Degree Project in Electrical Engineering

30 credits

Maximizing the performance of point cloud 4D panoptic segmentation using AutoML technique

TENG MA

Stockholm, Sweden, 2022

b|

Maximizing the performance of point cloud 4D panoptic segmentation using AutoML technique

TENG MA

Master’s Programme, ICT Innovation, 120 credits
Date: December 8, 2022

Supervisors: Sara Afshar, Sung Ki Won
Examiner: Fredrik Kilander

Host company: Volvo CE
Swedish title: Maximera prestandan för punktmoln 4D panoptisk segmentering med hjälp av AutoML-teknik

Abstract

Environment perception is crucial to autonomous driving. Panoptic segmentation and objects tracking are two challenging tasks, and the combination of both, namely 4D panoptic segmentation draws researchers' attention recently. In this work, we implement 4D panoptic LiDAR segmentation (4D-PLS) on Volvo datasets and provide a pipeline of data preparation, model building and model optimization.
The main contributions of this work include: (1) building the Volvo datasets; (2) adopting an 4D-PLS model improved by Hyperparameter Optimization (HPO). We annotate point cloud data collected from Volvo CE, and take a supervised learning approach by employing a Deep Neural Network (DNN) to extract features from point cloud data. On the basis of the 4D-PLS model, we employ Bayesian Optimization to find the best hyperparameters for our data, and improve the model performance within a small training budget.

Keywords

LiDAR perception, 4D panoptic segmentation, Hyperparameter Optimization, Deep learning, Automated Machine Learning.

Sammanfattning

Miljöuppfattning är avgörande för autonom körning. Panoptisk segmentering och objektspårning är två utmanande uppgifter, och kombinationen av båda, nämligen 4D panoptisk segmentering, har nyligen uppmärksammat forskarna.
I detta arbete implementerar vi 4D-PLS på Volvos datauppsättningar och tillhandahåller en pipeline av dataförberedelse, modellbyggande och modelloptimering.
De huvudsakliga bidragen från detta arbete inkluderar: (1) bygga upp Volvos datauppsättningar; (2) anta en 4D-PLS-modell förbättrad av HPO. Vi kommenterar punktmolndata som samlats in från Volvo CE och använder ett övervakat lärande genom att använda en DNN för att extrahera funktioner från punktmolnsdata. På basis av 4D-PLS-modellen använder vi Bayesian Optimization för att hitta de bästa hyperparametrarna för vår data och förbättra modellens prestanda inom en liten utbildningsbudget.

Nyckelord

LiDAR-uppfattning, 4D-panoptisk segmentering, hyperparameteroptimering, djupinlärning, automatiserad maskininlärning.

Acknowledgments

I would like to thank Mohammad, Sara, Ali, Masoud from Volvo CE for their help and guidance. I would like to thank Ki Won and Fredrik from KTH for their detailed comments on the thesis writing. I also would like to thank my parents and friends for the support they give to me.

Stockholm, December 2022
Teng Ma

Contents

List of Figures

List of Tables

List of acronyms and abbreviations

4D-PLS	4D panoptic LiDAR segmentation
AI	Artificial Intelligence
AutoML	Automated machine learning
CNN	Convolutional Neural Network
DNN	Deep Neural Network
EI	Expected Improvement
GPR	Gaussian Process Regression
HPO	Hyperparameter Optimization
IoU	Intersection over Union
LiDAR	Light Detection And Ranging
LSTQ	LiDAR Segmentation and Tracking Quality
MLP	Multi-layer perception
MOT	Multiple Object Tracking
MOTS	Multi-Object Tracking and Segmentation
NAS	Neural Architecture Search
RADAR	RAdio Detection and Ranging
TPE	Tree-structured Parzen Estimator

Chapter 1

Introduction

Scene understanding is one of the main tasks in the field of autonomous driving. To recognize the surrounding objects and environment, the perception module of the autonomous driving system interprets data collected from sensors such as camera, RAdio Detection and Ranging (RADAR) and Light Detection And Ranging (LiDAR). With more high-quality point cloud data from LiDAR available, many LiDAR-based studies have been conducted on related tasks such as object detection, semantic segmentation, and instance segmentation.

Notably, panoptic segmentation, as a popular scene understanding problem that first emerged in the 2D image domain [1], aims at a holistic solution by unifying semantic and instance segmentation. However, segmentation in 3D point clouds is not a trivial task for the reason that point cloud data are usually noisy, sparse, and unorganized [2]. Furthermore, to interact with the dynamic environments, tracking objects over time is also important for autonomous vehicles. 4D panoptic segmentation [3] merges panoptic segmentation and object tracking, and extends scene understanding to the 4D spatio-temporal domain. Figure 1.1 shows the 4D panoptic segmentation results on a sequence of point cloud scans.

In recent years, Deep Neural Network (DNN) has become the main method of object detection and segmentation problems in both 2D images and 3D point clouds. Designing a satisfactory network for a specific task is usually time-consuming, due to a wide range of parameters that need to be set manually before network training. These parameters are called hyperparameters, and the selection of which is critical to network performance [4].

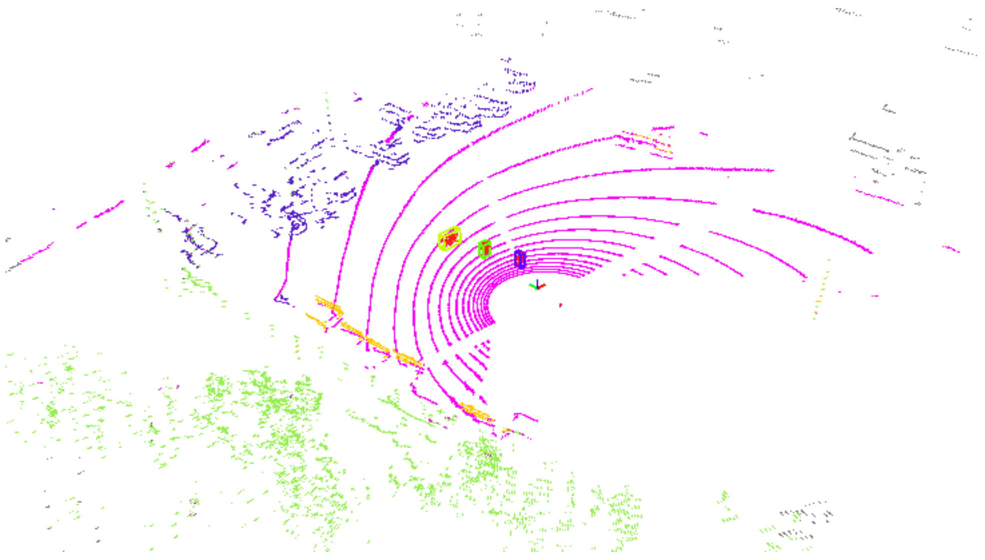


Figure 1.1: 4D panoptic segmentation displayed on 18 consecutive frames of point cloud data from Volvo CE

1.1 Background

Artificial Intelligence (AI) takes us one step closer to autonomous driving. Learning-based methods has accelerated the research process in many fields, in particular the perception module of autonomous systems. Volvo Construction Equipment (Volvo CE) is one of the largest manufacturers of construction equipment in the world. Volvo CE is committed to building safer and smarter construction machines by investing in the research and development of autonomous vehicles for construction purposes [5]. To ensure the safety of people, which is the top priority of autonomous driving, 3D perception models are required to achieve high accuracy and low latency at the same time [6]. It is essential to develop cost-efficient and high-performance perception modules. In the construction domain, a self-driving machine can detect and recognize obstacles, humans, and other moving machines in the surrounding area by continuously interpreting sensory data in 3D space and time. 4D panoptic segmentation is a way to understand temporal semantic scenes by predicting semantics for each point in point clouds, and a unique temporally consistent instance ID for each object.
Deep learning based methods achieve state of the art in point cloud semantic segmentation tasks, while hyperparameters selection of deep neural networks remains a challenging problem. Tuning the network manually or using simple grid search consumes a lot of human effort and time. Automated

machine learning (AutoML) techniques help finding suitable hyperparameters automatically and maximize the model's performance.

1.2 Problem

Reliable environmental perception is a prerequisite for safe operations of automated vehicles. Understanding the semantics of the environments and tracking movable objects are both necessary for subsequent tasks, such as path planning and collision avoidance. Research questions:

- How to realize panoptic segmentation and multiple object tracking using DNN on LiDAR point cloud datasets collected from Volvo?
- How to find the hyperparameters that maximize the performance of the deep neural network?

1.3 Purpose

The purpose of this thesis was to improve the LiDAR perception system of construction vehicles using AI technologies.
This project was conducted in Volvo Construction Equipment, as part of the development of autonomous driving technology in construction scenarios.

1.4 Goal

The goal of this project was to provide a whole pipeline of 4D panoptic segmentation on point cloud data, including data labeling, network training, and hyperparameter optimization. This was divided into the following three sub-goals.

1. Subgoal 1
   Preparing Volvo point cloud dataset, including data collection, data preprocessing and data labeling via open source data annotation tools.
2. Subgoal 2
   Building a deep learning model for 4D panoptic segmentation.
3. Subgoal 3
   Maximizing the performance of the deep neural network by optimizing hyperparameters with AutoML techniques.

Expected result: the trained model gives instance IDs and semantic labels for each point of a point cloud sequence.

1.5 Benefits, Ethics and Sustainability

The improvement of perception ability promotes the development of autonomous driving. More self-driving trucks will be involved in construction activities and help to conduct tasks more efficiently and safely. One ethic issue here is safety. Accurate perception of the environment can reduce the probability of accidents caused by vehicles. Evidence shows that autonomous driving is safer than manually driving [7]. From a sustainability perspective, self-driving vehicles contribute to lower emissions and reduction of energy consumption [8].

1.6 Methodology

With data from Volvo, a data-driven model was adopted in this project. Supervised learning, to be specific, deep learning method was used to achieve the goals. Extensive research [9, 10, 11] shows that deep neural network is a powerful tool for feature extraction, especially for complex tasks such as semantic segmentation where designing handcrafted features for different kinds of objects are extremely hard.
The assumption was that an end-to-end deep neural network could fulfill 4D panoptic segmentation task, and the performance of network could be improved after hyperparameter optimization. Quantitative and qualitative analysis of the obtained results is available, including the visualizations of results and the model performance measured by some evaluation metrics.

