML as models whose design is intrinsically interpretable. That is, the parameters of the model should be meaningful and inherently explain its predictions. For example, linear regression with a small number of parameters is interpretable, each model coefficient represents the importance of the respective input feature to the predictions of the model. In turn, explainable ML tries to provide *post hoc* explanations for existing black box models. That is, given an opaque model, explanation techniques try to find reasons behind its predictions. For

example, attention methods can pinpoint what parts of an image are relevant or omitted by a classifier.

Explanations can take numerous forms: logic, visual, symbolic, textual, and input data points, among others. In recent literature, neuro-symbolic explanation systems [57] and entropy-based networks [58] have been proposed to relate concepts onto target classes and provide logic explanations of the decision process. However, these techniques depend on symbolic input and output spaces. In real-world problems, such as computer vision, it is difficult to define a consistent ontology for concept representation. In addition, [52, 56] argue that explanations do not have perfect fidelity with respect to the original model. If that were the case, the explanation would be equal to the model itself, so the model would be considered interpretable. In other words, explanations provided by *post hoc* techniques cannot perfectly decipher the reasoning of a black box, instead they provide possible reasons for its predictions, which might not be faithful to what the original model is actually computing.

Contrary to opaque methods, interpretable models provide an inherent understanding of their predictions through their parameters. They are also known as white-box models, since their transparency contrasts with the opaqueness of black boxes. In general, this intrinsic interpretability is achieved by imposing constraints that reflect some domain knowledge in the model definition. For instance, one can impose sparsity, causality, or physical constraints in order to uphold desirable properties [56, 59].

Lipton [52] defines transparency in white-box models through three properties: (i) *simulatability* indicates that a model should be comprehensible to a human, taking into account both the input and model parameters, in a reasonable amount of time; (ii) *decomposability* states that every part of a transparent model, its input, parameters, and calculations should provide an intuitive explanation; (iii) *Algorithmic transparency* alludes that transparency should not only be endowed to the model, but also to its learning process.

This definition suggests that, in order to achieve transparency, a model should be simple [52]. Moreover, by enforcing domain constraints on the model, it may miss some hidden patterns that black box models could uncover [56]. This leads to a belief that it exists a trade-off between accuracy and interpretability. I.e., transparent models cannot achieve the same performance as black box models because they are subject to both constraints and simple designs. However, the introduction of novel methods, such as concept whitening [60], refutes this thesis. The work of Chen, Bei, and Rudin [60] proposes a white-box model for image recognition that forces the latent space of a convolution to be aligned along the axis of concepts. Specifically, concept whitening decorrelates the post-convolution latent space so that the covariance matrix between channels is the identity. That is, it disentangles the latent space so that it aligns with given concepts. They experiment with this idea in several case studies and conclude that concept whitening can be added to CNNs without loss in performance using black box models as a baseline.

Even though adding problem-specific constraints to a model does not necessarily hinder its performance when compared to traditional unconstrained methods, they make white-box models significantly harder to construct and compute [56]. Not only do interpretability constraints, such as causality, require domain and mathematical expertise, but they also lead to computationally hard optimization problems. In contrast, it is much easier to employ a black box ML method than to define, troubleshoot and solve problems with interpretability constraints. However, high-risk scenarios and the general application of ML models in vital companies, such as utilities, often justify the extra effort of designing interpretable ML.

3

Group Equivariant Non-Expansive Operators (GENEOs)

3.1 Introduction

Machine Learning-based strategies have an essential role in numerous real-world applications nowadays, from autonomous driving to medical imaging and diagnosing. Their wide use variety stems from the fact that raw data is sufficient to train a model, and achieve a performance higher than when the task is performed by humans. However, most Machine Learning methods employed in critical fields, such as healthcare, are considered black boxes, i.e., models whose results cannot be explained or interpreted with certitude by humans. This raises urgency for the development of theories that formally describe the development of interpretable Deep Learning models.

Group equivariant non-expansive operators (GENEOs) are the building-blocks of a novel mathematical framework [28, 29] that formally describes ML agents as a set of operators acting on the input data. These operators provide a measure of the world, just as CNN kernels learn essential features to, for instance, recognize objects. Thus, such ML agents can be thought as *observers* that analyze data. They transform it into higher-level representations while respecting some set of properties (i.e., group of transformations). An appropriate observer transforms data in such a way that respects the right group of transformations, that is, it commutes with these transformations. Formally, we say that the observer is *equivariant* with respect to a group of transformations. This way, prior knowledge is effectively encoded into an ML agent, we determine the transformations preserved by our observer, which ideally describe some features of interest in the original data.

Equivariance can be seen as a form of symmetry between two function spaces with respect to an action group, meaning that a function space is transformed into another function space while preserving a group of symmetries. In practical terms, let us consider a function f and a symmetry group S, $f : X \to Y$ is considered equivariant with respect to S if: $f(g \cdot x) = g \cdot f(x) \forall g \in S, x \in X$. Moreover, the concept of equivariance generalizes

invariance, in the sense that a function is deemed invariant w.r.t. a symmetry transformation if its values remain unchanged after said symmetry is applied. For example, the area of a triangle is invariant to Euclidean transformations (e.g., translations and rotations), since rotating a triangle does not modify its area. On the other hand, a centroid does not demonstrate this property, since moving the triangle will also cause its centroid to move. Instead, the centroid is equivariant to Euclidean transformations, meaning that applying the transformation and then computing the centroid produces the same result as first computing the centroid, and then applying the same transformation.

Interestingly, some Deep Learning techniques are already adept of this concept. Convolutional neural networks learn different kernels that transform the input image into a new one that, for example, is more easily classified as training progresses. The learned convolutional kernels are GENEOs, since convolutions are operators that transform the input into a new function space and that, by definition, are equivariant with respect to translations. Bergomi et al. [28] believe that the restriction to a specific family of operators and the equivariance with respect to interpretable transformations are key aspects for the success of this architecture.

3.2 Data Representation

The GENEO framework takes advantage of topological data analysis (TDA) to describe the input data as topological spaces. These spaces are sets endowed with structure (i.e., a topology) where the properties of geometric objects are preserved under continuous deformations, such as stretching. For example, Euclidean spaces are topological spaces, seeing as their metric defines a topology.

A crucial concept in topology is homeomorphism, a continuous function between topological spaces that has a continuous inverse function. Thus, it preserves all topological properties of a given space when transforming it. Two spaces are topologically equal (i.e., homeomorphic) if they present a homeomorphism between them. For instance, a coffee mug and a donut are homeomorphic to each other, since we can deform one under some continuous function that would result in the other, and vice-versa.

The main objective of employing TDA is to represent data as a continuous real-valued function space, since GENEOs are only applicable under such conditions. The authors take advantage of persistent homology, a branch of topology that captures topological information at multiple scales. By representing input data as topological spaces, patterns can be recognized with greater ease and transformed into more useful representations, which improves the effectiveness of the chosen operators.

Specifically, a set of data X is represented by a topological space Φ with admissible functions $\varphi\colon X \to \mathbb{R}^3$. Φ can be thought of as a set of admissible measurements that we can perform on the measurement space X. For example, images can be seen as functions assigning RGB values to pixels. This not only provides uniformity to the framework, but

also allows us to shift our attention from raw data to the space of measurements that characterizes it.

Formally, Bergomi et al. [28] quantify the distance between $x_1, x_2 \in X$ by comparing the values of x_1 and x_2 in the set of admissible functions Φ with a pseudo-metric

$$D_X(x_1, x_2) = \sup_{\varphi \in \Phi} |\varphi(x_1) - \varphi(x_2)|.$$

A pseudo-metric is a distance d without the property that $d(a,b) = 0 \implies a = b$. In particular, D_X defines that two points are distinguishable if and only if they assume different values for some admissible function φ. By endowing the space X with the topology induced by the pseudo-metric D_X, we formalize the attention shift from X to Φ.

3.3 Transforming Data

Now that we have established how to represent the input data, let us introduce how the framework defines prior knowledge. Topological spaces X are transformed by the means of maps $g\colon X \to X$ that are Φ-preserving homeomorphisms w.r.t. D_X. This means that the set of admissible measurements Φ on X remains topologically the same when transformed. Thus, the set $Homeo_\Phi(X)$ denotes the set of Φ-preserving homeomorphisms.

Let us consider a group $G \subseteq Homeo_\Phi(X)$ that represents the set of transformations on data for which we require equivariance to be respected. We can define a pseudo-distance D_G that models the distance between two homeomorphisms on G as the difference of their actions on Φ:

$$\begin{aligned} &D_G\colon G \times G \to \mathbb{R} \\ &D_G(g_1, g_2) = \sup_{\varphi \in \Phi} D_\Phi(\varphi \circ g_1, \varphi \circ g_2) \\ &\text{with } D_\Phi(\varphi_1, \varphi_2) = \|\varphi_1 - \varphi_2\|_\infty \end{aligned}$$

That is, D_G provides a way to compare two transformations by analyzing the effects of their actions on the set Φ of possible measurements.

Lastly, Bergomi et al. [28] define the natural pseudo-distance d_G on the space Φ.

Definition 1. *The natural pseudo-distance $d_G\colon \Phi \times \Phi \to \mathbb{R}$ associated with the group G acting on Φ is defined by the setting:*

$$d_G(\varphi_1, \varphi_2) = \inf_{g \in G} D_\Phi(\varphi_1, \varphi_2 \circ g)$$

d_G represents the ground truth in the model, it compares functions and vanishes for pairs of functions that are equivalent w.r.t. the action of our group of homeomorphism G, which expresses the equivalences between data. In other words, d_G compares the admissible functions in Φ by providing a measure of how topologically similar a pair of functions are with respect to the action group of G. For instance, let us take into

account two possible measurements $\varphi_1, \varphi_2 \in \Phi$ of our data, if they are supposed to be equivalent when acted on by transformations in G, meaning that such measures are equivariant with respect to G, then $d_G(\varphi_1, \varphi_2) = 0$. Alas, d_G is very difficult to compute, so the authors approximate this pseudo-distance using group equivariant non-expansive operators (GENEOs) and persistent homology.

3.4 Group Equivariant Non-Expansive Operators (GENEOs)

Following the results of the previous sections, we now formally define the concept of GENEOs. First, let us consider the notion of a perception pair (Φ, G): it is composed of all admissible measurements from our data to real values Φ and a set of transformations G that preserve homeomorphism on Φ.

Assuming that (Φ, G), (Ψ, H) are perception pairs and that $T : G \to H$ is a homomorphism, that is, a map between two structures of the same type that preserves their operations. Formally, $f : A \to B$ preserves the operation μ defined on both A and B if $f(\mu_A(a_1, \ldots, a_k)) = \mu_B(f(a_1), \ldots, f(a_k))\ \forall a_1, \ldots, a_k \in A$. This means that T allows us to convert the set of transformations G into H, while preserving the effects of their operations on the respective sets of measurements. For instance, imagine that a transformation $g \in G$ on $\varphi \in \Phi$ causes a simple translation. A homomorphism $T : G \to H$ let us achieve an equivalent transformation to g for a topological space Ψ. Thus, the transformation $T(g)$ applied to $\psi \in \Psi$ produces a corresponding translation.

Definition 2 (Group Equivariant Non-Expansive Operator (GENEO))**.** *Consider two perception pairs (Φ, G) and (Ψ, H) and a homomorphism $T : G \to H$. A map $F : \Phi \to \Psi$ is a group equivariant non-expansive operator if it exhibits equivariance*

$$\forall \varphi \in \Phi, \forall g \in G, F(\varphi \circ g) = F(\varphi) \circ T(g), \tag{3.1}$$

and is non-expansive

$$\forall \varphi_1, \varphi_2 \in \Phi, D_\Psi(F(\varphi_1), F(\varphi_2)) \leq D_\Phi(\varphi_1, \varphi_2). \tag{3.2}$$

Bergomi et al. [28] add to this result with Proposition 1, which explains how the ground truth d_G is approximated using GENEOs. The interested reader can find the proof of this proposition in [28].

Proposition 1. *If F is a GENEO from (Φ, G) to (Ψ, H) associated with $T : G \to H$, then it is a contraction with respect to the natural pseudo-distances d_G, d_H.*

To put it differently, a GENEO can transform data into higher-level representations while preserving its structure, and, by extension, its properties and relations. The homomorphism T between the sets of homeomorphic transformations guarantees that equivariance is maintained when transforming Φ into Ψ. For example, by defining a GENEO that is equivariant with respect to planar translations, which is a property present in the

TS40K dataset (a tower is still a tower no matter its coordinates), we can create a higher representation of the data that is still equivariant to said transformations.

Non-expansivity and convexity are essential properties for the applicability of GENEOs in a Machine Learning setting. When the spaces Φ and Ψ are compact, non-expansivity guarantees that the space of all GENEOs $\mathfrak{F}$ is compact as well. Compactness ensures that any operator can be approximated by a finite set of operators sampled in the same space. Moreover, by assuming that Ψ is convex, Bergomi et al. prove that $\mathfrak{F}$ is also convex. Convexity guarantees that new GENEOs can be obtained through the convex combination of preexisting GENEOs. Therefore, these results prove that any GENEO can be efficiently approximated by a certain number of other GENEOs in the same space.

3.5 Overview and MNIST Case Study

To summarize, Bergomi et al. [28] and Cascarano et al. [29] contribute with a formal mathematical framework for Machine Learning, based on the study of metric and topological properties of operators acting on function spaces, which allow us to formally define an agent (e.g., a neural network) as a collection of operators.

Instead of focusing on raw data, their approach works with a set of admissible measurements that can describe data through function spaces. Then, they search for suitable operators that show equivariance w.r.t. specific transformations, in order to embed an agent with background knowledge. Let us consider an example in a Deep Learning context, imagine that we wish to classify images of different flowers. This problem is equivariant to rotations and reflections for example, i.e., such transformations do not alter the contextual meaning of a flower image. Therefore, we can take advantage of this by embedding a neural network with GENEOs equivariant w.r.t rotations and reflections, meaning that higher representations of the initial images in the network will maintain these properties.

In order to assess their contributions, the authors developed a CNN embedded with GENEO-kernels in the convolutional layers, to be compared against a conventional CNN with identical structure in the MNIST, fashion-MNIST, and CIFAR10 datasets. The developed GENEOs presented equivariance w.r.t. isometries, such as rotations. The original data was augmented with translations, reflections and rotations to properly study the impact of adding the mentioned GENEOs when compared to a random kernel initialization in the traditional CNN. Contrarily to the typical kernels, GENEO-kernels are not tuned during training, since that would disrupt the sought-out equivariance. Note that the compared networks have identical structures, the only aspect that differs is the kernel initialization and training, which is the intended subject of study in the experiment. Results show that the GENEO-model outperforms the conventional CNN in all datasets, consistently demonstrating high accuracy and low loss, whereas the latter shows an irregular performance, with average accuracies and high losses.

A crucial step of this experiment is the selection and sampling of operators that will be employed in the model. To this end, Bergomi et al. [28] developed an algorithm that, first, elects a subset of operators in a certain GENEOs space, capable of giving meaningful representations of the data with respect to their labeling. Following the described experiments, they choose to work with a parametric family of GENEOs composed of Gaussian mixtures, seeing as they provide equivariance w.r.t. isometries. From this family, an operator is elected if it is capable of discriminating elements with the same label under a certain threshold, which ensures that the chosen GENEOs provide relevant representations of the input data according to their classes. Secondly, they take advantage of defined pseudo-distances to compare the space of two GENEOs, allowing them to eliminate redundant operators if their spaces are too closely related. With such a system in place, the authors guarantee maximal diversity of the sampled operators when evaluated within and in between classes.

This approach to operator selection and sampling resembles Machine Learning strategies, in a sense that requires human evaluation of certain parameters in order to elect the thought to be the most adequate GENEOs in the current context. However, like most ML models, there is no guarantee that the chosen collection of GENEOs is better than all other combinations. Therefore, Cascarano et al. [29] contribute to this framework by providing a way to endow a GENEOs space $\mathfrak{F}$ with the structure of a Riemannian manifold, making available the use of gradient descent methods for the minimization of a cost function on $\mathfrak{F}$ [29]. In practical terms, this implies that it is possible to develop a neural network that learns the best type of operators to employ in order to minimize its loss while preserving the equivariance w.r.t. the pertinent transformations.

3.6 GENEOnet: an Application of GENEOs to Protein Pocket Detection

Bocchi et al. [61] innovatively inject knowledge into a learning model using GENEOs to detect protein pockets. Drug design and development are now taking advantage of computational approaches to increase the speed of drug discovery. Specifically, structure-based drug design technologies can predict the binding affinity of novel compounds, which allows for rapid and effective simulation of protein interaction at an atomic level. In order to evaluate the drug binding sites, algorithms analyze the geometrical structure of proteins to detect empty regions (i.e., pockets) and perform a physicochemical analysis to rank the found pockets in terms of their binding affinity.

In their work, Bocchi et al. [61] demonstrate how to take advantage of the GENEO-framework to develop an interpretable learning agent. First, they represent the original data, i.e., 3D point clouds of the atomic structure of proteins, by the means of admissible measurements based on the geometrical, physical, and chemical properties of proteins. Next, they process these input channels with a GENEO-layer, where each channel φ_i is

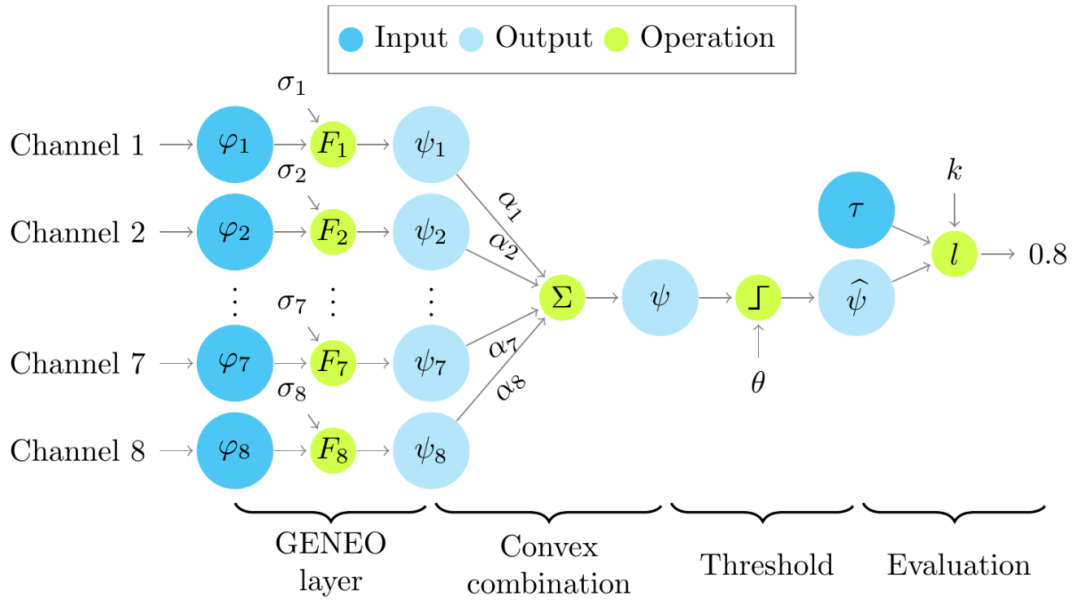


Figure 3.1: GENEOnet model workflow [61]. The input channels $\varphi_1,\ldots,\varphi_8$ are fed to the respective GENEOs $F_1,\ldots,F_8$ defined by parameters $\sigma_1,\ldots,\sigma_8$. The observation of the GENEO-layer results in the outputs $\psi_1,\ldots,\psi_8$, which are then combined through convex combination with weights $\alpha_1,\ldots,\alpha_8$ into an overall analysis ψ. Lastly, a threshold θ is applied, resulting in the classification $\hat{\psi}$, which is compared to the ground truth τ.

processed by a respective GENEO F_i defined by a single shape parameter σ_i. The resulting observations ψ_i are combined through convex combination with weights $\alpha_1,\ldots,\alpha_n$ into an overall analysis ψ of the properties embedded in the defined GENEOs. Then, they define a thresholding operation to classify the predicted protein pockets and reach a binding score, which is then compared with the ground truth during training.

Experimental results show that this approach surpasses state-of-the-art methods for pocket detection. The weights $\alpha_1,\ldots,\alpha_n$ represent how important each of the input channels and respective GENEO is to protein pocket binding evaluation. The shape parameters $\sigma_1,\ldots,\sigma_n$ define an instance of each GENEO F_i. These operators encode the background knowledge that experts deem useful to detect pockets. During training, it is these parameters along with the convex weights that are fine-tuned with backpropagation.

In comparison, our model performs 3D semantic segmentation instead of an overall evaluation of 3D point clouds. We developed GENEOs that are composed of more than one shape parameter, which corroborates the general application of GENEO-based models. Bocchi et al. [61] take advantage of several measurements on protein structure in their proposal, we show that even with a single measurement available, GENEO-based models are able to achieve good performance. Lastly, we demonstrate that GENEO-powered models cope with a severely imbalanced dataset and, contrary to classical models, are not affected by noisy labeling.

4

Proposed Method: SCENE-Net for Interpretable Scene Understanding

In this chapter, we introduce the overall architecture of SCENE-Net. First, we discuss the basis for the development of GENEO-kernels. Secondly, we explain the measurement function used to describe 3D point clouds. Then, we describe in detail the workflow of SCENE-Net as well as its components. Lastly, we detail the loss function used to train the observer.

4.1 The Convolution Operation

We decided to use the convolution operation as the basis for our GENEO operators. Not only is this operation widely studied in Machine Learning and signal processing, but all previous work on GENEOs [28, 29, 61] uses this operation as a foundation for GENEO-kernels. Simply put, convolution is an operation between two functions (f and ψ) that outputs a third one ($f * \psi$) expressing how the shape of f reacts to ψ, formally, in a one-dimensional setting,

$$(f * \psi)(t) = \int_{-\infty}^{\infty} f(\tau)\psi(t-\tau)\,d\tau.$$

In other words, the output of a convolution is a function analogous to f that expresses how the space of ψ is present within f. Therefore, by definition, convolution provides equivariance with respect to planar translations, meaning that patterns in f can be detected regardless of their location. This result grants CNNs great performance in areas such as image classification and speech recognition, leading to a wide variety of research around this operator. In our context, we wish to endow the convolution operation with additional equivariance w.r.t. specific groups of transformations G that express some property of the input data, such as equivariance w.r.t. rotations around the z-axis.

Cohen and Welling [62] follow this objective when introducing group equivariant convolutional networks (G-CNNs) [62], a generalization of conventional CNNs that extend their equivariance to G-transformations. G-CNNs employ G-convolutions, a new

layer type that presents a higher degree of weight sharing and significantly increases the expressive power of networks. The authors also propose an implementation of pooling and non-linear transformations that preserve the equivariance of previous operations, this way, several layers can be freely stacked into deep neural networks. This architecture was tested with two different groups of transformations, $p4$ and $p4m$, against conventional CNNs (denominated $\mathbb{Z}2$-convolutions) on the MNIST and CIFAR10 datasets. The $p4$ group consists of all compositions of translations and rotations by 90 degrees in any rotation center in a square grid, whereas $p4m$ generalizes the $p4$ group by also considering reflections in the compositions. The obtained results show that $p4$ and $p4m$ models outperform the $\mathbb{Z}2$-CNN without increasing the number of parameters, and the $p4m$-CNN outperformed all published results on the CIFAR10 dataset at the time.

The work of Cohen and Welling [62] provides us essential insight: First, they reinforce that convolution is the operator of choice when seeking equivariance w.r.t. specific G-transformations in a Deep Learning context. Second, crucial operations in convolutional networks, such as pooling, can be computed in such a way that preserve the equivariance achieved in previous layers. Third, their results demonstrate the efficacy of group-equivariance when applied to popular datasets in the field, which is further consolidated by the GENEO framework [28].

4.2 TS40K Data Representation

4.2.1 Voxelization

Even though the convolution operation provides many advantages in our context, it also requires some compromises for its application on 3D point clouds. Standard convolution operators need structured 3D data in order to apply kernel functions with the same number of dimensions, this entails that raw point clouds are unfit to be convolved due to their lack of structure. As discussed in Section 2.1.2, state-of-the-art methods employ voxel-based or multi-view-based techniques, transforming the point cloud into a voxel grid or a series of 2D representations, respectively. For 3D semantic segmentation, voxelization is the method of choice, since it preserves the three dimensions of the original data and mitigates the information loss from the data discretization better than multi-view approaches (this is explained in detail in Section 2.1.2.2). Moreover, it is paramount that we preserve the geometrical properties and relations of the 3D scene in order to maintain the symmetries of the original space and, this way, take full advantage of the GENEO theory.

Therefore, TS40K point clouds presented in Section 5.1 will be subject to voxelization at the beginning of our framework, which makes SCENE-Net a voxel-based method. The biggest limitation of voxel-based approaches is their high computational cost and memory footprint, which bounds the resolution of voxel grids to usually 64^3. This is in consequence of state-of-the-art methods being complex and composed of numerous layers, which leads to an exponential increase in training time with higher resolutions.

Since SCENE-Net has a simple architecture, we can effectively employ higher resolutions. In fact, SCENE-Net can be trained with lower resolutions, such as 64^3, and be applied to higher resolution voxel grids, with shapes of 128^3 and 256^3 (this is further explained in Section 5.2.7).

4.2.2 Measurement Function

In order to utilize GENEOs on voxelized point clouds, we require a function φ that operates on the voxel grid space and provides a real-valued measurement of the 3D data (as introduced in Section 3.2). An appropriate measurement should preserve the geometry of the 3D scene and facilitate the detection of patterns of interest when convolving the voxel grid. Thus, φ will assign to each voxel a real value that offers an appropriate representation of their respective subset of points from the original point cloud. Specifically, we define $\varphi\colon \mathbb{R}^3 \to \{0,1\}$ as a geometrical function that signals the presence of 3D points in a voxel. That is, if a voxel contains any 3D point, φ outputs one, otherwise, the result is zero to represent empty voxels. Seeing as there are few points classified as a tower, this measurement takes full advantage of the raw data to emphasize the geometry of supporting towers. This representation also helps to mitigate the data imbalance in TS40K, since the number of points of each class is not relevant to the definition of φ. What matters is the space that tower points occupy in a voxel grid.