1.7 Delimitations

This thesis work focuses on 4D panoptic segmentation of point clouds. Due to the limited time of this degree project, the deep learning model structure will be built on the basis of the first 4D panoptic LiDAR segmentation (4D-PLS) model [3], and more efforts are put in model adaptation and optimization on Volvo point cloud dataset.

1.8 Structure of the thesis

In chapter 2, theoretical background knowledge is introduced including basic concepts and concise summary of related work. In chapter 3, the methods used in this degree project are described in detail. Experiments results are clearly presented and compared in chapter 4. In chapter 5, conclusions are drawn by reviewing the whole project, followed by the future work.

Chapter 2

Theoretical Background

This chapter provides basic background information about point cloud data, point cloud segmentation, multiple object tracking, and automated machine learning. Additionally, this chapter describes related works in point cloud based scene understanding and object tracking, as well as hyperparameter optimization.

2.1 An Introduction to Point Cloud Data

Point cloud data contains information of a set of points in three-dimensional space. Each point is represented as a vector of its 3D Cartesian coordinates (x, y, z), plus other information such as color (RGB), reflection intensity (Intensity) etc. Three main properties of the points are stated as follows [12]:

- Being unordered, which means that rearranging the input order of all points causes no change to a point cloud.
- There are correlations between points that reveal the features of local structures.
- Invariance under transformations. The property of a point cloud should not be changed after certain transformations (e.g. rotating and translating) applied to the whole point cloud.

Point cloud data can be acquired from LiDAR, RADAR, RGB-D cameras, or derived from images using multi-view stereo vision algorithms [13].
With high precision LiDAR point cloud data available, high resolution LiDARs have been deployed on autonomous vehicles as a part of the perception system. One advantage of LiDAR compared to cameras is the

robustness to light changes in the environments, for that it detects ranges by actively emitting laser pulses. However, due to the large range of outdoor scenes, the point cloud data from LiDAR mounted on the car is usually sparse, which leads to a loss of some information. Figure 2.1 is one scan from LiDAR, which shows the sparsity of far points.

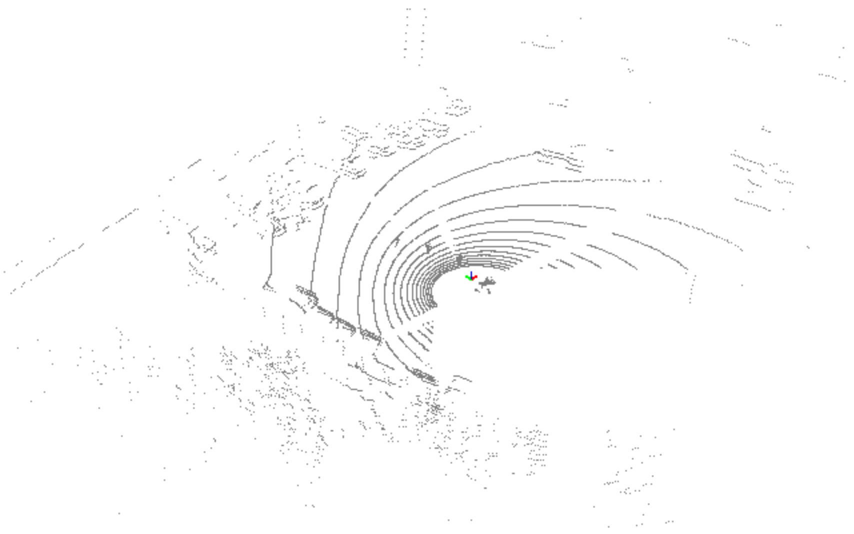
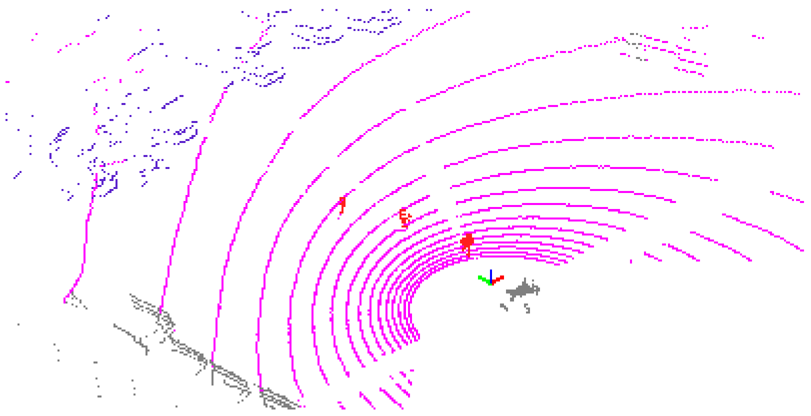


Figure 2.1: LiDAR point cloud data collected by Volvo CE

2.2 Point Cloud Segmentation

Segmentation of point clouds has decades of history [14]. Before effective supervised learning methods were widely used, point cloud segmentation was to cluster points with similar characteristics into homogeneous regions. Inspired by intensive research in computer graphics and computer vision, many approaches were proposed to segment point cloud data. [2] summarizes methods for point cloud segmentation in earlier years.
Instead of only clustering points, researchers took a step further and tried to infer a semantic label for every point. As a result, assigning each 3D point a semantic label, i.e. **semantic segmentation** has become the new target in point cloud segmentation. After many explorations in supervised machine learning methods including Support Vector Machine [15], Gaussian Mixture Model [16], Random Forest [17] and statistical contextual models, such as Conditional Random Field [18], deep learning based methods now achieve the state of the art for point cloud segmentation.

In the realm of robotics, information at the instance level is crucial for robots/self-driving cars to interact with the environment and make right decisions [3]. **Instance segmentation** is a technique to realize detection and semantic segmentation of objects at the same time, by giving different labels for separate instances belonging to the same class [19]. On the basis of semantic segmentation and instance segmentation, **panoptic segmentation** as a popular scene understanding problem that emerged recently, is to predict semantics of foreground countable objects such as people, animals, tools, and amorphous regions or background objects such as grass, sky, road [1]. For point cloud data, the task of panoptic segmentation is to assign each point a semantic label and instance ID if the point belongs to an object (see Figure 2.2).


(a) Semantic segmentation

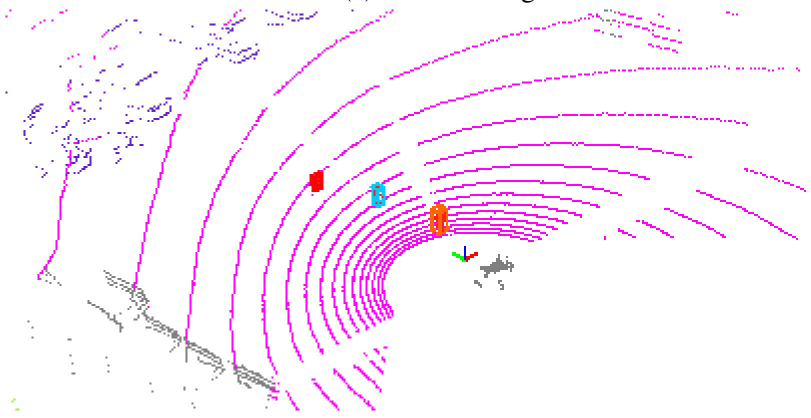

(b) Panoptic segmentation

Figure 2.2: Comparison between Semantic segmentation (a) and Panoptic segmentation (b). (b) contains three different bounding boxes for instances of people (red points)

2.3 Deep Learning in Point Cloud Data

As a deep neural network was adopted in the project, this part briefly reviews deep learning, and elaborate how point cloud data is represented in deep neural networks.

2.3.1 An Introduction to Deep Learning

Problems that can be described formally are easy to solve by programming following a series of mathematical rules. It turns out that the real challenge for AI is solving tasks that are easy for people to perform but hard for people to formally describe [20]. Deep learning allows computers to solve intuitive problems, such as recognizing spoken words or faces in images, by building computational models of multiple process layers and learning from data.
Facilitated by the improvement of computation power and a large amount of data available, deep learning has achieved the state-of-the-art in a variety of tasks in computer vision and natural language processing. Particularly, deep convolutional neural networks have fueled great strides in 2D image processing tasks, such as image classification [9], semantic segmentation [21] and object detection [10]. The tremendous success of Convolutional Neural Network (CNN) in the 2D image domain promoted the development of 3D CNN modules for point cloud data.

2.3.2 Point Cloud Data Representation

However, deep neural networks designed for 2D raster images cannot be directly applied to point cloud segmentation, due to the fact that point cloud data is disordered and unstructured [14]. Based on the way data is represented, existing methods can be broadly divided into three categories: point-based, voxel-based, and projection-based methods.
Point-based methods like Pointnet series [12, 22] process point cloud data directly and keep the original information preserved to the greatest extent. The problem is that point-wise operation requires large memory space and a lot of inference time, especially applied to outdoor scenes.
In voxel-based methods, a point cloud is first transformed into regular 3D volumetric grids and then processed in 3D convolutions [23]. Considering that vanilla 3D convolution is also both memory and computationally intensive, many new approaches like Cylinder3D [24] emerged.
Projection-based methods project point clouds onto a regular 2D grid

representation using sphere project [25] or scan unfolding [26], in order to use well researched 2D convolution architectures. Bird's eye view is also an option of projection [27].
Recently, some methods fuse two or more different views together, for example, fusing bird's eye view and range images, or point and voxel together. More details are introduced as related work in section 2.6. In another direction, graph convolution networks emphasize edge relationships instead of points relative positions. EdgeConv [28] constructs a directed graph using k-nearest neighbor, combines global and local features by integrating point position and the relative distances of surrounding points into feature encodings.

2.4 Multiple Object Tracking

Multiple Object Tracking (MOT) is another well researched topic in the field of computer vision. It relies on objects detection results from instance segmentation, and grounds tasks such as pose estimation, behavior analysis and understanding that play important roles in human computer interaction, virtual/augmented reality, surveillance and autonomous driving. The tasks of MOT are mainly divided into locating multiple objects, maintaining their identities, and generating their respective trajectories in a continuous sequence of frames [29]. Compared with single object tracking, MOT faces a much more complicated situation where multiple objects in one category could be entangled. Similarly, deep learning based MOT methods come to the top in the public benchmark test [29, 30].