4.3 Overview

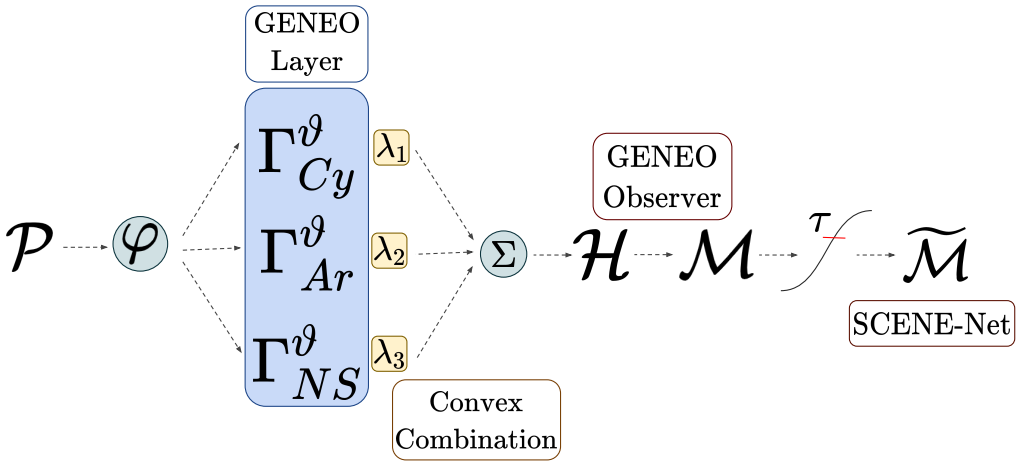


Figure 4.1: Pipeline of SCENE-Net: an input point cloud $\mathcal{P}$ is measured according to function φ and voxelized. This representation then is fed to a GENEO-layer, where each operator Γ_i^{ϑ} separately convolve the input. A GENEO observer $\mathcal{H}$ is then achieved by convex combination of the operators in the GENEO-layer. $\mathcal{M}$ transforms the analysis of the observer into a probability of belonging to a tower. Lastly, a threshold operation is applied to classify the voxels. Note that this final step occurs after training is completed.

3D Point clouds are generally denoted as $\mathcal{P} \in \mathbb{R}^{N\times(3+d)}$, where N is the number of points and $3+d$ is the cardinality of spatial coordinates plus any point-wise features, such as colors or normal vectors. The input point cloud is first transformed in accordance to a measurement function φ that signals the presence of 3D points in a voxel discretization. Next, the transformed input is fed to a layer of multiple GENEOs (GENEO-layer), each chosen randomly from a parametric family of operators, and defined by a set of trainable shape parameters ϑ_i (Figure 4.1). Such GENEOs are in the form of convolutional operators with carefully designed kernels as described later. During training, it is not the kernels themselves that are fine-tuned with backpropagation, since this would not preserve equivariance at each optimization step. Instead the error is propagated to the shape parameters ϑ_i of each operator and their convex coefficients λ_i. Following the GENEO-layer, the set of operators $\Gamma = \{\Gamma^{\vartheta}\}$, with shape parameters $\vartheta = \vartheta_1, \ldots, \vartheta_n$, are combined through convex combination with weights $\lambda = (\lambda_1, \ldots, \lambda_n)^T$ as follows:

$$\begin{aligned} &\underset{\lambda,\vartheta}{\mathscr{H}} : \mathcal{P} \to \mathcal{P} \\ &\underset{\lambda,\vartheta}{\mathscr{H}}(x) = \lambda^T \, \Gamma^{\vartheta}(\varphi)(x). \end{aligned} \tag{4.1}$$

Since the convex combination of GENEOs is also a GENEO [28], $\mathscr{H}$ preserves the equivariance of each operator $\Gamma^{\vartheta} \in \Gamma$. In fact, $\mathscr{H}$ defines a GENEO observer that analyzes the 3D input scenes looking for the geometrical properties encoded in Γ. The convex coefficients λ represent the overall contribution of each operator Γ_i^{ϑ} to $\mathscr{H}$ to the analysis. The geometrical parameters λ, ϑ grant our model its intrinsic interpretability. They are learned during training and represent geometric properties and the importance of each Γ^{ϑ} in modeling the ground truth.

Next, we transform the analysis of the observer into a probability of each 3D voxel belonging to a supporting tower as a model

$$\begin{aligned} &\underset{\lambda,\vartheta}{\mathscr{M}} : \mathcal{P} \to [0,1]^N \\ &\underset{\lambda,\vartheta}{\mathscr{M}}(x) = \left(\tanh\left(\underset{\lambda,\vartheta}{\mathscr{H}}(x) \right) \right)_+, \end{aligned}$$

where $(t)_+ = \max\{0, t\}$ is the rectified linear unit (ReLU). Negative signals in $\mathscr{H}(x)$ represent patterns that do not exhibit the sought-out geometrical properties. Conversely, positive values quantify their presence. Therefore, tanh compresses the value distribution of the observer into [-1, 1], and the ReLU is then applied to enforce a zero probability to negative signals. Lastly, a probability threshold $\tau \in [0,1]$ is defined through hyperparameter fine-tuning and applied to $\mathscr{M}$ resulting in a map

$$\begin{aligned} &\underset{\lambda,\vartheta}{\widetilde{\mathscr{M}}} : \mathcal{P} \times \mathbb{R} \to \{0,1\}^N \\ &\underset{\lambda,\vartheta}{\widetilde{\mathscr{M}}}(x,\tau) = \left\{ \underset{\lambda,\vartheta}{\mathscr{M}}(x) \right\} \geq \tau, \end{aligned}$$

where $\widetilde{\mathscr{M}}$ represents the complete definition of **SCENE-Net**.

4.4 Knowledge Engineering via GENEOs

In this section, we formally define the knowledge embedded in the observer $\mathscr{H}$. The following GENEOs describe power line supporting towers in order to fully discriminate them from their environment.

4.4.1 Cylinder GENEO.

The most striking characteristic of supporting towers against the rural environment is their long, vertical and narrow structure. As such, their identification is equivariant w.r.t. rotations along the *z-axis* and translations in the xy plane, which we encode by the means of a cylinder

$$f_{Cy}\colon \mathbb{R}^3 \to \{0,1\}$$
$$f_{Cy}(x) = \begin{cases} 1 & \text{if } \|\pi_{-3}(x) - \pi_{-3}(c)\|^2 = r^2 \\ 0 & \text{if otherwise} \end{cases},$$

where $\pi_{-3}(x) = [\pi_1(x), \pi_2(x), 0]$ nullifies the third coordinate of a three-element vector and π_i defines a projection function of the ith element of the input vector. However, in data patterns show smoothing, so we relax this condition by defining

$$g_{Cy}\colon \mathbb{R}^3 \to [0,1]$$
$$g_{Cy}(x) = e^{\frac{-1}{2\sigma^2}(\|\pi_{-3}(x)-\pi_{-3}(c)\|^2 - r^2)^2}.$$

Function g_{Cy} defines a hollow cylinder centered in c by the means of a Gaussian function, with the distance between x and the cylinder radius (r) as its mean. The shape parameters of the Cylinder are the standard deviation of the Gaussian and radius r and are defined by $\vartheta_{Cy} = [r, \sigma]$.

GENEOs act on functions, transforming them in a way that remain equivariant to a specific group of transformations. Our GENEOs act upon Φ, the topological space representing $\mathcal{P}$ with admissible functions $\varphi\colon \mathbb{R}^3 \to \{0,1\}$. Specifically, we work with appropriate $\varphi \in \Phi$ functions that represent point clouds and preserve their geometry. For instance, φ can be a function that signals the presence of 3D points in a voxel grid. Therefore, the cylinder GENEO Γ_{Cy} transforms φ into a new function that detects sections in the input point cloud that demonstrate the properties of g_{Cy} and, simultaneously, preserves the geometry of the 3D scene

$$\Gamma^{\vartheta}_{Cy}\colon \Phi \to \Psi, \qquad \psi_{Cy} = \Gamma^{\vartheta}_{Cy}(\varphi)$$
$$\psi_{Cy}(x) = \int_{\mathbb{R}^3} \tilde{g}_{Cy}(y)\varphi(x-y)dy$$

where Ψ is a new topological space that represents $\mathcal{P}$ with functions $\psi\colon \mathbb{R}^3 \to [0,1]$ and $\tilde{g}_{Cy}$ defines a normalized Cylinder. The kernel g_{Cy} is normalized to have a zero-sum

in order to promote stability of the observer. This way, we encourage the geometrical properties that exhibit the sought-out group of transformations and punish those which do not. Thus, $\psi_{Cy}(x)$ assumes positive values for 3D points near the radius, whereas negative values discourage shapes that do not fall under the g_{Cy} definition. This leads to a more precise detection of the encoded group of transformations. The cylinder kernel

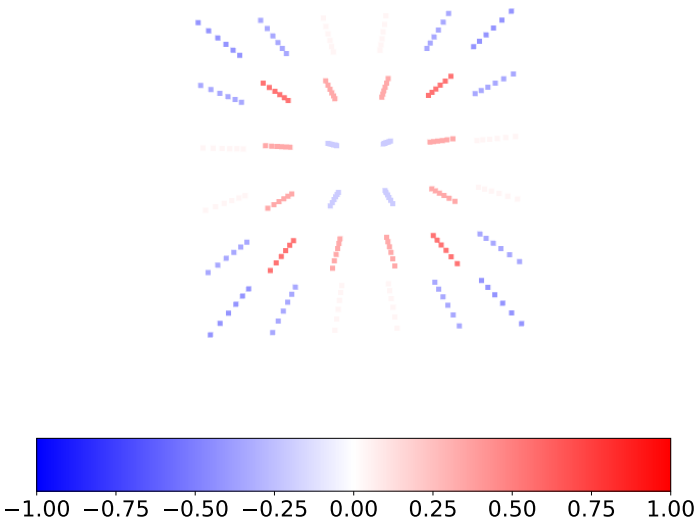


Figure 4.2: Cylinder kernel discretized in a voxel grid and colored according to weight distribution.

discretized in a voxel grid can be seen in Figure 4.2.

4.4.2 Arrow GENEO.

Towers are not the only element in rural environments characterized by a vertical narrow structure. The identification of trees also shows equivariance w.r.t. rotations along the *z-axis*. Therefore, it is not enough to detect the body of towers, we also require the power lines that they support. To this end, we define a cylinder following the rationale behind the Cylinder GENEO with a cone on top of it. This arrow defines equivariance w.r.t. the different angles that power lines may find their supporting tower. Formally, the Arrow function can be defined as

$$f_{Ar}\colon \mathbb{R}^3 \to \{0,1\}$$

$$f_{Ar}(x) = \begin{cases} 1 & \text{if } \|\pi_{-3}(x) - \pi_{-3}(c)\|^2 = r^2 \\ & \wedge \pi_3(x) < h \\ 1 & \text{if } \|\pi_{-3}(x) - \pi_{-3}(c)\|^2 = r_c^2 \sin\theta \\ & \wedge \pi_3(x) \geq h \\ 0 & \text{if otherwise} \end{cases},$$

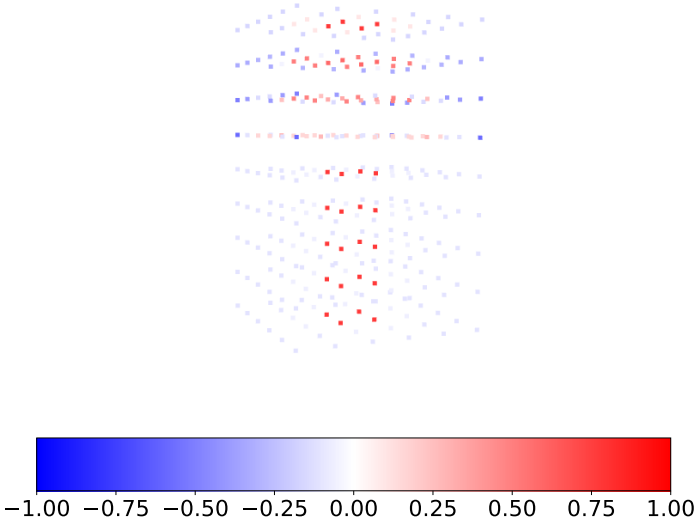


Figure 4.3: Arrow kernel discretized in a voxel grid and colored according to weight distribution.

with $\theta = \beta \frac{\pi}{2+\pi_3(x)-h}$ and $\beta \in [0,1]$ defining the inclination of the cone. The radii of the cylinder and the cone are defined by r and r_c, respectively, with c as their center. Lastly, h defines the height at which the arrow is placed on top of the cylinder. This definition is too strict to yield any feasible results in real-world scenarios, so we smooth the conditions as follows

$$g_{Ar}\colon \mathbb{R}^3 \to [0,1]$$
$$g_{Ar}(x) = \begin{cases} e^{\frac{-1}{2\sigma^2}(\|\pi_{-3}(x)-\pi_{-3}(c)\|^2-r^2)^2} & \text{if } \pi_3(x) < h \\ e^{\frac{-1}{2\sigma^2}(\|\pi_{-3}(x)-\pi_{-3}(c)\|^2-r_c^2\sin\theta)^2} & \text{if otherwise} \end{cases}.$$

Thus, the shape parameters of the Arrow are defined by the vector $\vartheta_{Ar} = [r, \sigma, h, r_c, \beta]$. We are also interested that this kernel sums to zero, so we define the GENEO Γ_{Ar} as

$$\Gamma^{\vartheta}_{Ar}\colon \Phi \to \Psi, \qquad \psi_{Ar} = \Gamma^{\vartheta}_{Ar}(\varphi)$$
$$\psi_{Ar}(x) = \int_{\mathbb{R}^3} \tilde{g}_{Ar}(y)\varphi(x-y)dy,$$

where $\tilde{g}_{Ar}(y)$ represents a normalized Arrow kernel. Its discretization is depicted in Figure 4.3.