2.5 Automated Machine Learning

Machine learning, especially deep learning methods have brought a revolution in computer vision, natural language processing and many other application domains. However, many machine learning models perform well only with the proper selections of network architecture, training procedures, regularization methods, and other hyperparameters in the model. Making a series of right decisions is repetitive and tedious, and it takes a lot of time and energy for engineers, even experts sometimes, let alone people who want to adopt machine learning methods but do not have the resources to learn about the technologies behind it in detail. AutoML aims to make these decisions in a data-driven, objective, and automated way: the user simply provides data, and the AutoML system automatically chooses the approach that performs best for this particular application [31]. An AutoML framework aims to automate the whole pipeline of machine learning including data preparation, feature selection, model generation and evaluation, so that researchers can pay more attention to the problem itself. Figure 2.3 give a general grasp of what an AutoML framework can do.

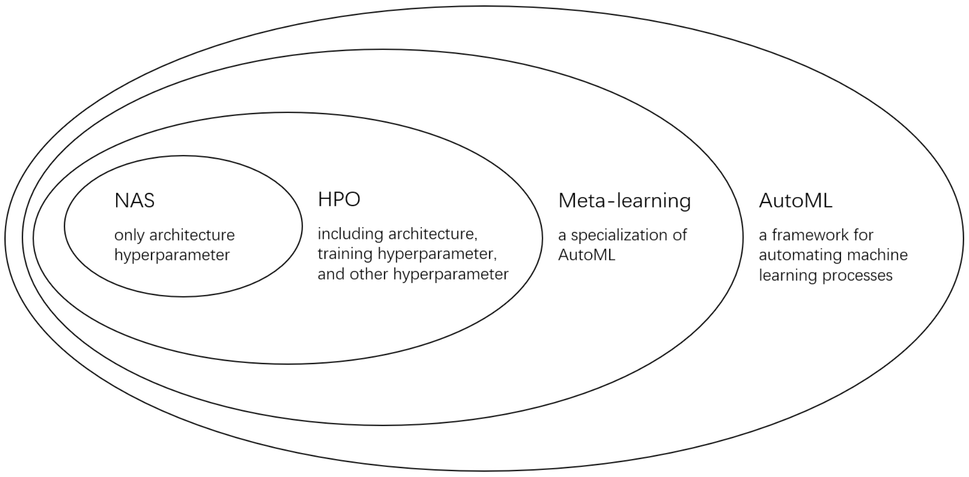

NAS
only architecture hyperparameter
HPO
including architecture, training hyperparameter, and other hyperparameter
Meta-learning
a specialization of AutoML
AutoML
a framework for automating machine learning processes

Figure 2.3: An overview of AutoML

2.5.1 Hyperparameter Optimization

The most common problem that AutoML considers is Hyperparameter Optimization (HPO). The goal of HPO is find the hyperparameters that maximize or minimize the objective function. It is usually a non-convex optimization problem for a machine learning model As machine learning models become more and more complex, hyperparameters could be continuous, discrete, categorical, and conditional. Tuning hyperparameters automatically not only reduces human effort, but also improves the performance of machine learning models and makes these models more reproducible. In particular, key hyperparameters of deep learning models can be roughly divided into three categories [32]:

- Architecture design hyperparameters, e.g. the number of hidden layers and the number of neurons.
- Training settings, e.g. learning rate, drop-out rate, batch size and the number of epochs.

- Function types, e.g. loss function type, optimizer type.

Due to the limited time and computation resources, this project focus on HPO using AutoML techniques.

2.5.2 Meta-learning

Meta-learning, or learning to learn can be seen as a subset of AutoML. Meta-learning is proposed to promote machine learning methods in tasks where the data is rare and computation resources are insufficient. Considering that our brain can neither store huge amount of data nor conduct complex computation compared to computer, we perform much better in many learning tasks than computers. One of the reasons is that we human learn from experiences. Meta-learning is a data-driven process of improving upcoming learning performance by distilling the prior knowledge from experience [33].

2.5.3 Neural Architecture Search

Recently Neural Architecture Search (NAS) attracts a lot of attention due to the wide application of deep learning since manually designing neural architectures is time-consuming and error-prone [34]. Searching a neural architecture follows this pattern: first, define the search space, then find a search strategy to select an architecture configuration and evaluate the performance of the searched network, which is returned to the search strategy for the next searching loop. It is one of the most challenging tasks for that the design space is extremely large and a single evaluation of a neural network can take a very long time.

2.6 Related Work

This section briefly reviews previous deep learning based methods in point cloud semantic segmentation, LiDAR-based object detection/segmentation, and tracking, as well as some well-established HPO algorithms and frameworks.

2.6.1 Point cloud Semantic Segmentation

Semantic segmentation of 3D point clouds is to infer the label of each three-dimensional point. As mentioned in 2.3.2, previous mainstream deep learning methods employed on point cloud data can be classified as point-based, voxel-based, projection-based, or fusion methods which combine 2 or 3 data representations together.

Point-based networks can be further subdivided into point-wise Multi-layer perception (MLP) networks and point-wise convolution networks where convolution kernels for points are explicitly defined. Pointnet [12] first proposed a network structure that can directly process point cloud data. MLP is used on every point followed by a global max-pooling. The shared MLP encodes points and max-pooling captures the global information. The performance of Pointnet is limited since no local spatial relationships are learned. To make up for this deficiency, Pointnet++ [22] introduced set abstraction as an imitation of CNN block to learn the local information. MLP is a powerful feature extractor, yet it makes networks hard to converge and increases the complexity after combining point convolutions.

To get rid of the intermediate representation of an MLP, KPConv is an explicit convolution kernel where the weights carried by points are directly learned. Kernel points with a learnable weight matrix are regularly placed in a sphere space, correlated with the distance between neighbors and the point where KPConv is applied to. A deformable version of KPConv inspired by [35] is also presented, in this case the position of kernel points are also learned from the network. KPConv outperforms other methods with well-designed point convolution kernels such as [36, 37].

For large outdoor scenes, point-based methods cannot run in real time due to computational and memory limitations. In exchange for less processing time, RandLA-Net [38] uses random sampling with a local feature aggregation, which is a compensation for the information loss brought by random operation by progressively increasing the receptive field for each 3D point.

Voxel-based methods kicked off with these early ideas [23, 39, 40] that intuitively generalized 2d convolution operation to 3D space. Researchers transformed the point clouds into voxels and applied vanilla 3D convolutions, which cost more memory and computation resources than 2D convolutions. Early volumetric methods lower the resolution of voxels as a compromise of the resource deficiency, and suffer serious information loss and act poorly in outdoor scenes. More recently, methods [41, 42] built on sparse 3D convolutions accelerate the 3D convolution operation by utilizing the sparsity of points. Cylinder3D [43] partitions point clouds in the form of cylindrical grids that adapt to the density difference of points in space, and reaches decent performance in a different way within acceptable runtime and memory consumption. As the state-of-the-art of voxel based methods, AF2S3Net adds

two novel attention blocks named Attentive Feature Fusion Module(AF2M) and Adaptive Feature Selection Module(AFSM) to the baseline model of MinkNet42 [41], and optimize the network with a mixed loss function.
Projection-based methods project point clouds onto 2D images and adopt 2D CNN for semantic segmentation, then project the predictions back to 3D point clouds. Most existed methods project the point cloud into range images [44, 25, 45] or bird's eye views [27]. SqueezeSeg [44] and RangeNet++ [25] use spherical projection to obtain the intermediate range images, while [26] advocates scan unfolding that yields better results on ego-motion corrected data. Salsanext [45], the next version of Salsanet [46] with an improved encoder-decoder structure and loss function, is the most successful attempt among projection-based methods so far.
Multi-view Fusion aims to leverage information from different views and overcome the shortcomings in single view methods: Voxel-based methods suffer information loss caused by the low resolution of voxels. Point-based methods spend too much runtime on structuring the irregular point cloud data [47]. Projection-based methods ignore depth information that reveals 3D geometric features of points. Multi-view fusion is a promising direction that could abandon the drawbacks in single view methods. FusionNet [48] aggregates voxel features by sparse convolution layers and point features from the voxel-based "mini-PointNet", which saves a lot of time finding neighbours for points. SPVCNN [6] replace the voxel convolution in PVCNN [47] with sparse convolution, further reduces the runtime and achieves relatively high accuracy on the large-scale point cloud. RPVNet [49] is a deep framework that allows interactions between range images, points and voxels during training, using an RPV interaction mechanism in a multi-view fusion.

2.6.2 Multi-Object Tracking and Segmentation

Instance segmentation replaces the masks (bounding boxes) in object detection with finer ones (pixels), in order to obtain pixel level detection with semantic meanings. Many successful methods [11, 50, 51] follow a top-down scheme: first, locate the bounding box of each instance, and then perform semantic segmentation inside the box to obtain the mask of each instance; Other methods [52, 53] take a bottom-up approach by grouping pixels belonging to the same instance.
As the accuracy of instance segmentation increases, MOT is under a transformation from coarse bounding box level tracking to fine pixel level

tracking. Multi-Object Tracking and Segmentation (MOTS) considers instance segmentation and MOT together, overcomes the ambiguities caused by the bounding box representation and thus gain further improvements [54]. A tracking-by-detection paradigm is adopted in many MOT methods, the idea of which is to first detect objects independently in each frame and then associate detection results across time. Such models are usually computationally demanding due to employing different networks for object detection and data association separately. Recently, proposed an end-to-end bottom-up approach was proposed by modeling a video clip as a 3D spatio-temporal volume, achieved good performance with less computation [55].

2.6.3 4D Panoptic Segmentation

4D Panoptic Segmentation was first described in [3] as a beginning of temporal LiDAR panoptic perception, which supplements background semantic information for multi-object tracking and segmentation on LiDAR point cloud sequences. The model operates in a 4D spatio-temporal volume which is formed by merging multiple consecutive point clouds together, predicts semantic labels of points and parameter maps used for cluster points into instances. The authors chose KPConv [56] as the only encoder network and trained the model in an end to end fashion. A new evaluation metric, LiDAR Segmentation and Tracking Quality (LSTQ) metric is provided in the paper.
Another direction is following the tracking-by-detection paradigm, namely associate instances detected in panoptic segmentation across time. An instance association approach inspired by contrastive learning techniques was proposed lately [57]. The authors employed an off the shelf panoptic segmentation methods, DS-Net, as the panoptic backbone, followed by a well-designed contrastive aggregation network (CA-Net) which pushes encodings of the same instance in different frames lie close together, meanwhile far from encodings belonging to other instances. In addition to LSTQ metric, mIoU and mAQ are also commonly used metrics for evaluation of 4D Panoptic Segmentation.

2.6.4 Hyperparameter Optimization Methods

Due to the large range of hyperparameter configurations, simple methods such as greedy search and babysitting (manually tuning) are not feasible within certain constraints/budgets. Gradient-based methods are rarely seen in HPO, for that the gradient of the loss function with respect to hyperparameters are available only if the hyperparameters are continuous.

Grid Search is a basic HPO method with a clear mathematical expression. Grid search evaluates the Cartesian product of a finite set of hyperparameters, the values of which are defined by users. Because the computation consumed grows exponentially when the number of hyperparameters increase, grid search suffers from the curse of dimensionality. Another issue with grid search is that increasing the resolution of discretization also significantly increases the computation required. For the reasons mentioned above, to work with grid search, users have to narrow down the search space according to their professional knowledge in advance.

Random Search is a simple and improved alternative to grid search [58]. As the name implies, random search hyperparameter configurations at random until the search budget is depleted. Different from grid search where the budget for evaluating a configuration is fixed, random search can assign budgets independently. This brings significant advantages when some hyperparameters are much more important than others. Since grid search arrange each trial asynchronously and there is no communication between works, it allows for easier parallelization. Another benefit of random search is its flexibility of resource allocation. It can be easily extended with additional samples; in contrast, the number of points on a grid must be determined beforehand [59].

Although random search is more effective than grid search in most cases, it is still a computationally intensive method. In the early stages of HPO, random search is recommended to rapidly narrow the search space before using a guided algorithm to obtain a finer result. Random search is frequently used as the baseline of HPO to assess the effectiveness of newly designed algorithms. In general, random search consumes more time and computational resources than other guided search methods.

Bayesian optimization is a traditional global optimization approach that achieves state-of-the-art results in tuning hyperparameters of deep neural networks. Bayesian optimization methods search for global optimum by analysing past results. A probabilistic surrogate model of the objective function is employed to fit all currently-observed searching results, followed by an acquisition function that chooses points to be evaluated next iteration. BO models balance the exploration (expand search to areas that are never sampled from before) and exploitation (dig in the promising regions based on current information) processes in order to find the most likely optimal locations while avoiding missing better configurations in unknown areas [60]. Guided by past trials, on the one hand, Bayesian optimization models reach the desired result faster, on the other hand, they are more difficult to parallelize due to the sequential historical data. Gaussian process (GP) [61], random forest (RF) [62], and the tree-structure Parzen estimator (TPE) [63] are common choices for surrogate models. Gaussian process regression works well on continuous functions and hyperparameters, yet many hyperparameters are categorical in a DNN. As a result, BO-RF and BO-TPE are more suitable for tuning deep neural networks.

Population-based methods utilize metaheuristic algorithms such as genetic algorithms, evolutionary algorithms, particle swarm optimization to find global optimum by generating and iterating over a population, i.e., a set of configurations. These methods can be easily parallelized since theoretically the evaluation of N individuals in a population can be allocated to N threads at most.

Nowadays, training a DNN with a single hyperparameter set on large datasets can easily take several hours and up to several days. One major problem to be addressed when applying HPO to DNN is to shorten the execution time. One common approach to speed up the optimization process is reducing the datasets and training the model in a low-fidelity manner, for example, by using a subset or training with fewer iterations.

Multi-fidelity Optimization is an excellent technique to save time. One basic idea here is not evaluating every hyperparameter configuration after training with full iterations or whole datasets, since quite a few configurations are far from the optimum and can be dropped during searching. Early-stopping based on the learning curve is a well-known way to filter configurations. Another way is using bandit-based Algorithms such as successive halving [64], which eliminates half of the poorly-performing hyperparameter configurations each iteration. BOHB [65] Bayesian optimization and Hyperband [66] (an improved version of successive halving) achieves state-of-the-art in HPO.

2.6.5 Hyperparameter Optimization Frameworks

There are many packages and frameworks for HPO, only ones that are compatible with Pytorch will be listed in this part. Optuna [67] is an open-source HPO framework that allows dynamically constructing the search space. It implements both searching and pruning strategies efficiently and supports parallel deployment on GPU. Tune [68] is a library of Ray from Berkeley RISELab. It is also a scalable framework that enables the easy reproduction and integration of a wide variety of aforementioned hyperparameter search algorithms. Auto-Pytorch [69] pursues fully automated deep learning, which includes data preparation, model selection and hyperparameter optimization. It combines multi-fidelity optimization with portfolio construction for warmstarting and ensembling of DNN, equipped with SMAC3 [70] optimizer. Auto-Pytorch provides a standard pipeline, in which all of the modules are tightly coupled, thus it doesn't support tasks with customized datasets and network.

Chapter 3

Methods

This chapter first describes the methods for the 4D-PLS task, which is to predict for each 3D point in a given sequence of LiDAR scans (1) a semantic label, and (2) a unique, persisted object instance ID. Then the HPO methods used in this degree project follow up. In addition, this chapter summarizes my efforts in building the Volvo dataset. Model evaluation and experiment setup are at the end of this chapter. The Data is collected by the team I was in from Volvo.

3.1 4D Panoptic LiDAR Segmentation Model

We adopt the 4D-PLS model proposed in [3]. It tackles point cloud panoptic segmentation by jointly clustering points that belong to instances and interpreting semantic meanings of points in the 4D continuum. Figure 3.1 gives an overview of 4D-PLS. First, several consecutive point clouds are aligned to form a 4D volume, the input of an encoder-decoder network. The outputs include an objectness map O that contains potential object centers, a semantic map S that reveals point-level semantics, an embedding map ϵ that gives the learned embeddings of points, and a variance map Σ that is used in clustering.

3.1.1 Network Backbone

The encoder-decoder network comes from KPFCNN [56], a fully convolutional network for segmentation. The encoder part contains 6 layers, each of which consists of 1 strided convolutional block followed by 2 regular convolutional blocks (except for the first layer with 2 regular blocks).

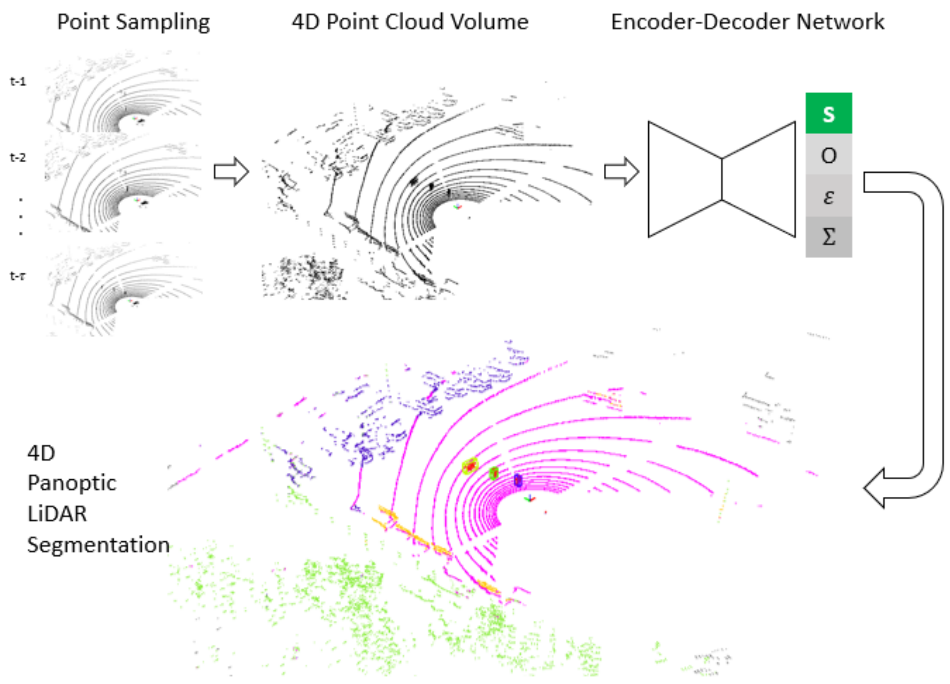
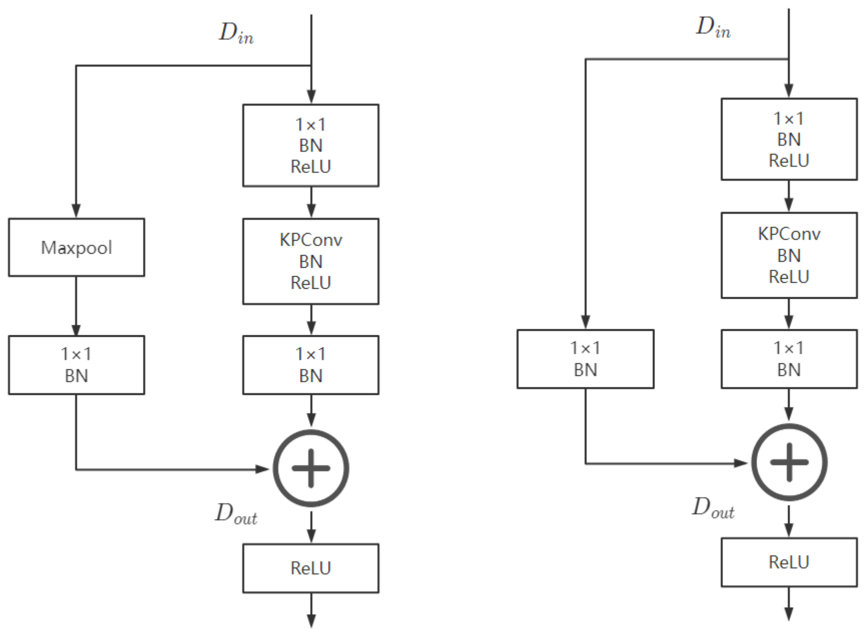


Figure 3.1: Overall Structure of the 4D-PLS model. First merge multiple frames to generate a 4D volume, which is then processed by an encoder-decoder network with 4 outputs. The semantics are predicted by the semantic head S, object instances are identified and tracked by integrating the object center head O, the variance head Σ, and the point embeddings head ϵ.