4.4.3 Negative Sphere GENEO.

Detecting power lines does not exclude the remaining objects in the scene whose identification also demonstrates equivariance w.r.t. rotations along the *z-axis*. Arboreal elements, such as bushes, are especially frequent in the TS40K dataset. Thus, we designed a negative sphere to diminish their detection and, simultaneously, punish the geometry of trees:

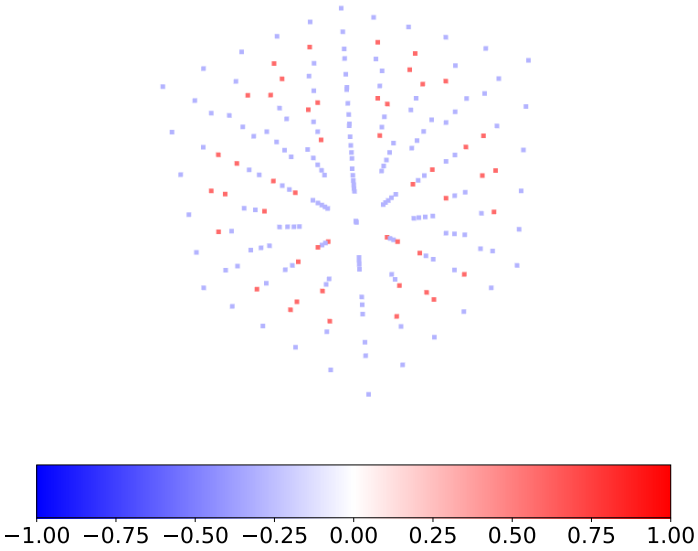


Figure 4.4: Negative sphere kernel discretized in a voxel grid and colored according to weight distribution.

$$f_{NS}\colon \mathbb{R}^3 \to \{-\omega, 1\}$$
$$f_{NS}(x) = \begin{cases} 1 & \text{if } \|x-c\|^2 = r^2 \\ -\omega & \text{if otherwise} \end{cases},$$

where $\omega \in]0,1]$ defined a small negative weight that punishes the spherical shape. Next, we proceed with the relaxation of the sphere definition with function g_{NS} as

$$g_{NS}\colon \mathbb{R}^3 \to [-\omega, 1[$$
$$g_{NS}(x) = e^{\frac{-1}{2\sigma^2}(\|x-c\|^2 - r^2)^2} - \omega,$$

with shape parameters defined as $\vartheta_{NS} = [r, \sigma, \omega]$.

Since we wish to discourage spherical patterns following the definition of g_{NS}, we do not enforce that its space sums to zero, obtaining

$$\Gamma^{\vartheta}_{NS}\colon \Phi \to \Psi_{NS}, \qquad \psi_{NS} = \Gamma^{\vartheta}_{NS}(\varphi)$$
$$\psi_{NS}(x) = \int_{\mathbb{R}^3} g_{NS}(y)\varphi(x-y)dy,$$

where Ψ_{NS} is a topological space containing functions $\psi\colon \mathbb{R}^3 \to [-\omega, 1[$. Figure 4.4 depicts the computation of this kernel in a voxel grid.

4.5 GENEO Loss

The use of GENEOs in knowledge embedding forces our model to uphold convexity of the observer during training. Thus, our problem statement is represented by the following

optimization problem

$$\begin{aligned} \underset{\lambda,\vartheta}{\text{minimize}} \quad & \mathop{\mathbb{E}}_{X,y,\alpha,\epsilon}\left\{\mathscr{L}_{seg}(\lambda,\vartheta)\right\} \\ \text{s.t.} \quad & \vartheta \geq 0, \\ & \lambda^T \mathbf{1} = 1, \\ & \lambda \geq 0, \end{aligned}$$

where the segmentation loss $\mathscr{L}_{seg}$ is defined as

$$\mathscr{L}_{seg}(\lambda,\vartheta) = \frac{1}{\|X\|}\sum f_w(\alpha,\epsilon,y)\Big(\underset{\lambda,\vartheta}{\mathscr{M}}(X) - y\Big)^2.$$

Here $\|X\|$ denotes the number of samples in X and $\mathscr{M}(X)$ is the likelihood predicted by SCENE-Net that voxels in X are towers. The loss uses a weighted squared error following the weighting scheme f_w proposed in [63] to mitigate data imbalance. The hyperparameter α emphasizes the weighting scheme, whereas ϵ is a small positive number that ensures positive weights. Thus, $\mathbb{E}$ represents the expectation of the segmentation loss over the data distribution. The above constraints ensure that our model $\mathscr{M}$ maintains convexity throughout training, with $\mathbf{1}$ denoting a vector of ones. Next, we performed a simple redefinition of the variables λ to obtain an equivalent optimization problem, as $\lambda_n = 1 - \sum_{i=1}^{N-1}\lambda_i$, thus obtaining Problem (4.2),

$$\begin{aligned} \underset{\lambda,\vartheta}{\text{minimize}} \quad & \mathop{\mathbb{E}}_{X,y,\alpha,\epsilon}\left\{\mathscr{L}_{seg}(\lambda,\vartheta)\right\} \\ \text{s.t.} \quad & \vartheta \geq 0 \\ & \lambda \geq 0 \end{aligned} \quad . \tag{4.2}$$

Then, we ensure non-negativity of the trainable parameters λ, ϑ by relaxing Problem (4.2) and introducing a penalty in the optimization cost definition as

$$\underset{\lambda,\vartheta}{\text{minimize}} \quad \mathop{\mathbb{E}}_{X,y,\alpha,\epsilon}\left\{\mathscr{L}_{seg}(\lambda,\vartheta)\right\} + \rho_l\Big(\sum_i^N h(\lambda_i)\Big) + \rho_t\Big(\sum_i^N h(\vartheta_i)\Big), \tag{4.3}$$

where $h(x) = \Big(-x\Big)_+$, ρ_l and ρ_t are scaling factors of the negativity penalty h illustrated in 4.5. GENEO final loss optimization is formalized in Problem (4.3). It consists of a data fidelity component (i.e., $\mathscr{L}_{seg}$), and two penalties to ensure non-negative parameters.

4.5.1 Density Based Weighting Scheme for Data Imbalance in Regression

The voxelization of point clouds worsens the data imbalance already present in the TS40K dataset. Since 3D scenes are now organized in a voxel grid, the ground truth encompasses more volume than the original raw data, which leads to a lot of empty voxels (i.e., with probability zero) and only a fair few voxels with tower points. In regression problems,

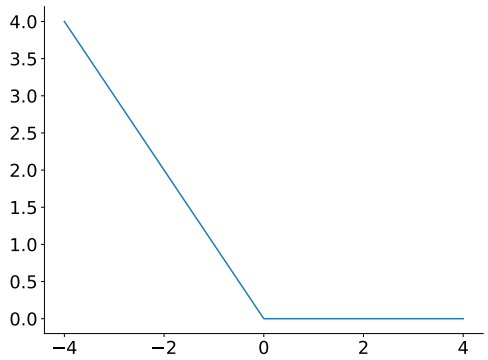


Figure 4.5: Graphic representation of the negative penalty h.

the standard loss metric is Mean Squared Error (MSE), transmitting how far, on average, each prediction $\hat{y}_i$ is from its target value y_i

$$MSE(\hat{y}, y) = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

where N is the number of samples. In order to mitigate this data imbalance, we incorporate a weighting function f_w on the MSE that weights data points according to the density of their target values. That is, each sample will be assigned an importance measure that is inversely proportional to the probability of the occurrence of its target value. This way, a voxel with a target value of zero will be assigned a small weight since the density of the value zero dominates the target values. Conversely, a voxel with a target value of one will have a bigger weight because there are very few voxels with such a target.

This idea follows the work of Steininger et al. [63], they define a weighting scheme w and a function f_w to enforce it. Specifically, the squared error between the prediction of the model and the ground truth y is weighted inversely to the value density of y:

$$\begin{aligned}
&\underset{\alpha,\epsilon}{\mathscr{L}}(\hat{y}, y) = \frac{1}{N}\sum f_w(\alpha, \epsilon, y) MSE(\hat{y}, y) \\
&f_w(\alpha, \epsilon, y) = \frac{f''_w(\alpha, \epsilon, y)}{\frac{1}{N}\sum f''_w(\alpha, \epsilon, y)} \\
&\text{with, } \; f''_w(\alpha, \epsilon, y) = \max(1 - \alpha p(y), \epsilon),
\end{aligned}$$

where $\alpha \in [0, \infty[$ emphasizes the weighing scheme w, ϵ is a small positive number that ensures positivity for all weights, and p is a normalized density estimator of the target values y. As such, f_w offers the following properties:

- Samples with common target values are assigned smaller weights than rarer samples;
- f_w yields uniform weights for $\alpha = 0$;

- Data points are never weighted negatively (seeing as models would try to maximize the difference between the prediction and their true value during training);
- No weight should be 0 to avoid models ignoring parts of the dataset;
- The mean weight over all data points is 1; Since α scales all gradients without normalization, it directly scales the learning rate of the model. By ensuring a mean weight of 1, different α values do not change the average magnitude of the gradient.

Steininger et al. [63] employ kernel density estimation (KDE) to approximate the density of target values (function p). In our case, seeing as we assume a probability of one for tower voxels and zero for empty voxels, we simply calculate the density distribution of both values in the voxelized data.

5

Experiments

In this chapter, we introduce the TS40K dataset used in our experiments. Next, we assess the properties of our model SCENE-Net that help electrical companies in the inspection of power lines: (1) interpretability of the model, (2) accuracy, (3) robustness to noisy labels, (4) training and inference time, and (5) inference performance with high resolution, when trained with low-resolution voxel grids. We benchmark our model with a traditional CNN with similar architecture. Then, we detail several performed experiments that were paramount for us to reach the final definition of SCENE-Net.

5.1 TS40K Dataset

5.1.1 Data Description

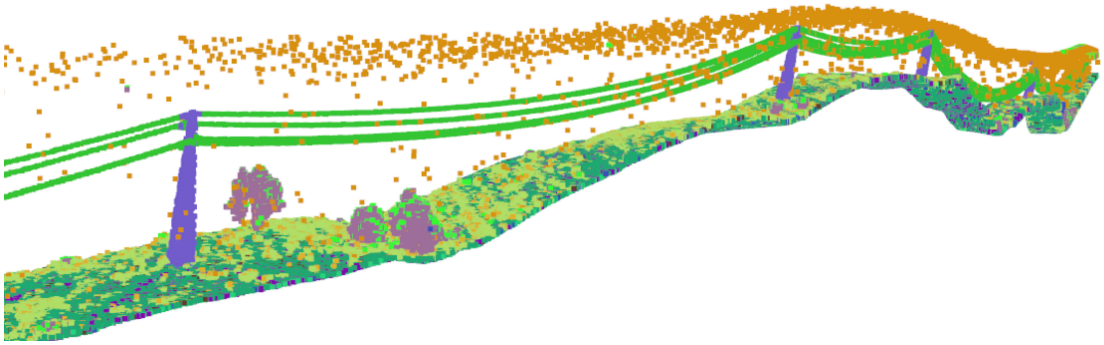


Figure 5.1: Visualization of TS40K raw point cloud with colored labels.

Electrical companies are responsible for the maintenance and inspection of the transmission system. They deploy low-flying helicopters to scan rural environments, from a BEV perspective, where the electrical grid is located. The produced point clouds exhibit different data properties when compared to 3D scenes captured from other viewpoints, such as from a vehicle. Namely, they show high point density and no object occlusion, scene elements present homogeneous density and no sparsity.

- *No Occlusion*: Most of the available datasets in 3D point clouds are built from sensors integrated into day-to-day devices in order to emulate real-life conditions. For instance, the majority of benchmark datasets [8, 10, 11, 44] are acquired from vehicles to simulate an autonomous driving environment. Consequently, the equipped 3D sensor cannot capture the entire 3D scene around it since objects may occlude others, which hinders the performance of employed Deep Learning methods. Fortunately, EDP point clouds do not present any object occlusion since they are captured from a BEV perspective;
- *Homogeneous Object Density*: The density of each element in the point cloud does not fluctuate, meaning that the number of points that compose a particular object is consistent throughout its entire structure. Still, different elements of a scene present different densities, for instance, the ground has a higher density of points than the power lines;
- *High Point Density*: In addition to density homogeneity, elements in the 3D environment generally encase a high density of points describing its structure.

Then, the acquired 3D data is processed by maintenance personnel. Specifically, seeing as the raw point clouds are quite verbose and mainly encompass campestral areas, data is sectioned into strips of land focused on the transmission system as shown in Figure 5.1. The raw data is composed of several LiDAR files containing roughly 40 000 kilometers of the above land strips. 3D points therein are labeled with one out of 22 possible classes, such as power lines and their supporting towers, low and medium vegetation, rivers, railroads, human-made structures that do not belong to the transmission network, the ground, optic cables, among others. Table 5.1 depicts these classes and their density in the dataset. Rail lines and road surfaces constitute the majority of the dataset (63%), whereas classes of interest, such as power lines, make up less than 5% of the overall data. In particular, less than 1% of 3D points belong to supporting towers, and among these roughly 50% are mislabeled patches of ground. The maintenance personnel normally labels all 3D points near supporting towers as so to optimize inspection time and guarantee that any high-risk situations are detected. In a Machine Learning context, this implies a dataset that is not only severely imbalanced but also exhibits noisy labeling.