Following the design of bottleneck ResNet blocks [9], a convolutional block is formed by a rigid KPConv, a batch normalization and a leaky ReLU activation. Figure 3.2 illustrates the strided and regular convolutional block structures.
The decoder part contains 5 layers, each layer is a combination of a nearest upsampling block and a unary block (used to adjust feature dimensions). Features in encoder blocks are concatenated to the upsampled blocks (with shortcuts). The input of the network are N three dimensional points (to guide the KPConv) and two dimensional features, and the output of the decoder includes a $N \times C$ semantic matrix, a $N \times D_e$ point embedding matrix, a $N \times 1$ object center matrix and a $N \times D_v$ variance matrix. Figure 3.3 describes the network architecture.



(a) Strided KPConv block (b) Regular KPConv block

Figure 3.2: KPConv blocks. The Batch normalization in the shortcut is to adjust input feature dimension, only used when $D_{in} \neq D_{out}$.

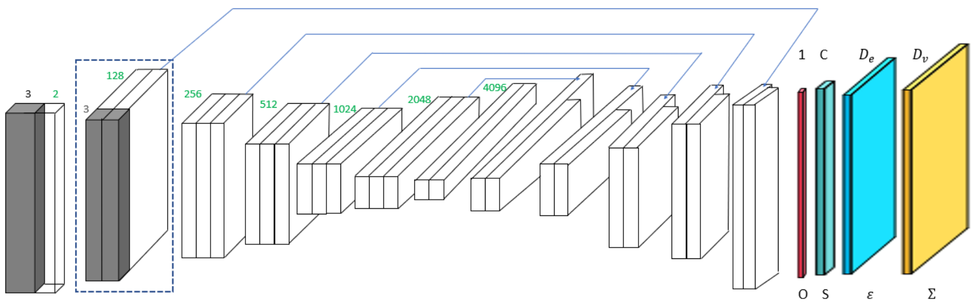


Figure 3.3: Overall Structure of the KPFCNN model. Input N points (x, y, z) concatenated with fearures (f1, f2), the encoder part uses KPConv blocks for feature extraction and downsampling, the feature map dimension is shown in green near the corresponding layers. The decoder part upsamples from the previous layer and generates 4 output matrixs with height N and width labeled above the blocks respectively. The points are not displayed after the first layer.

3.1.2 Instance Representation

Object instances are segmented by a bottom-up approach, which groups points based on probability density. Points are assigned to their respective instance by evaluating each point under the Gaussian probability density function based on the point's embedding vectors. The embedding vectors are concatenations of the 4D point coordinates (x, y, z, t) and learned point features, to take both the spatial and temporal information into account.
Specifically, given an instance center p_i with its embedding vector e_i, a query point p_j with its embedding vector e_j, and a diagonal matrix Σ_i, the probability of a point p_j belonging to the center point p_i is modeled as:

$$\hat{p_{ij}} = \frac{1}{\sqrt{(2\pi)^D |\Sigma_i|}} exp(-\frac{1}{2}(e_i - e_j)^T \Sigma_i^{-1}(e_i - e_j)) \tag{3.1}$$

Here the dimension of point's embeddings and variance matrix is denoted as D. Note that the Gaussian assumption is only valid for short-term prediction.

3.1.3 Network Training

To train the network in an end-to-end manner, the loss function is a combination of four terms:

$$L = L_{class} + L_{obj} + L_{ins} + L_{var} \tag{3.2}$$

Semantic Loss L_{class}: Cross-entropy loss.
Object center Loss L_{obj}: Objectness is introduced to predict the proximity of the point to its instance center. For point p_i, its objectness o_i is Euclidean distance between p_i and its instance center normalized to [0, 1].
Instance loss L_{ins}: As shown in equation (3.1), $\hat{p_{ij}}$ measures the probability of point p_j assigned to center p_i based on point embeddings. Training with L_{ins} promotes instance segmentation in a 4D volume by maximizing $\hat{p_{ij}}$ when p_j belongs to p_i and vice versa.
Variance smoothness loss L_{var}: To ensure the consistency of variance values for every object instance, L_{var} [71, 55] is applied to instance points to minimize the variance of the variances output.
To make the network converge faster while obtaining a decent result, we first pre-train the network with semantic loss only, then train the network with all of the four losses.

3.1.4 Inference

The inference of instances takes two steps. First, points with high objectness scores are selected as seeds of clusters, and points with high probabilities are assigned for each seed to form an instance prediction within a 4D volume. Then the cross-volume instance association in a sequence is conducted based on the overlap score.

3.2 Hyperparameter Optimization

After model establishment, tuning hyperparameters is an indispensable step to unleash the full potential of the model. HPO is a process of finding a hyperparameter set that yields the best model performance on a validation dataset. In practice, to address the problem of HPO, three steps should be taken first:

1. Define the search space, i.e. pick hyperparameters to be optimized and give the range of values for each hyperparameter;
2. Select the evaluation metrics;
3. Find a search strategy to guide the searching process and evaluate the candidate configurations.

Before introducing the methods in detail, a mathematical description of HPO is given.

3.2.1 Problem Statement

Assume there are N hyperparameters to be tuned in the DNN, the domain of the i-th hyperparameter is denoted as D_i. A hyperparameter configuration is a combination of all the hyperparameters, each of which a certain value is assigned to. The whole configuration space is denoted as $D = D_1 \times D_2 \times ...D_N$, a configuration is then represented as a N dimentional vector $x \in D$. A model is first determined by a configuration, then trained on training set. The goal is to find a hyperparameter configuration x_{target} that minimizes the loss L of the corresponded trained model evaluated on validation set:

$$x_{target} = argmin_{x \in D} L(x) \tag{3.3}$$

Note: Loss function L is one of the most commonly used evaluation function f. In general, f refers to any user-defined metrics applied to validating the model.

3.2.2 Search Space

I study 5 training hyperparameters in total: batch size, learning rate, momentum, weight decay and learning rate scheduler. Table 3.1 lists the range, type, and baseline setting of these hyperparameters.

Hyperparameter	Baseline setting	Range	Type
Batch size	500	[100, 500], step = 50	Integer
Learning rate	1e-2	[1e-4, 1e-2]	Float
Momentum	0.98	[0.1, 0.99]	Float
Weight decay	1e-3	[1e-5, 1e-2]	Float
Lr scheduler	ExponentialLR*	[StepLR, ExponentialLR, CosineAnnealingWarmRestarts, ReduceLROnPlateau]	Categorical

Table 3.1: Hyperparameters to be tuned. * The learning rate in baseline model decays by $(0.1)^{\frac{1}{200}}$ every epoch from 0.01, namely tenfold shrinks every 200 training epochs.

3.2.3 Bayesian Hyperparameter Optimization

Since evaluating a hyperparameter configuration in the objective function f is expensive (takes several hours), Bayesian optimization is a perfect candidate for search strategy.

Sequential model-based optimization methods [62], a formalization of Bayesian optimization, only evaluates the most promising hyperparameter configuration x^* that maximizes the acquisition function S that employs a surrogate model M. M models $y = f(x)$ with some probability distributions, the parameters of which are easy to update and approach the true distribution as the observations $H = \{(x_1, y_1), (x_2, y_2), ..., (x_i, y_i)\}$ increases, which makes S much easier and cheaper to optimize than the objective function f. T is the number of evaluations and subjected to resource or time constraints.

The pseudo-code of the model-based optimization method is summarised in Algorithm 1.
Unlike methods that are uninformed by past evaluations (e.g. grid search, random search), Bayesian hyperparameter optimization makes fewer calls to the objective function with the cost of a little more time on hyperparameters selection (neglectable compared to the time spent on model evaluation), overall accelerates the HPO process significantly.

Algorithm 1 Model-based hyperparameter optimization

```
1: Input: objective function $f$, configuration domain $D$, acquisition
   function $S$, surrogate model $M$
2: Initialization: $t = |H|, H = (x_1, y_1), ..., (x_i, y_i)$
3: repeat
4:     Fit a new model $M_t$ to $H$
5:     $p(y|x, H) \leftarrow M_t$
6:     $x^* \leftarrow argmax_{x \in D} S(x, p(y|x, H))$
7:     $H \leftarrow H \cup (x^*, f(x^*))$
8:     $t \leftarrow t + 1$
9: until $t = T$
10: return $H$
```

The optimization criterion adopted to propose x^* (step 6), i.e. the acquisition function is Expected Improvement. I employ Tree-structured Parzen Estimator (TPE) algorithm to construct the surrogate model and fit a model to the observations H (step 4) [63].
Tree-structured Parzen Estimator is a model strategy for high dimentional HPO problems with small evaluation budgets. TPE works well with all types of hyperparameters and outperforms the most popular Gaussian Process Regression (GPR) according to [63]. Different from GPR that directly model $p(y|x)$, TPE models $p(x|y)$ and $p(y)$.
Strictly speaking I use Parzen-window density estimators since there is no tree structure in the configuration space. The main idea is that each configuration defines Gaussian distribution with a specific mean and variance, the overall probability density function of $y = f(x)$ is a mixture of these Guassians. The interesting part of TPE is that $p(x|y)$ is defined by two density functions:

$$p(x|y) = \begin{cases} l(x) & \text{if } y < y^* \\ g(x) & \text{if } y \geq y^*, \end{cases} \tag{3.4}$$

where $l(x)$ means the density of The observation set H is divided into two groups based on y. $l(x)$ is the density formed by good configurations which yields evaluations smaller than y^* (the smaller the better), and $g(x)$ is the density formed by the rest configurations. The criterion y^* is decided by a user-defined parameter $\gamma = P(y < y^*)$. The TPE approach finds the x^* by maximizing the ratio $l(x)/g(x)$, which means that x^* is more likely from good configurations and less likely from the bad ones.
Expected Improvement is a common choice of acquisition function. Given x, $EI(x)$ is the average decrease relative to the threshold y^*. By maximizing Expected Improvement (EI), a configuration that is most likely to achieve smaller y than y^* would be obtained:

$$EI_{y^*}(x) = \int_{-\infty}^{+\infty} max(y^* - y, 0)p(y|x)dy \tag{3.5}$$

The deviation below proves that maximizing $EI(x)$ is equivalent to maximizing $\frac{l(x)}{g(x)}$. According to Bayes' theorem:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} \tag{3.6}$$

According to the Law of total probability and substitute γ, $p(x)$ can be written as:

$$\begin{aligned} p(x) =& \int_{-\infty}^{+\infty} p(x|y)p(y)dy \\ =& \int_{-\infty}^{y^*} p(x|y)p(y)dy + \int_{y^*}^{+\infty} p(x|y)p(y)dy \\ =& \gamma l(x) + (1-\gamma)g(x) \end{aligned} \tag{3.7}$$

The $EI(x)$ given y^* can be written as:

$$\begin{aligned} EI_{y^*}(x) =& \int_{-\infty}^{y^*} (y^* - y)\frac{p(x|y)p(y)}{p(x)}dy \\ =& \int_{-\infty}^{y^*} (y^* - y)\frac{l(x)p(y)}{\gamma l(x) + (1-\gamma)g(x)}dy \\ =& \frac{\int_{-\infty}^{y^*}(y^* - y)p(y)dy}{\gamma + (1-\gamma)\frac{g(x)}{l(x)}} \propto \frac{1}{\gamma + (1-\gamma)\frac{g(x)}{l(x)}} \end{aligned} \tag{3.8}$$

where $EI_{y^*}(x)$ only depends on $g(x)$, $l(x)$. To maximize $EI(x)$, x that maximizes the ratio $\frac{l(x)}{g(x)}$ should be chosen as the configuration to be evaluated next in f.