5.1.2 Exploratory Data Analysis

The knowledge engineering phase described in Section 4.4 required a thorough study of the properties of power line supporting towers, their relation with the environment and the analysis of other scene elements that may exhibit similar features to the ones defined. To this end, we performed an exploratory data analysis of the raw point clouds provided by EDP. Specifically, we took advantage of DBSCAN [41], a density clustering algorithm, to aggregate the points of each tower instance into individual groups. Essentially, DBSCAN clusters points into different neighborhoods according to a minimum number of

Label	Class	Density(%)	Label	Class	Density(%)
0	Created	0	11	Road surface	**18.758**
1	Unclassified	0.571	12	Overlap points	23.403
2	Ground	0.529	13	Medium Reliability	0
3	Low vegetation	0.681	14	Low Reliability	0
4	Medium vegetation	0.241	15	Power line support tower	**0.519**
5	Natural obstacle	1.069	16	Main power line	0.907
6	Human structures	0	17	Other power line	0.002
7	Low point	0.362	18	Fiber optic cable	0
8	Model keypoints	0	19	Not rated object to be consider	8.205
9	Water	0	20	Not rated object to be ignored	0
10	Rail	**44.752**	21	Incidents	0

Table 5.1: Available classes in the TS40K dataset and their distribution. Rail lines and road surface constitute the majority of the dataset (63%). Whereas our class of interest, power line support tower, only makes up 0.52%. Moreover, around 40% of tower points are mislabeled.

neighbors n at distance ϵ or less, if any point does not meet this criterion, it is considered noise.

(a) 3D points labeled as supporting towers. (b) DBSCAN output

Figure 5.2: Application of DBSCAN on 3D points of power line supporting towers.

The parameters n and ϵ were fine-tuned to form neighborhoods where each cluster represents a tower. Figure 5.2 illustrates this process, Figure 5.2a shows a point cloud where only the points labeled as power-line supporting towers were selected (which can be directly obtained from the original point cloud), and Figure 5.2b presents the output of the DBSCAN algorithm. By examining the latter, we note that each tower is perfectly segmented, allowing us to further analyze and compute their attributes.

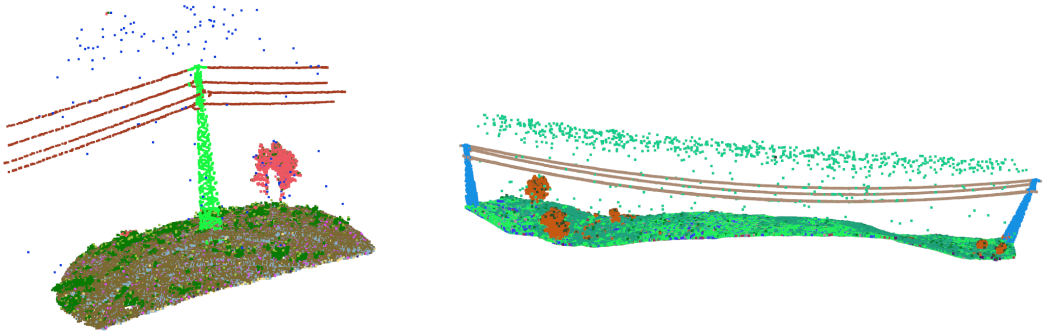
Power line supporting towers exhibit the following geometrical properties:

- The number of points that constitute a tower ranges between 334 and 5860;
- The base of the tower is squared shaped, with average dimensions of 4.15 width and 4.2 length;
- Their height is on average 34.74 and can vary between 18 and 40 meters;
- The distance between towers is between 245 and 500 but averages at 300 meters;
- There are no towers without power lines;

- Their shape can be described as a square pyramid.

Lastly, the results obtained from applying the DBSCAN algorithm on the tower point cloud serve as empirical proof that employing this algorithm as a post-processing step on the output of a 3D semantic segmentation model yields adequate results for the tower coordinate resolution problem. Therefore, focusing our study and development on 3D semantic segmentation is justified.

5.1.3 Ancillary Dataset



(a) Radius sectioning. (b) Tower to tower sectioning.

Figure 5.3: Point cloud segments sectioned for better evaluation of proposed methods.

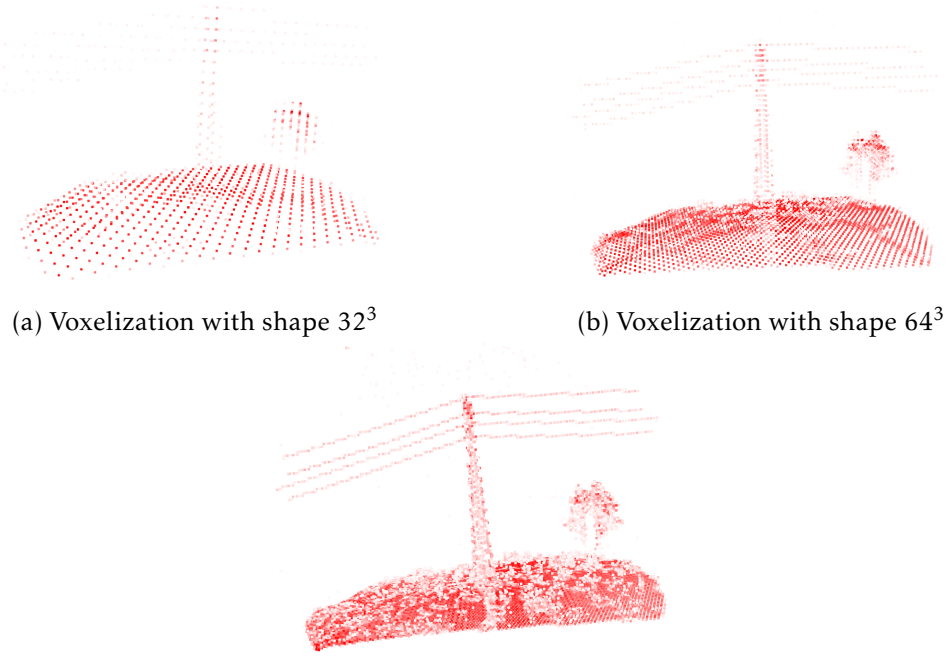
To mitigate the severe data imbalance on the TS40K dataset that is further aggravated by voxelization, we create an ancillary dataset focused on power line supporting towers with 2823 samples. For each tower in the 3D data, we crop the surrounding ground with a radius equal to its height as shown in Figure 5.3a. This process introduces bias in classical Machine Learning agents, such as CNNs, since we are presenting them with an unrealistic density of supporting towers to the model. This leads to overfitting and higher predicted likelihoods for the overall 3D scene. Contrastingly, SCENE-Net is optimized, after incorporating the appropriate prior knowledge, to detect the chosen features that describe the ground truth. A biased training dataset results in an agent tailored to detect the geometry of supporting towers. Elements in the 3D scene that do not align with these properties are not signaled by SCENE-Net.

5.1.4 Voxelization Results

In order to endow TS40K point clouds with structure, we developed a function that receives a point set P and a voxel grid dimension s, and produces a 3D array describing the voxelization of P with shape s. Specifically, we first represent P as a voxel grid, its space is divided into K voxels that contain a subset of points $P_i \subseteq P$, $1 \leq i \leq K$ of the original point cloud. This way, the input point set is endowed with structure, each voxel is neighbor to other voxels in predefined directions, analogous to pixels in an image.

Then, the structured point cloud and ground truth are subject to a measurement φ. This function distinguishes between occupied and empty voxels, assigning them a value of one and zero, respectively. Formally, φ acts upon 3D points in order to provide an appropriate measure according to the GENEO framework. In practice, we apply it directly to the 3D voxels for easier computation:

$$\varphi'(P_K) = \begin{cases} 1 & \text{if } P_K \neq \emptyset \\ 0 & \text{if otherwise} \end{cases} .$$



(a) Voxelization with shape 32^3

(b) Voxelization with shape 64^3

(c) Voxelization with shape 128^3

Figure 5.4: Voxelization of the point cloud in Figure 5.3a. Figure 5.4a shows that, although the tower and power lines are present in the voxel grid, their resolution compromises the geometrical properties of the scene. In Figure 5.4b, we can clearly visualize the elements in the 3D point cloud. The 128^3 resolution in Figure 5.4c produces sharper results, but does not seem to add crucial details to the structure of the power-grid. The point clouds are colored in order to discriminate the captured details in each voxel-resolution.

Voxelization techniques introduce a bottleneck in performance due to their computational cost. In our case, voxelization with resolutions of 32^3, 64^3 and 128^3 averaged, respectively, a processing time of 0.16 s, 0.45 s and 1.18 s on an Intel CPU of 8th generation without parallelization. Nevertheless, the high quality of the TS40K point clouds allow us to voxelize the data with lower resolutions and still maintain a good level of detail. This representation and measurement function emphasize the geometry of supporting towers, while preserving the geometric properties of 3D scenes. Even though voxelization worsens the data imbalance in the TS40K dataset, the cropped sampling mitigates this issue.

5.2 Results

5.2.1 Training Protocol.

During the end-to-end training process of SCENE-Net, we adopt the following settings: batch size is 8 for a total of 50 epochs. We employ the Root Mean Squared Propagation (RMSProp) optimizer, with a learning rate of 0.001. The weighting scheme parameters α and ϵ are set to 5 and 0.1, respectively. While both scaling factors of the non-positive penalty ρ_l and ρ_t are set to 5. The kernel size used to discretize the GENEO operators is 9^3. The GENEOs parameters ϑ are randomly initialized under suitable and positive ranges. While the convex coefficients $\lambda_0, \ldots, \lambda_{n-1}$ are randomly initialized in the range $[0, \frac{2}{N-1}]$ to promote a valid convex space for $\mathscr{H}$. To demonstrate that SCENE-Net achieves good results even with less data, we use 20% of the dataset for training, 10% for validation and 70% for testing. All experiments were conducted on an NVIDIA GeForce RTX 3070 GPU.

5.2.2 Interpretability of the trained SCENE-Net. The meaning of the 11 learned parameters.

To understand if SCENE-Net is interpretable, we inspect its 11 trainable parameters ϑ and λ after training. Each $\vartheta_i \in \vartheta$ holds the learned shape parameters of a geometrical operator Γ_i, such as their height or radius. The convex coefficients λ weigh each operator Γ_i in the analysis of our model. For example, we can conclude that the instance ϑ_{NS} of the Negative Sphere GENEO (Γ_{NS}) holds a weight of 76.34% on the output of SCENE-Net (Figure 5.5). The geometric nature of the observer and combination parameters endow intrinsic **interpretability** to SCENE-Net.

5.2.3 Post hoc interpretation for specific predictions.

The geometric operators in SCENE-Net also enable a post hoc interpretation of its predictions. Specifically, we can correlate the detection of scene elements, such as vegetation, to the contributions of each GENEO. This provides an extra layer of transparency to our model. For instance, Figure 5.6 illustrates the convolution of each GENEO-kernel with a TS40K scene. The Arrow kernel is responsible for the detection of towers, the Cylinder aids this process and diminishes the detection of vegetation and the Negative Sphere stabilizes the model by balancing the contributions of the previous kernels. By dissecting the analysis of the observer, we can fast-track the knowledge engineering phase of GENEO-based models. Interpretable models are defined as constrained problems in order to ensure human understanding. Such constraints are hard to formulate and ensure that they fully describe our problem. In our case, these constraints are present in the form of GENEO-kernels and are ensured by the devised GENEO loss. However, it is difficult and ambiguous to assert if the defined prior knowledge is sufficient to model the problem. A post hoc interpretation of the predictions of SCENE-Net allows us to analyze

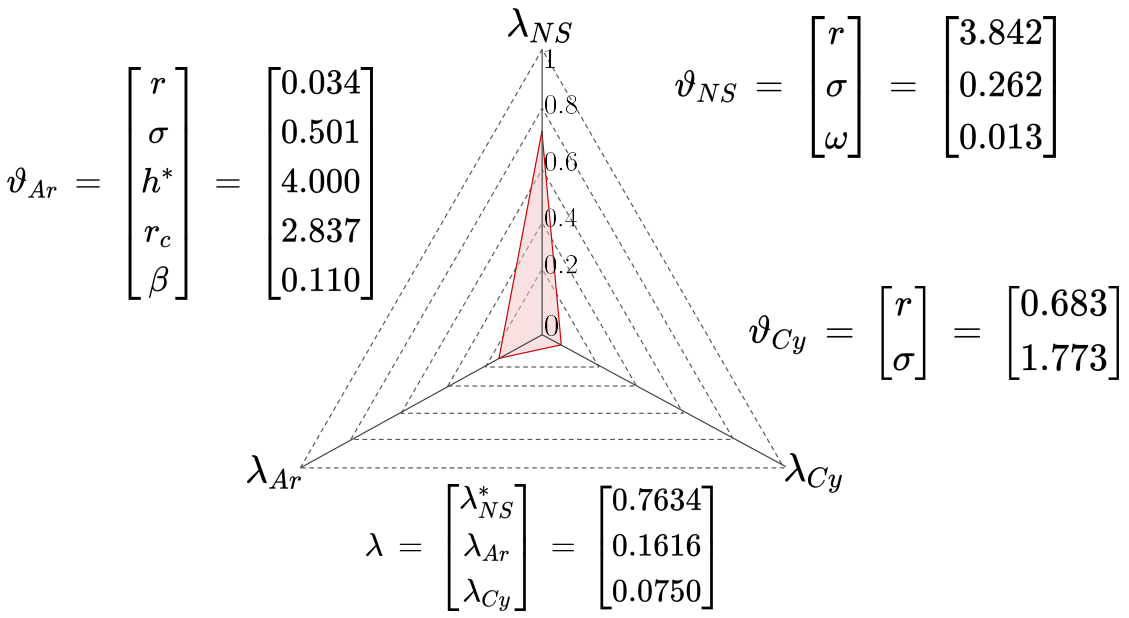
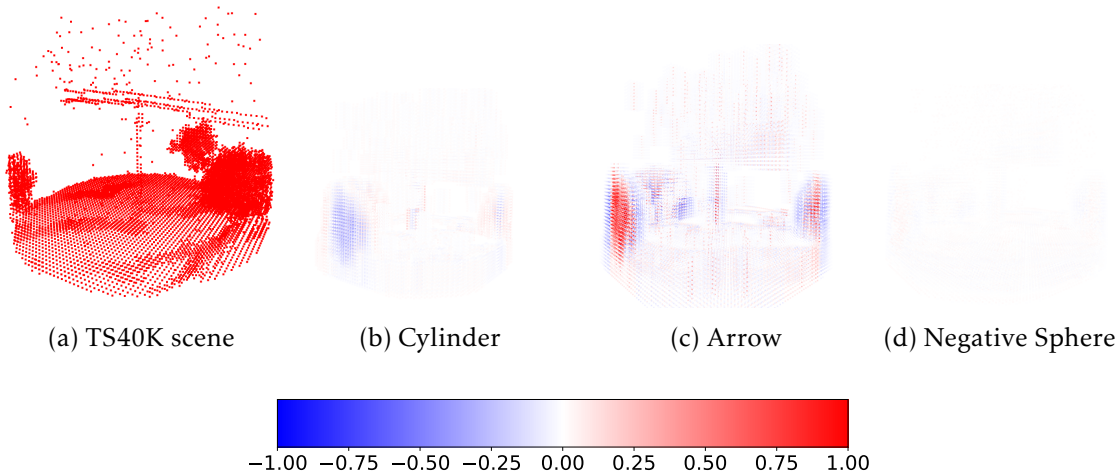


Figure 5.5: The trainable parameters of SCENE-Net ϑ and λ in an interpretable visualization. Parameter h^* is not trainable, and λ^*_{NS} is defined as a function of the other mixing weights $\lambda^*_{NS} = 1 - \lambda_{Ar} - \lambda_{Cy}$.

how the different GENEO-kernels behave and what properties are not being captured, which leads to a faster and more complete definition of our model.



(a) TS40K scene (b) Cylinder (c) Arrow (d) Negative Sphere

Figure 5.6: Post hoc analysis of SCENE-Net. We can examine the activation of each geometric operator and correlate it to the detection of certain elements in the scene. We see that the Arrow is responsible for the most activation, while the Negative Sphere has smaller absolute value.

5.2.4 Qualitative accuracy and quantitative metrics: SCENE-Net more precise in detecting towers than the baseline CNN.

5.2.4.1 Useful Metrics.

		Actual Value: Positive	Actual Value: Negative
Predicted value	Positive	True Positive (TN)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Table 5.2: Confusion Matrix

During training, SCENE-Net regresses the probability of each voxel belonging to a supporting tower. Afterward, a threshold probability τ is fine-tuned in order to discern between non-tower and tower voxels and, this way, optimize the detection of supporting towers. This binary segmentation of 3D point clouds can be used to draw useful evaluation metrics, namely Precision, Recall, IoU, and $F_\beta$score. Table 5.2 illustrates a conventional confusion matrix in ML applications. A voxel is correctly segmented if it is either a True Positive or True Negative, meaning that our prediction coincides with the actual value. False Positives represent voxels incorrectly classified as a tower, and False Negatives, on the contrary, denote tower voxels misclassified as non-tower.

Supporting towers normally make up a small volume of the entire 3D scene, and the empty space captured during voxelization leads to a greater discrepancy between tower and non-tower voxels. Consequently, the number of True Negatives overshadows the other cases, which makes $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$ unfit to assess both SCENE-Net and the CNN baseline. Therefore, we resort to Precision and Recall. $Precision = \frac{TP}{TP+FP}$ tells us from all voxels predicted positively, what percentage did the model classify correctly. This metric provides insight with regard to the impact of false positive predictions, a low Precision means that our model is overconfident in its predictions and classifies most scene elements as a tower. In turn, $Recall = \frac{TP}{TP+FN}$ measures how many tower voxels were correctly classified, which essentially lets us evaluate how our agent is modeling supporting towers when compared to the ground truth. A low Recall entails that very few tower voxels were classified as such by our model. However, the noisy labeling present in the TS40K dataset affects the true meaning of this metric in our context. A higher Recall does not necessarily imply a better model.

The F_1Score is the harmonic mean between Precision and Recall, so it portrays the balance between these metrics. Since we value Precision over Recall, we specifically work with the F_βScore where $\beta = 0.5$, meaning that we weigh Precision twice as much as we do Recall.

Intersection over Union (IoU) follows the definition presented in 2.1.7. With respect

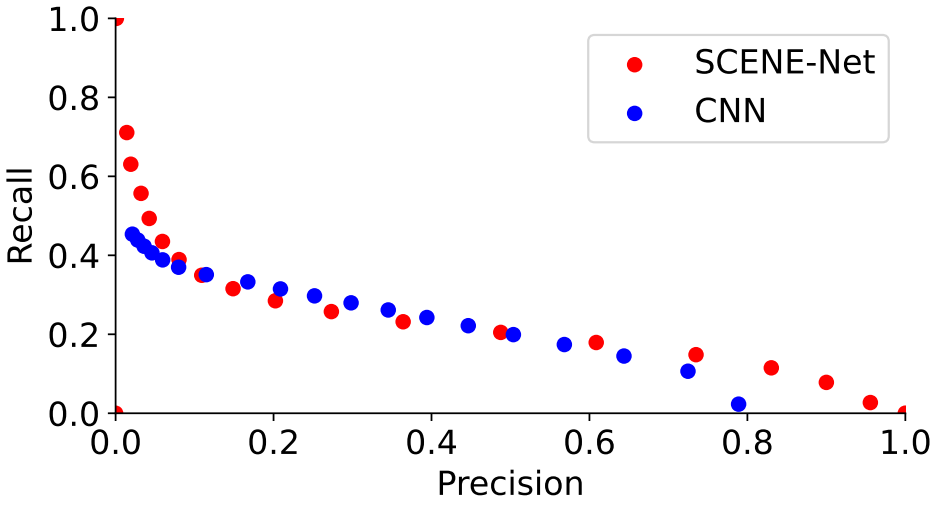


Figure 5.7: Precision-Recall curve for SCENE-Net and the CNN benchmark, with changing detection threshold. Although our model SCENE-Net has two orders of magnitude fewer parameters than the CNN, it attains a comparable area under the P-R curve.

to the confusion matrix, IoU is defined as $IoU = \frac{TP}{TP+FN+FP}$, which essentially measures the overlap between the prediction and the ground truth, over the total volume they occupy.

5.2.4.2 Performance of SCENE-Net.

To evaluate if SCENE-Net can correctly identify towers in landscapes of the noisy TS40K dataset, we chose the task of 3D semantic segmentation of supporting towers, and trained SCENE-Net and a baseline CNN with similar architecture according to the protocol described in Section 5.2.1. Domain experts are interested in detecting supporting towers as well as computing their (x, y) center in geographical coordinates to integrate this model in their inspection procedures. The application penalizes false positives more, thus we emphasize Precision. Due to the imbalanced nature of the labels, we measured overall Precision, Recall and IoU. Quantitatively, we observe a lift in Precision of 24%, and 5% in IoU, and a drop of 10% in Recall (Table 5.3).

Method	Precision	Recall	IoU
CNN	0.44 (± 0.07)	**0.26** (± 0.02)	0.53
SCENE-Net	**0.68** (± 0.08)	0.16 (± 0.05)	**0.58**

Table 5.3: Semantic segmentation metrics on TS40K.

The lower Recall of SCENE-Net is due to noisy labels in the ground truth. As shown in Figures 1.1 and 5.9, the ground surrounding supporting towers as well as power lines are often mislabeled as a tower.

In Figure 5.8, we can analyze the qualitative results on the TS40K dataset (a) of SCENE-Net (c) against a CNN is similar architecture as a baseline (b). Even though the CNN achieves a higher Recall on the majority of the samples, it classifies most vertical scene elements as a tower, which ultimately leads to a poor segmentation of supporting towers and a less precise xy tower coordinate prediction. In contrast, SCENE-Net segments the body of towers and rejects other vertical objects that do not exhibit the prior knowledge encoded in the model.

5.2.5 SCENE-Net is robust to noisy labels.

It is important to assess the resilience to noisy labels in the ground truth (GT), since 3D point clouds show more than 50% of mislabeled points. These examples are abundant in the dataset and SCENE-Net is able to recover the body of the tower without detecting ground and power line patches that are mislabeled as tower (Figure 5.9). The CNN has, in general, the same behavior. Most noisy labels on this kind of dataset are due to annotation excess around the object of interest, and they are not randomly distributed. These consistently incorrect labels entail low Recall values (Table 5.3).

5.2.6 SCENE-Net has modest training and inference time in common hardware.

To assess if the models can be used with the computational resources of a utility company, we computed the average training and inference times of both the CNN and SCENE-Net in an NVIDIA GeForce RTX 3070 GPU. Only one in five trained CNNs returns non-zero predictions. Each training session (50 epochs) takes in average 85 mn for a trained SCENE-Net and a CNN. In inference, SCENE-Net takes 20 ms while the CNN takes 43 ms, less 23 ms per inference. The CNN has 2190 trainable parameters, whereas SCENE-Net has 11. The difference in trainable parameters grows exponentially with larger kernel sizes, since SCENE-Net has 11 parameters regardless of kernel size. Running models for 3D point cloud semantic segmentation [20, 22, 24, 49] was not done due to their computational requirements. Training times allow SCENE-Net to be retrained from scratch in less than 90 mn. The CNN is also trainable, taking on average 8 h to obtain a useful model.

5.2.7 SCENE-Net inference in high resolution, when trained with low-resolution kernel sizes.

One of the issues of voxel-based models is the computational cost of 3D convolutions with large kernels and high-resolution voxel grids. Here, a CNN architecture leads to an exponential increase in training time. SCENE-Net has a continuous functional observer of the raw input providing an analysis of its components. Unlike traditional models, this definition is **independent** of the input size as well as its own discretization (kernel size). In this experiment, we trained SCENE-Net with voxel grids of 64^3 and then applied to

higher resolutions, such as 128^3, with good qualitative results (Figure 5.10). The kernel size used to discretize the operators can be fine-tuned to enhance performance (Tab. 5.4).

Kernel-size	Precision	Recall	IoU
$9 \times 9 \times 9$	0.37 ($\pm$ 0.02)	**0.22** ($\pm$ 0.01)	0.58
$9 \times 5 \times 5$	**0.68** ($\pm$ 0.08)	0.16 ($\pm$ 0.05)	**0.58**

Table 5.4: Performance of SCENE-Net with different kernel sizes on TS40K. SCENE-Net is trained with a kernel size of 9^3, which is later fine-tuned to find the sweet spot between Precision and Recall.

5.3 Experiments

5.3.1 Running State-of-the-Art Models

Even though utility companies, such as EDP, do not have the requirements to employ 3D semantic segmentation state-of-the-art methods, we made an effort to train them with the TS40K dataset in order to provide an additional benchmark to our research. The publicly available repositories [17, 20–22, 49] are tailored to emulate the performed experiments on autonomous driving datasets, such as SemanticKITTI [10]. Moreover, the available code often lacks documentation and is written in old DL frameworks, like TensorFlow 1.0. Two high-level APIs for running state-of-the-art DL models [64, 65] were also tested. However, [65] specialized to work with famous benchmark datasets and did not account for custom datasets such as ours. We tried to perform transfer learning with the existing models on the framework of Chaton et al. [64], seeing as we did not have the resources or time to train them from scratch. Unfortunately, the API is not compatible with this sort of operation.

5.3.2 Measurement Function

Choosing appropriate measurement functions is a crucial step when taking advantage of the GENEO-framework. These functions allow us to shift our attention from the raw data, to the possible measurements we can extract from it, which leads to a more diverse definition of GENEO observers. Initially, we defined our measure φ as a density function, that outputs a normalized count of the number of points in each voxel of a discretized point cloud. This way, we hoped to stay true to the original data distribution and make the model take advantage of the fact that objects of interest, such as supporting towers, usually have less point density than non-relevant classes, such as the ground. However, the achieved results were subpar to what we expected. By assessing the analysis of the GENEO observer $\mathscr{H}$, we noticed that, even though SCENE-Net was able to detect towers accurately, it was not able to output a substantial activation for objects with lower point density because it also had to diminish the importance of high point density elements,

such as the ground. Therefore, we decided to employ a measurement that considers all occupied voxels as equals. By providing a uniform distribution of values to different 3D elements, GENEO-kernels were able to tailor their embedded knowledge to detecting objects of interest. This decision produced a performance increase of 42% in terms of loss.

5.3.3 GENEO Loss

One could argue that our input and output data define a traditional segmentation problem with only one class of interest, which usually takes advantage of binary cross entropy to define the data fidelity component in loss functions. Cross entropy was developed as a response to the vanishing gradient of squared errors, such as MSE, when the derivative is close to zero for classification problems. However, both at a conceptual level and in terms of performance, binary cross entropy is not suited for this problem. The values of zero and one in the ground truth of the TS40K dataset do not represent classes (i.e., tower or non-tower), they represent a level of confidence that a particular voxel contains any points classified as a tower. Thus, at its core, we are approximating the probability of a voxel containing tower points, and not classifying voxels. Nevertheless, we substituted the squared error in the data fidelity component of $\mathscr{L}_{seg}$ with binary cross entropy and assessed the performance of SCENE-Net. Training and testing demonstrated all-around worse quantitative and qualitative results.