3.3 Dataset Preparation

3.3.1 Volvo Datasets

We mount the LiDAR on the front head of a truck, and collect 3 datasets in 3 scenarios. Two datasets are from stationary scenes, where the truck/LiDAR stays and people walk around. One dataset comes from a dynamic scene, where both the LiDAR and human are moving.
To carry out our work, my teammate Wangkang and I annotate 2 datasets collected from stationary scenes, in which only people are regarded as objects and others as background. In this project, we only use labeled Volvo datasets for training and testing the model.

3.3.2 Point Cloud Annotation

While building a Volvo point cloud dataset for supervised learning tasks, point cloud data annotation is the most tedious and labor-intensive job. The first and foremost step is choosing the right annotation tool, which could shorten the labeling process by days or weeks. There are many open source tools for point cloud data annotation, such as LATTE [72] and LabelCloud [73]. The ideal tool would be able to label objects and background at the same time. Here are three tools that I tried to label data we collected.
SUSTechPOINTS [74] is a portable point cloud annotation tool with an annotation transfer function to label the same objects in different data frames, which allows high speed instances labeling. Unfortunately it does not provide functions for annotating the background. Hitachi semantic segmentation editor is a web-based labeling tool for creating AI training data sets. A bounding box surrounds all the points by default, which requires eliminating distant noise points of the input point clouds for visualization.
Kitti point cloud labeling tool [75] is the official annotation tool provided by SemanticKitti. The input and output format strictly follows the official data format of SemanticKitti datasets. It supports labeling multiple frames at once, which could save a lot of time, particularly for annotating static scenes.
Table 3.2 summarizes the comparison between these tools. According to the requirement of this project, which is labeling both the semantics of points and instances IDs, we choose the Kitti labeling tool.

Tools	Pros	Cons
SUSTechPOINTS	Batch automated editing	Only for instance
Hitachi editor	Well-designed UI	No batch operations
SemanticKitti tool	Multiple functions	Output in Kitti format

Table 3.2: Comparison between labeling tools

3.3.3 Data Preprocessing

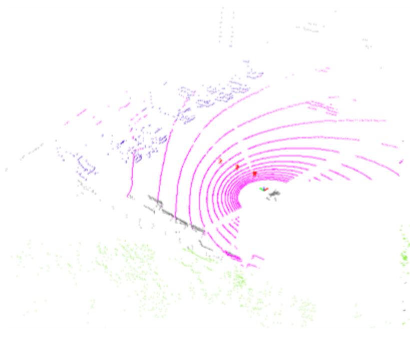
Point cloud data obtained from LiDAR cannot be passed to the model directly, due to hardware limitations. On the one hand, the data is affected by noise (from sunlight, I assume). Raw point cloud data contains outlier points that are ridiculously far away from the LiDAR sensor, the distance magnitude could be 10 to the 30th meters. On the other hand, most of the points in the point cloud data are useless. Points belong to the truck's surface, the data collection device, and points with zero values account for two-thirds of the total points. Therefore, I first filter out the outliers (if a point has no neighbors within a sphere with a radius of 1000 meters, it is an outlier), and remove useless points near the LiDAR. The size of a point cloud shrinks to one-third of its original size after the preprocessing, which significantly reduces the GPU memory usage during training.

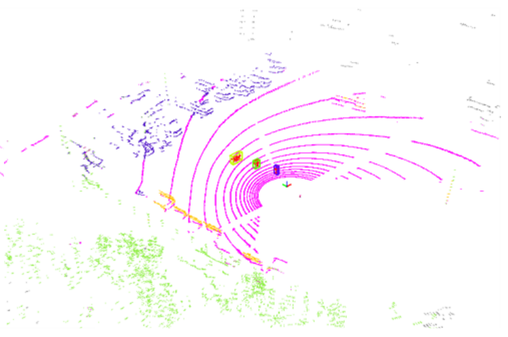
3.4 Evaluation

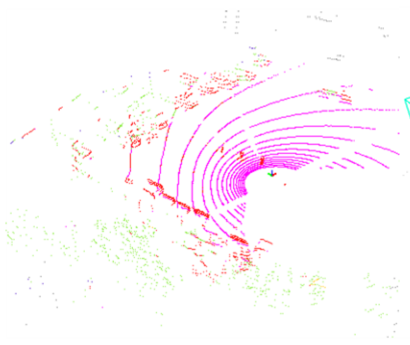
Both qualitative and quantitative evaluations of the models is given in this section.

3.4.1 Qualitative Evalutaion

For qualitative evaluation, the quality of point cloud panoptic segmentation will be assessed by human eye observation. Things from different classes and instances of different objects are clearly separated in a good segmentation result. Figure 3.4 (b) gives a perfect result of 4D-PLS; (c) (d) show two examples that fail to segment things and stuff respectively.


(a) Ground Truth


(b) 4dpls in good quality


(c) Bad semantic segmentation

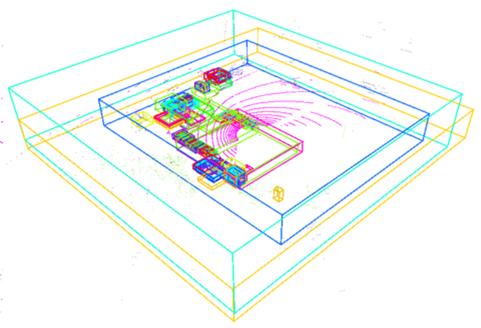

(d) Bad instance segmentation

Figure 3.4: Segmentation results

3.4.2 Quantitative Evaluation

To evaluate the 4D panoptic segmentation task as a whole, LiDAR Segmentation and Tracking Quality (LSTQ) is the main evaluation metric [3], which measures both the segmentation quality and object tracking quality. Meanwhile, I keep the most intuitive metric that are commonly used in classification/segmentation: accuracy.
LSTQ is a point-level metric that evaluates a sequence of point clouds from two sides:

- The classification quality that denotes whether the semantic prediction for each point is correct, measured by classification score S_{cls}.
- The association quality that denotes whether each point from thing classes is assigned to the correct object instance, measured by association score S_{assoc}.

Each point p corresponds to a ground-truth value $p.gt$ (manually annotated label) and a prediction value $p.pr$. Both ground-truth and predicted values

contain two attributes: semantic class $i \in [0, C]$ (0 for points with no labels, C is the number of semantic classes) and instance identification $j \in [0, N]$ (0 for background points, N is the number of object instances), the predictions of which are assessed by S_{cls} and S_{assoc} separately.
Classification Score. Ground-truth set $gt(i)$ and predicted set $pr(i)$ of class i for semantic classification is defined as:

$$\begin{aligned} gt(i) =& \{p | p.gt\_semantic = i\} \\ pr(i) =& \{p | p.pr\_semantic = i\} \end{aligned}$$

The true positive (TP_i), false positive (FP_i), false negative (FN_i) sets of class i is:

$$\begin{aligned} TP_i =& |pr(i) \cap gt(i)| \\ FP_i =& |pr(i) - pr(i) \cap gt(i)| \\ FN_i =& |gt(i) - pr(i) \cap gt(i)| \end{aligned}$$

The classification score S_{cls} is defined in the same way as mean Intersection over Union (IoU) for semantic segmentation.

$$S_{cls} = \frac{1}{C} \sum_{i=1}^{C} IoU(i) = \frac{1}{C} \sum_{i=1}^{C} \frac{|TP_i|}{|TP_i| + |FP_i| + |FN_i|} \tag{3.9}$$

Association Score Ground-truth and predicted sets and TP, FP, FN sets of instance j for point-instance association is defined as:

$$\begin{aligned} gt(j) =& \{p | p.gt\_instance = j, p.gt\_semantic \in [0, C]\} \\ pr(j) =& \{p | p.pr\_instance = j, p.pr\_semantic \in [0, C]\} \\ TPA_j =& |pr(j) \cap gt(j)| \\ FPA_j =& |pr(j) - pr(j) \cap gt(j)| \\ FNA_j =& |gt(j) - pr(j) \cap gt(j)| \end{aligned}$$

The association score S_{assoc} is defined as a weighted sum of IoU:

$$S_{assoc} = \frac{1}{N} \sum_{j=1}^{N} \frac{TPA_j}{|gt(j)|} IoU(j) \tag{3.10}$$

LSTQ is the geometric mean of S_{cls} and S_{assoc}:

$$LSTQ = \sqrt{S_{cls} \times S_{assoc}} \tag{3.11}$$

Accuracy describes the proportion of correctly predicted samples in the total sample.

$$Accuracy = \frac{|TP| + |TN|}{|TP| + |TN| + |FP| + |FN|} \tag{3.12}$$

3.5 Experiments Setup

To investigate the extent to which the performance of the baseline model can be improved by HPO, I compare the searched models with the baseline model. Considering the two-stage network training scheme, I first conduct HPO on the pre-training stage, then try to merge the two stages based on the knowledge got from the first set of experiments (e.g. learning curves) and perform HPO on the whole training process.