In addition, the initial definition of the penalty function h punished negative weights quadratically. This would often result in small negative convex coefficients because of low derivative values near zero, which violates the convex space of the GENEO observer. Therefore, we changed it to a linear penalization as illustrated in Figure 4.5 to correct this behavior.

The weighting scheme enforced by function f_w proved essential for the performance of SCENE-Net, without it the error measured by $\mathscr{L}_{seg}$ would always be insignificant and no training would actually be done. The hyperparameters α and ϵ of f_w were the most important values to calibrate due to the severe data imbalance that was worsened by voxelization, so we tested a total of 55 combinations, specifically $\alpha \in [0.001, 0.1, 1, 2, 3, 5, 8, 10, 20, 50, 100]$ and $\epsilon \in [0.00001, 0.0001, 0.001, 0.01, 0.1]$. α emphasizes the weighting scheme, it essentially quantifies how much we prioritize regressing tower from non-tower voxels. A high α leads to a high number of False Positives since the model only focuses on modeling tower voxels, whereas a low α makes SCENE-Net obsolete because tower voxels were a negligible component to the loss measure and therefore not detected. In turn, ϵ defines the weight given to non-tower voxels, seeing as these are the grand majority in the dataset. This parameter exhibits a symmetric behavior to α. High ϵ values result in a model purely focused on not detecting non-tower voxels, and low values make SCENE-Net assume that regressing non-tower voxels is not important, so most scene elements are deemed as towers by our model.

5.3.4 Training Protocol

The training protocol presented in 5.2.1 was subject to a lot of experimentation and fine-tuning. Specifically, we tried the following settings:

- Given that SCENE-Net exhibits fast training, we tested a different combination of epochs and batch size, with $epochs \in [10, 25, 50, 100, 500, 1000]$ and $batch\_size \in [2, 4, 8, 16, 24, 32]$. Bigger batch sizes lead to worse training times due to GPU parallelization, and SCENE-Net yielded effective results with no more than 10 epochs, but 50 epochs seemed to provide more consistency to the model across different runs;

- We experimented with different learning rates lr and tried to impose a learning rate decay lr_decay, specifically, $lr \in [0.1, 0.001, 0.0001, 0.00001]$ and a $lr\_decay \in [5\%, 10\%, 20\%, 25\%]$ every 10, 20, or 25 epochs. Due to the empty volume in the voxelization, the error provided by the loss function ranges between 0.0022 and 0.0006. As a result, any kind of learning rate decay strategy would immediately halt training for the parameters of the network, so this technique is not employed;

- As detailed in Section 5.2.7, the kernel size employed during training greatly affects the performance of SCENE-Net. So, we experimented with sizes (6, 5, 5), (6, 6, 6), (9, 5, 5), (9, 6, 6), (8, 8, 8) and (9, 9, 9) in the form (height, width, length). We concluded that bigger kernel sizes endow SCENE-Net with more freedom to explore the embedded prior knowledge, which results in a better and more consistent performance;

- Random initialization is a crucial factor for the performance of traditional ML, such as CNNs, and our model is no different. The trainable parameters of SCENE-Net λ, ϑ are randomly initialized within suitable ranges to promote a stable convex space for the network to navigate. Using other initialization strategies, such as not enforcing ranges or using handcrafted values, resulted in a worse performance by the model, or it would take longer to converge to an acceptable state;

- We examined the impact of using different optimizers, namely Stochastic Gradient Descent (SGD), adaptive moment estimation (Adam) and RMSProp. There was no noticeable difference in the performance of SCENE-Net, with RMSProp showing slightly better results than the other two;

- The data split used in our experiments is unorthodox when compared to conventional DL techniques. Usually, the biggest subset of data is reserved for training the model because of the millions of parameters that require fine-tuning. Seeing as SCENE-Net is composed of only 11 trainable parameters, training is effectively performed with little more than 500 samples. This is paramount for traditional companies, such as utilities, that do not have a lot of labeled data at their disposal.

This way, they can achieve a straightforward and interpretable model with a lot less data than state-of-the-art black-box methods.

5.3.5 Ablation Studies.

Model	Cylinder	Arrow	Neg. Sphere	Precision	Recall	IoU
A	1	0	0	0	0	0.50
B	0	1	0	0	0	0.50
C	1	0	1	0.34	0.01	0.52
D	0	1	1	0.13	0.01	0.50
E (Ours)	**1**	**1**	**1**	**0.68**	**0.16**	**0.58**
F	2	2	2	0.56	0.03	0.53
G	3	3	3	0.37	0.22	0.56

Table 5.5: Ablation Study of SCENE-Net on TS40K validation set.

We conducted ablation studies on the architecture of SCENE-Net, specifically on the number of instances of each GENEO. All ablated models were tested on the TS40K validation set. Table 5.5 shows the following results: Models A and B are each equipped with a single GENEO, demonstrate an overall poor performance. The Negative Sphere (NS) GENEO proved essential for our observer to disregard arboreal elements in the scene. Models C and D study if employing the Cylinder or Arrow combined with NS is enough to analyze the TS40K scenes. However, the architecture of SCENE-Net (model E) yields better results. Lastly, models F and G test the use of multiple instances of each GENEO, but this proved to decrease performance when compared to model E.

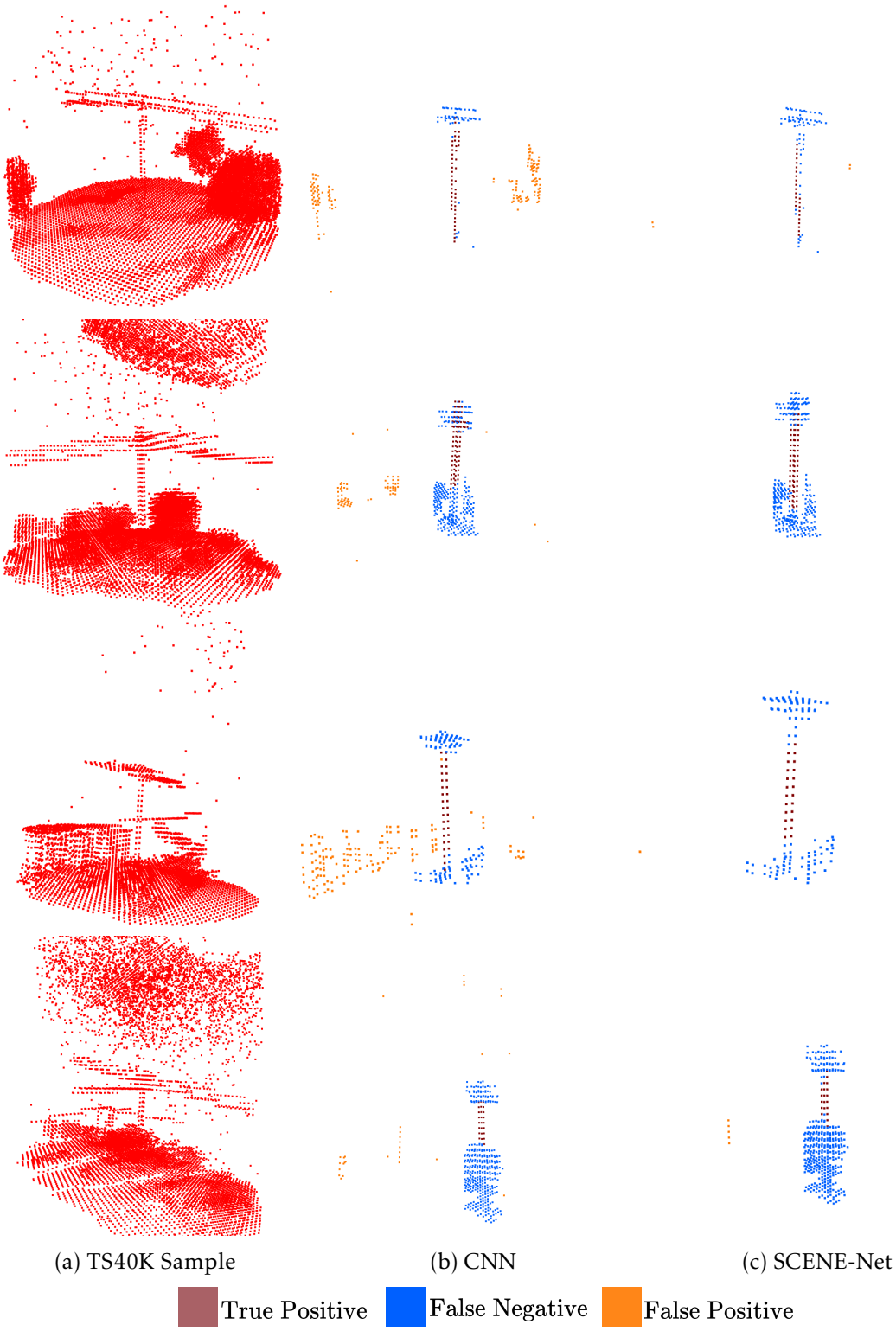
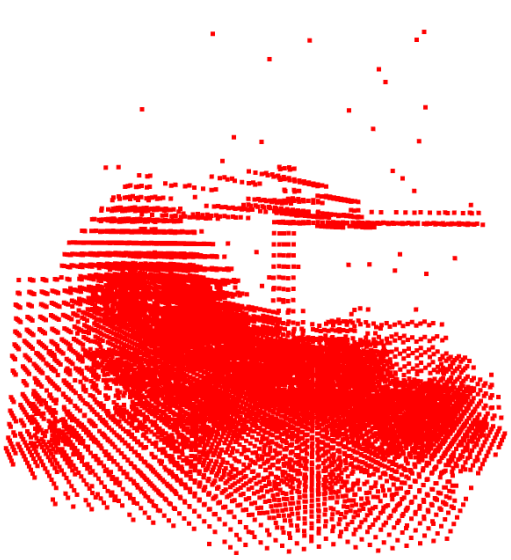
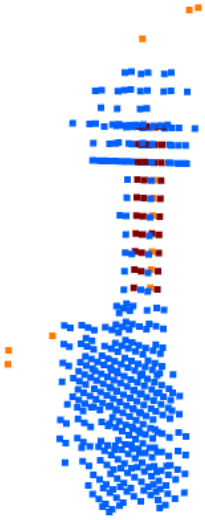


Figure 5.8: Qualitative results of SCENE-Net on the testing set of TS40K, against a CNN with similar architecture. Note that, in the last example, SCENE-Net clearly identifies a second unlabeled tower. For the same sample the CNN identifies both the secondary tower and vegetation as towers.



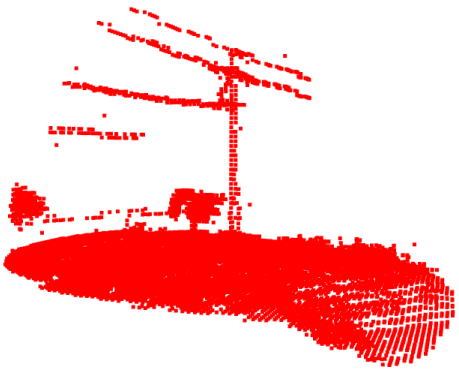
(a) TS40K sample.



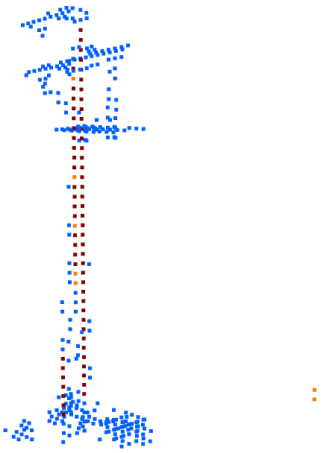
(b) SCENE-Net prediction against the GT.

True Positive False Negative False Positive

Figure 5.9: SCENE-Net is robust against mislabeled data. Figure 5.9b compares the prediction of SCENE-Net against the ground truth in Figure 5.9a. SCENE-Net detects the body of the tower, ignoring the patch of ground mislabeled as tower.



(a) Sample discretized in a 128^3 grid.



(b) SCENE-Net prediction against the GT.

True Positive False Negative False Positive

Figure 5.10: SCENE-Net is independent from the input and kernel size. Our model was trained with voxel grids of shape 64^3 and kernel size 9^3. Figure 5.10b shows a SCENE-Net prediction against the ground truth of the 128^3 input grid in Figure 5.10a using a kernel size of $12 \times 5 \times 5$.

6

Conclusion

With data becoming the new oil, Machine Learning strategies have taken a crucial role in tackling real-world problems, such as in healthcare and power grid inspection. But traditional companies, like utilities, need ML to shed light on its opaqueness. They require resource-efficient, straightforward, and trustworthy ML models in order to guarantee their responsible application to real-life scenarios. For instance, utility companies, such as EDP, would greatly benefit from an agent capable of segmenting real-world point clouds while providing sound reasoning behind its decision-making, so that inspecting thousands of kilometers of the power grid can be both fast and safe.

In this dissertation, we propose **SCENE-Net**, an intrinsically interpretable 3D point cloud semantic segmentation model identifying signature shapes with GENEOs for power line tower segmentation on 3D rural landscapes. By taking advantage of GENEOs, a straightforward architecture, and an imbalanced-aware loss function, our model outperforms a benchmark CNN in Precision by 24% with only 11 trainable geometrical parameters, two orders of magnitude less than a Precision-comparable CNN. We focused on simple signature shapes: the cylinder, arrow, and negative sphere. To improve on metrics we could add complex invariant shapes finely describing properties of interest. Here we would pay the price of eliciting detailed properties and narrowing the application field of the model. This knowledge engineering stage, although necessary to guarantee interpretability, requires both expertise and a laborious process to ensure a complete definition of properties of interest. Thus, scenarios should be carefully evaluated to determine if the additional workload of achieving transparency is outweighed by its benefits. **From our experience in developing and implementing SCENE-Net with a utility company, such systems can critically help human decision-making**—here, by facilitating fast and careful inspection of power lines, with interpretable signals of observed geometrical properties. With only three shape observers, and 11 physically meaningful parameters, SCENE-Net can help diminish the risk of power outages and forest fires, by learning from data.

This work opens the door for exciting new research. On an immediate note, we can detect other relevant elements of the TS40K dataset, such as the power lines and vegetation,

to automatically detect potential damages to the power grid. On the other hand, it would be interesting to develop a high-level API to aid the development of GENEO observers in an ML paradigm. This way, we can expedite the knowledge engineering phase in other applications and encourage the wide use of GENEO-based models in Machine Learning.

Bibliography

[1] H. Gerard, E. I. R. Puente, and D. Six. "Coordination between transmission and distribution system operators in the electricity sector: A conceptual framework". In: *Utilities Policy* 50 (2018), pp. 40–48 (cit. on p. 1).

[2] M. Malveiro, R. Martins, and R. Carvalho. "Inspection of high voltage overhead power lines with UAV's". In: *Proceedings of the 23rd International Conference on Electricity Distribution*. 2015 (cit. on p. 2).

[3] J. Choi et al. "Multi-Target Tracking using a 3D-Lidar sensor for autonomous vehicles". In: *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. 2013, pp. 881–886 (cit. on pp. 2, 8).

[4] D. Alexiadis, D. Zarpalas, and P. Daras. "Fast and smooth 3D reconstruction using multiple rgb-depth sensors". In: *2014 IEEE Visual Communications and Image Processing Conference*. IEEE. 2014, pp. 173–176 (cit. on pp. 2, 7).

[5] H. Song, W. Choi, and H. Kim. "Robust vision-based relative-localization approach using an RGB-depth camera and LiDAR sensor fusion". In: *IEEE Transactions on Industrial Electronics* 63.6 (2016), pp. 3725–3736 (cit. on pp. 2, 7).

[6] Y. Guo et al. "Rotational projection statistics for 3D local surface description and object recognition". In: *International journal of Computer Vision* 105.1 (2013), pp. 63–86 (cit. on p. 2).

[7] Y. Guo et al. "3D object recognition in cluttered scenes with local surface features: A survey". In: *IEEE transactions on pattern analysis and machine intelligence* 36.11 (2014), pp. 2270–2287 (cit. on p. 2).

[8] M. A. Uy et al. "Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 1588–1597 (cit. on pp. 3, 8, 50).

[9] T. Hackel et al. "Semantic3d. net: A new large-scale point cloud classification benchmark". In: *arXiv* (2017) (cit. on pp. 3, 4, 21, 22, 24).

[10] J. Behley et al. "Semantickitti: A dataset for semantic scene understanding of lidar sequences". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9297–9307 (cit. on pp. 3, 8, 21, 22, 24, 50, 59).

[11] A. Geiger, P. Lenz, and R. Urtasun. "Are we ready for autonomous driving? the kitti vision benchmark suite". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2012, pp. 3354–3361 (cit. on pp. 3, 4, 8, 18, 20, 50).

[12] F. J. Lawin et al. "Deep projective 3D semantic segmentation". In: *International Conference on Computer Analysis of Images and Patterns*. Springer. 2017, pp. 95–107 (cit. on pp. 3, 9).

[13] D. Maturana and S. Scherer. "Voxnet: A 3D convolutional neural network for real-time object recognition". In: *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2015, pp. 922–928 (cit. on pp. 3, 9).

[14] J. Long, E. Shelhamer, and T. Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2015, pp. 3431–3440 (cit. on pp. 3, 10).

[15] D. Rethage et al. "Fully-convolutional point networks for large-scale point clouds". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 596–611 (cit. on pp. 3, 10).

[16] L. Tchapmi et al. "Segcloud: Semantic segmentation of 3D point clouds". In: *2017 International Conference on 3D vision (3DV)*. IEEE. 2017, pp. 537–547 (cit. on pp. 3, 10).

[17] C. R. Qi et al. "Pointnet: Deep learning on point sets for 3D classification and segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 652–660 (cit. on pp. 3, 10–12, 20, 22, 59).

[18] C. R. Qi et al. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space". In: *Advances in neural information processing systems* 30 (2017) (cit. on pp. 3, 10, 12, 18, 22, 24).

[19] S. Shi, X. Wang, and H. Li. "Pointrcnn: 3D object proposal generation and detection from point cloud". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 770–779 (cit. on pp. 3, 18, 22).

[20] H. Thomas et al. "Kpconv: Flexible and deformable convolution for point clouds". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 6411–6420 (cit. on pp. 3, 15, 17, 22, 24, 58, 59).

[21] Q. Hu et al. "Randla-net: Efficient semantic segmentation of large-scale point clouds". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11108–11117 (cit. on pp. 3, 16, 22, 24, 25, 59).

[22] X. Zhu et al. "Cylindrical and asymmetrical 3D convolution networks for lidar segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 9939–9948 (cit. on pp. 3, 22, 24, 58, 59).

[23] M. Xu, Z. Zhou, and Y. Qiao. "Geometry sharing network for 3D point cloud classification and segmentation". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07. 2020, pp. 12500–12507 (cit. on pp. 3, 24).

[24] X. Yan et al. "Sparse single sweep LiDAR point cloud segmentation via learning contextual shape priors from scene completion". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 4. 2021, pp. 3101–3109 (cit. on pp. 3, 25, 58).

[25] G. Tao et al. "Study on segmentation algorithm with missing point cloud in power line". In: *2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*. IEEE. 2019, pp. 1895–1899 (cit. on pp. 4, 27).

[26] T. Guo et al. "Research on point cloud power line segmentation and fitting algorithm". In: *2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. Vol. 1. IEEE. 2019, pp. 2404–2409 (cit. on pp. 4, 26, 27).

[27] L. Ding, J. Wang, and Y. Wu. "Electric power line patrol operation based on vision and laser SLAM fusion perception". In: *2021 IEEE 4th International Conference on Automation, Electronics and Electrical Engineering (AUTEEE)*. IEEE. 2021, pp. 125–129 (cit. on pp. 4, 26).

[28] M. G. Bergomi et al. "Towards a topological–geometrical theory of group equivariant non-expansive operators for data analysis and machine learning". In: *Nature Machine Intelligence* 1.9 (2019), pp. 423–433 (cit. on pp. 4, 24, 31–36, 38, 39, 41).

[29] P. Cascarano et al. *On the geometric and Riemannian structure of the spaces of group equivariant non-expansive operators*. 2021. arXiv: 2103.02543 [math.DG] (cit. on pp. 4, 31, 35, 36, 38).

[30] Q. Hu et al. "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*. 2021, pp. 4977–4987 (cit. on pp. 8, 21, 22, 24).

[31] S. A. Bello et al. "Review: Deep Learning on 3D Point Clouds". In: *Remote Sensing* 12.11 (May 2020), p. 1729. ISSN: 2072-4292 (cit. on pp. 9, 10, 22).

[32] H. Su et al. "Multi-view convolutional neural networks for 3D shape recognition". In: *Proceedings of the IEEE international Conference on Computer Vision*. 2015, pp. 945–953 (cit. on p. 9).

[33] Z. Yang and L. Wang. "Learning relationships for multi-view 3D object recognition". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 7505–7514 (cit. on p. 9).

[34] Z. Wu et al. "3D shapenets: A deep representation for volumetric shapes". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1912–1920 (cit. on p. 9).

[35] Y. Guo et al. "Deep Learning for 3D Point Clouds: A Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.12 (2021), pp. 4338–4364 (cit. on pp. 9–11, 22, 24, 25).

[36] D. Maturana and S. Scherer. "3D Convolutional Neural Networks for landing zone detection from LiDAR". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 3471–3478 (cit. on p. 10).

[37] Y. Zhou and O. Tuzel. "Voxelnet: End-to-end learning for point cloud based 3D object detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4490–4499 (cit. on pp. 10, 13, 19, 22).

[38] G. Riegler, A. Osman Ulusoy, and A. Geiger. "Octnet: Learning deep 3D representations at high resolutions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 3577–3586 (cit. on p. 10).

[39] M. Tatarchenko, A. Dosovitskiy, and T. Brox. "Octree generating networks: Efficient convolutional architectures for high-resolution 3D outputs". In: *Proceedings of the IEEE international Conference on Computer Vision*. 2017, pp. 2088–2096 (cit. on p. 10).

[40] P.-S. Wang et al. "O-cnn: Octree-based convolutional neural networks for 3D shape analysis". In: *ACM Transactions On Graphics (TOG)* 36.4 (2017), pp. 1–11 (cit. on pp. 10, 22).

[41] M. Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *kdd*. Vol. 96. 34. 1996, pp. 226–231 (cit. on pp. 15, 50).

[42] L. Landrieu and M. Simonovsky. "Large-scale point cloud semantic segmentation with superpoint graphs". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4558–4567 (cit. on pp. 17, 24).

[43] S. Shi et al. "Pv-rcnn: Point-voxel feature set abstraction for 3D object detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 10529–10538 (cit. on pp. 19, 20, 22).

[44] P. Sun et al. "Scalability in perception for autonomous driving: Waymo open dataset". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2446–2454 (cit. on pp. 20, 50).

[45] A. Dai et al. "Scannet: Richly-annotated 3D reconstructions of indoor scenes". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5828–5839 (cit. on pp. 20, 22).

[46] J. Deng et al. "Voxel r-cnn: Towards high performance voxel-based 3D object detection". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 2. 2021, pp. 1201–1209 (cit. on p. 22).

[47] S. Huang et al. "Spatio-temporal self-supervised representation learning for 3D point clouds". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 6535–6545 (cit. on p. 22).

[48] W. Zheng et al. "SE-SSD: Self-Ensembling Single-Stage Object Detector From Point Cloud". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 14494–14503 (cit. on p. 22).

[49] R. Cheng et al. "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*. 2021, pp. 12547–12556 (cit. on pp. 22, 24, 25, 58, 59).

[50] A. Nekrasov et al. "Mix3D: Out-of-context data augmentation for 3D scenes". In: *2021 International Conference on 3D Vision (3DV)*. IEEE. 2021, pp. 116–125 (cit. on p. 22).

[51] K. Liu et al. "Fg-net: Fast large-scale lidar point clouds understanding network leveraging correlated feature mining and geometric-aware modelling". In: *arXiv preprint arXiv:2012.09439* (2020) (cit. on p. 22).

[52] Z. C. Lipton. "The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery." In: *Queue* 16.3 (2018), pp. 31–57 (cit. on pp. 24, 29).

[53] L.-C. Chen et al. "Encoder-decoder with atrous separable convolution for semantic image segmentation". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 801–818 (cit. on p. 26).

[54] R. Raguram, J.-M. Frahm, and M. Pollefeys. "A comparative analysis of RANSAC techniques leading to adaptive real-time random sample consensus". In: *European conference on computer vision*. Springer. 2008, pp. 500–513 (cit. on p. 27).

[55] R. Caruana et al. "Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission". In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 2015, pp. 1721–1730 (cit. on p. 28).

[56] C. Rudin. "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead". In: *Nature Machine Intelligence* 1.5 (2019), pp. 206–215 (cit. on pp. 28–30).

[57] M. de Sousa Ribeiro and J. Leite. "Aligning artificial neural networks and ontologies towards explainable ai". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 6. 2021, pp. 4932–4940 (cit. on p. 29).

[58] P. Barbiero et al. "Entropy-based logic explanations of neural networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 6. 2022, pp. 6046–6054 (cit. on p. 29).

[59] D. V. Carvalho, E. M. Pereira, and J. S. Cardoso. "Machine learning interpretability: A survey on methods and metrics". In: *Electronics* 8.8 (2019), p. 832 (cit. on p. 29).

[60] Z. Chen, Y. Bei, and C. Rudin. "Concept whitening for interpretable image recognition". In: *Nature Machine Intelligence* 2.12 (2020), pp. 772–782 (cit. on p. 29).

[61] G. Bocchi et al. "GENEOnet: A new machine learning paradigm based on Group Equivariant Non-Expansive Operators. An application to protein pocket detection". In: *arXiv preprint arXiv:2202.00451* (2022) (cit. on pp. 36–38).

[62] T. Cohen and M. Welling. "Group equivariant convolutional networks". In: *International conference on machine learning*. PMLR. 2016, pp. 2990–2999 (cit. on pp. 38, 39).

[63] M. Steininger et al. "Density-based weighting for imbalanced regression". In: *Machine Learning* 110.8 (2021), pp. 2187–2211 (cit. on pp. 46–48).

[64] T. Chaton et al. "Torch-Points3D: A Modular Multi-Task Frameworkfor Reproducible Deep Learning on 3D Point Clouds". In: *2020 International Conference on 3D Vision (3DV)*. IEEE. 2020 (cit. on p. 59).

[65] Q.-Y. Zhou, J. Park, and V. Koltun. "Open3D: A Modern Library for 3D Data Processing". In: *arXiv:1801.09847* (2018) (cit. on p. 59).