The first set of experiments is for the pre-training stage. I select all the aforementioned hyperparameters except the learning rate scheduler and set the max number of training epochs to 50. Each searched hyperparameter configuration is trained from scratch, then evaluated on the validation set based on accuracy and IoU.

The second set of experiments explores the whole search space. I train 200 epochs in total on one stage: the first 40 epochs with only semantic loss to guarantee a good initialization, and another 160 epochs with the full loss. Experiment environment is Ubuntu 18.04, with CUDA 11.3 and Pytorch 1.12 installed on it. All the experiments are done on NVIDIA RTX A4000, 16 gigabytes of memory.

Chapter 4

Result and Analysis

This chapter presents the results of all the experiments I have done. Due to the two-step training scheme taken to train the baseline model, I conducted two sets of experiments: the first set for HPO on the pre-training stage, the second set for the whole stage HPO which is designed based on the knowledge obtained from the first set of experiments' results. Experiments are introduced in order, followed by the analyses of the results.

4.1 Experiments on the Pre-training Stage

In the first set of experiments, I study 4 hyperparameters: Batch size, learning rate, momentum, weight decay, the types and ranges of which are listed in chapter 3. I only use accuracy as the objective function / evaluation metric. Considering that the number of epochs was set to 200 for the baseline pre-training model, I first checked the learning curve of the baseline model. Figure 4.1 tells that the loss is fluctuating in the first 200 epochs, after that the loss almost stop decreasing. I assume that the fluctuation is triggered by the wrong hyperparameter selection. As a result, I infer that it is possible to reach convergence with fewer training epochs if the hyperparameters are set properly.

I first tried 50 training epochs. Figure 4.2 shows the learning curves, which reveals that the result was unexpectedly good. The accuracy after 50 epochs' training is 0.9126, while the accuracy of baseline (200 epochs) is 0.8945. To further investigate if the searched configuration remains high accuracy after 200 epochs training, I retrained the model following the same training setting applied on baseline model, only replace these 4 hyperparameters values

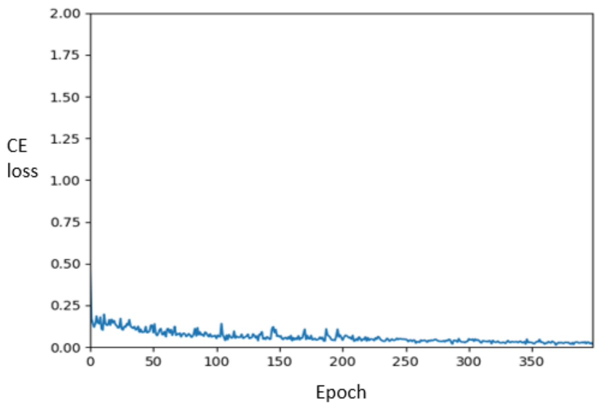


Figure 4.1: Learning curve of baseline model (L_{class})

with the searched ones. Table 4.1 contains the evaluations of both baseline model and searched model is based on accuracy and mean IoU, which is the classification score S_{cls} in LSTQ.

	Checkpoint (epoch)	Accuracy (%)	S_{cls} (%)
Baseline model	50	73.06	52.66
	100	80.18	59.71
	200	89.29	87.61
Searched model	50	91.26	90.89
	100	90.56	90.43
	200	89.87	88.58

Table 4.1: Performance of baseline model and searched model (pre-training stage)

Although the performance of searched model is better on every checkpoint, I noticed that the searched model is degenerating as the number of training epochs increase. The main reason of degeneration lies in the learning rate scheduler. I used the old learning rate scheduler that shrinks the learning rate conservatively, so that the learning rate remained relatively big when the model was converged (after 50 epochs), which leads to a back-off of model performance. This motivates us to study learning rate scheduler, especially when the training epochs go up.

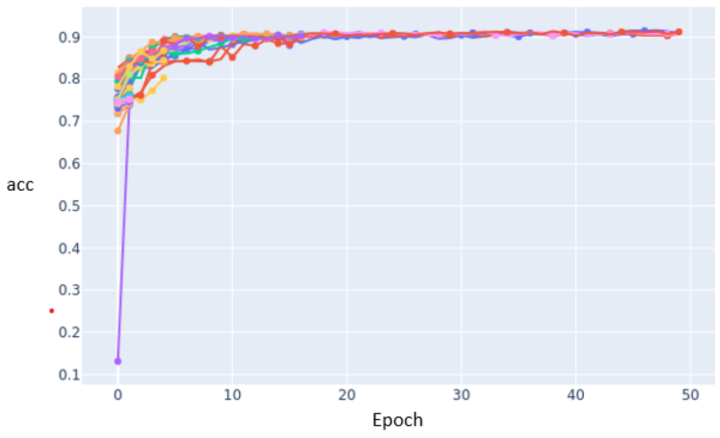


Figure 4.2: Learning curves of trials (accuracy)

Figure 4.2 shows that the model converges after 20 epochs with proper hyperparameter setting.

4.2 Experiment results

In the first 40 epochs, I set the model in pre-training mode where only CrossEntropyLoss applies, after that I trained the model with full loss. The model is trained for 200 epochs in total, and the learning curves is shown in Figure 4.3.

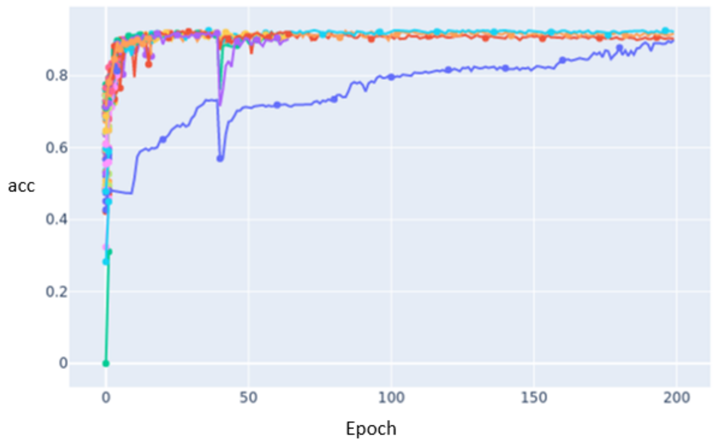


Figure 4.3: Learning curves of trials (accuracy). The gaps in epoch 40 denote a change of loss function during model training

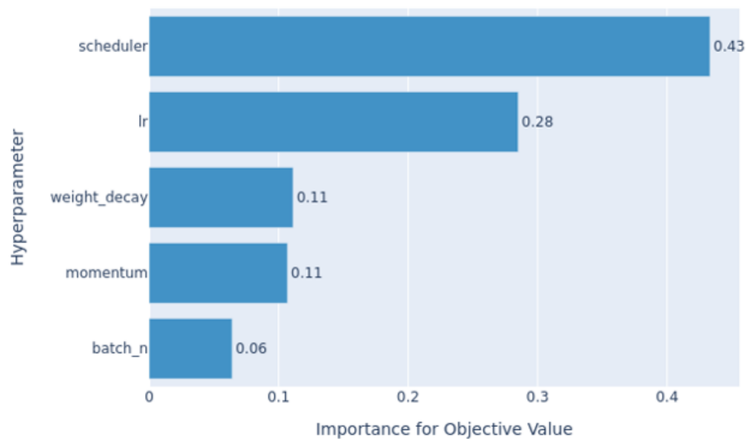


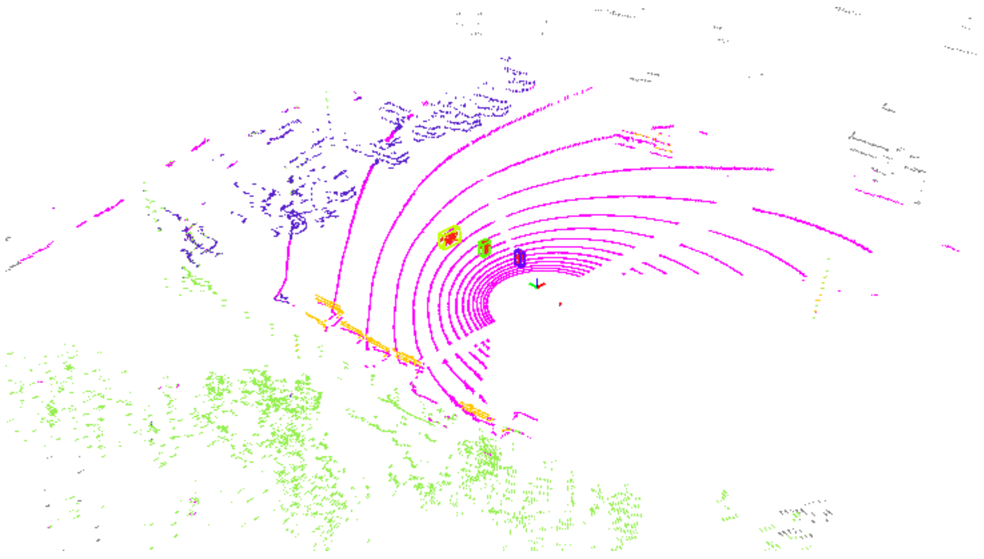
Figure 4.4: Hyperparameter Importance

Figure 4.4 reveals the importance between hyperparameters, which convinces me the learning rate is the most important hyperparameter to be tuned.
I compared the searched model with the baseline model, the results are in Table 4.2. According to the classification score S_{cls} and association score S_{assoc}, the searched model outperforms the baseline model in object association, while slightly worse in semantic segmentation. Follow the LSTQ metric, the searched model overall is better than the baseline model. Note: baseline model is trained for 1000 epochs in total, yet the searched model is only trained for 200 epochs.

	Accuracy (%)	S_{assoc} (%)	S_{cls} (%)	LSTQ (%)
Baseline model	91.02	95.20	92.76	93.99
Searched model	91.06	97.93	91.60	94.71

Table 4.2: Comparison between the baseline model and the searched model

The visualization of the results are shown in Figure 4.5. There is no any obvious difference bewteen two figures.


(a) Baseline model

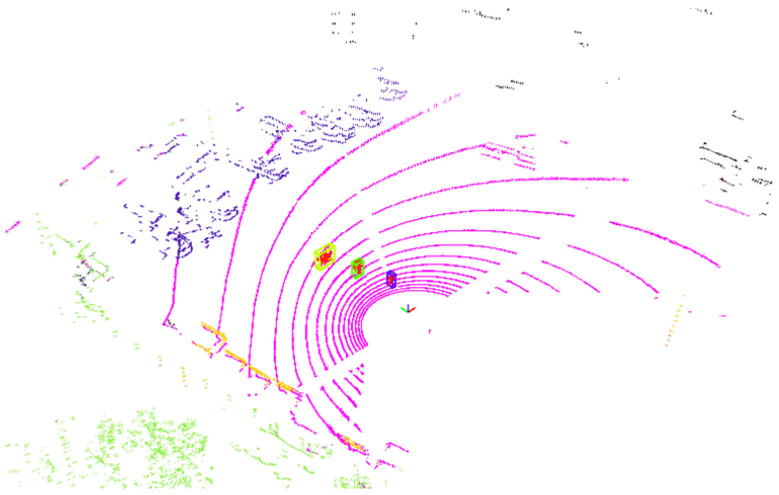

(b) Searched model

Figure 4.5: Visulizations of the results from baseline model and searched model. The only difference lies on the top left of the figure, where the first trunk on the right is predicted as grass (green) by the searched model

Chapter 5

Conclusions and Future work

In this chapter, I summarize the work I have done and draw conclusions related to the research questions. Besides, this chapter gives limitations of the work and what could be done in the future.

5.1 Conclusions

This work focused on the task of 4D-PLS and gave a solution that integrates AutoML techniques to maximize the performance of the 4D-PLS model. In fact, the model with searched hyperparameters outperformed the baseline model and was obtained in fewer training epochs. I helped our team in Volvo prepare the point cloud datasets, then built a 4D-PLS model and found the best hyperparameter settings to optimize the model.

5.2 Limitations

From the perspective of 4D-PLS, the limitations of our model are threefold. First, the model has the potential to identify instances of more objects from different classes, if given enough labeled data; Second, the application scenarios of our model were only stationary scenes, and can be extended to dynamic scenes; Third, the inference speed is slow. In terms of hyperparameter optimization, the limitations are mainly in the long runtime of the optimization process.

5.3 Future work

For the limitations in the 4D-PLS model, there is still a long way to go before the model is commercialized. To expand the recognition ability of the model, a huge amount of annotated point cloud data is necessary for supervised learning methods. One promising direction is annotating algorithms. Imagine that all the tedious data labeling work is done by a smart annotation tool in the future. To run the model in real-time, on the one hand, I need to synchronize with other engineers during the model deployment to cut off unnecessary operations (e.g. data format conversion); On the other hand, replace the encoder network or the logic of the decoder inference, to reduce parameters and computations.
For the limitations of HPO, a future research effort is needed to address the long runtime of searching processes. One possible solution is applying some advanced multi-fidelity methods to save time [65]. In addition, more work should be done to build an AutoML pipeline.

References

[1] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, “Panoptic segmentation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9404–9413. [Pages 1 and 9.]

[2] A. Nguyen and B. Le, “3D point cloud segmentation: A survey,” in *2013 6th IEEE conference on robotics, automation and mechatronics (RAM)*. IEEE, 2013, pp. 225–230. [Pages 1 and 8.]

[3] M. Aygun, A. Osep, M. Weber, M. Maximov, C. Stachniss, J. Behley, and L. Leal-Taixé, “4D panoptic lidar segmentation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 5527–5537. [Pages 1, 4, 9, 16, 21, and 31.]

[4] M. Feurer and F. Hutter, “Hyperparameter optimization,” in *Automated machine learning*. Springer, Cham, 2019, pp. 3–33. [Page 1.]

[5] M. Frank, R. Ruvald, C. Johansson, T. Larsson, and A. Larsson, “Towards autonomous construction equipment: supporting on-site collaboration between automatons and humans,” *International Journal of Product Development*, vol. 23, no. 4, pp. 292–308, 2019. [Page 2.]

[6] H. Tang, Z. Liu, S. Zhao, Y. Lin, J. Lin, H. Wang, and S. Han, “Searching efficient 3D architectures with sparse point-voxel convolution,” in *European conference on computer vision*. Springer, 2020, pp. 685–702. [Pages 2 and 15.]

[7] D. Watzenig and M. Horn, *Automated driving: safer and more efficient future driving*. Springer, 2016. [Page 4.]

[8] M. Ryan, “The future of transportation: ethical, legal, social and economic impacts of self-driving vehicles in the year 2025,” *Science and engineering ethics*, vol. 26, no. 3, pp. 1185–1208, 2020. [Page 4.]

[9] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778. [Pages 4, 10, and 22.]

[10] R. Girshick, “Fast R-CNN,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448. [Pages 4 and 10.]

[11] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969. [Pages 4 and 15.]

[12] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3D classification and segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660. [Pages 7, 10, and 14.]

[13] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, “A comparison and evaluation of multi-view stereo reconstruction algorithms,” in *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR’06)*, vol. 1. IEEE, 2006, pp. 519–528. [Page 7.]

[14] Y. Xie, J. Tian, and X. X. Zhu, “Linking points with labels in 3D: A review of point cloud semantic segmentation,” *IEEE Geoscience and Remote Sensing Magazine*, vol. 8, no. 4, pp. 38–59, 2020. doi: 10.1109/MGRS.2019.2937630 [Pages 8 and 10.]

[15] J. Zhang, X. Lin, and X. Ning, “Svm-based classification of segmented airborne lidar point clouds in urban areas,” *Remote Sensing*, vol. 5, no. 8, pp. 3749–3775, 2013. [Page 8.]

[16] J.-F. Lalonde, R. Unnikrishnan, N. Vandapel, and M. Hebert, “Scale selection for classification of point-sampled 3D surfaces,” in *Fifth International Conference on 3-D Digital Imaging and Modeling (3DIM’05)*. IEEE, 2005, pp. 285–292. [Page 8.]

[17] N. Chehata, L. Guo, and C. Mallet, “Airborne lidar feature selection for urban classification using random forests,” in *Laserscanning*, 2009. [Page 8.]

[18] J. Niemeyer, F. Rottensteiner, and U. Soergel, “Conditional random fields for lidar point cloud classification in complex urban areas,” *ISPRS Ann.*

Photogramm. Remote Sens. Spat. Inf. Sci, vol. 1, no. 3, pp. 263–268, 2012. [Page 8.]

[19] L.-C. Chen, A. Hermans, G. Papandreou, F. Schroff, P. Wang, and H. Adam, "Masklab: Instance segmentation by refining object detection with semantic and direction features," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4013–4022. [Page 9.]

[20] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org. [Page 10.]

[21] 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. [Page 10.]

[22] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017. [Pages 10 and 14.]

[23] 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. [Pages 10 and 14.]

[24] H. Zhou, X. Zhu, X. Song, Y. Ma, Z. Wang, H. Li, and D. Lin, "Cylinder3d: An effective 3D framework for driving-scene lidar semantic segmentation," *arXiv preprint arXiv:2008.01550*, 2020. [Page 10.]

[25] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "Rangenet++: Fast and accurate lidar semantic segmentation," in *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2019, pp. 4213–4220. [Pages 11 and 15.]

[26] L. T. Triess, D. Peter, C. B. Rist, and J. M. Zöllner, "Scan-based semantic segmentation of lidar point clouds: An experimental study," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1116–1121. [Pages 11 and 15.]

[27] Y. Zhang, Z. Zhou, P. David, X. Yue, Z. Xi, B. Gong, and H. Foroosh, "Polarnet: An improved grid representation for online lidar point clouds

semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9601–9610. [Pages 11 and 15.]

[28] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *Acm Transactions On Graphics (tog)*, vol. 38, no. 5, pp. 1–12, 2019. [Page 11.]

[29] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, and T.-K. Kim, "Multiple object tracking: A literature review," *Artificial Intelligence*, vol. 293, p. 103448, 2021. [Page 11.]

[30] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: the clear mot metrics," *EURASIP Journal on Image and Video Processing*, vol. 2008, pp. 1–10, 2008. [Page 11.]

[31] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019. [Page 12.]

[32] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, 2020. [Page 12.]

[33] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 9, pp. 5149–5169, 2021. [Page 13.]

[34] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 1997–2017, 2019. [Page 13.]

[35] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 764–773. [Page 14.]

[36] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "SpiderCNN: Deep learning on point sets with parameterized convolutional filters," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 87–102. [Page 14.]

[37] M. Atzmon, H. Maron, and Y. Lipman, "Point convolutional neural networks by extension operators," *arXiv preprint arXiv:1803.10091*, 2018. [Page 14.]

[38] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "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. 11 108–11 117. [Page 14.]

[39] 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. [Page 14.]

[40] 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. [Page 14.]

[41] C. Choy, J. Gwak, and S. Savarese, "4D spatio-temporal convnets: Minkowski convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3075–3084. [Pages 14 and 15.]

[42] B. Graham, M. Engelcke, and L. Van Der Maaten, "3d semantic segmentation with submanifold sparse convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9224–9232. [Page 14.]

[43] X. Zhu, H. Zhou, T. Wang, F. Hong, Y. Ma, W. Li, H. Li, and D. Lin, "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. [Page 14.]

[44] B. Wu, A. Wan, X. Yue, and K. Keutzer, "Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3D lidar point cloud," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1887–1893. [Page 15.]

[45] E. E. Aksoy, S. Baci, and S. Cavdar, "Salsanet: Fast road and vehicle segmentation in lidar point clouds for autonomous driving," in *2020 IEEE intelligent vehicles symposium (IV)*. IEEE, 2020, pp. 926–932. [Page 15.]

[46] T. Cortinhal, G. Tzelepis, and E. Erdal Aksoy, “Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds,” in *International Symposium on Visual Computing*. Springer, 2020, pp. 207–222. [Page 15.]

[47] Z. Liu, H. Tang, Y. Lin, and S. Han, “Point-voxel CNN for efficient 3D deep learning,” *Advances in Neural Information Processing Systems*, vol. 32, 2019. [Page 15.]

[48] F. Zhang, J. Fang, B. Wah, and P. Torr, “Deep fusionnet for point cloud semantic segmentation,” in *European Conference on Computer Vision*. Springer, 2020, pp. 644–663. [Page 15.]

[49] J. Xu, R. Zhang, J. Dou, Y. Zhu, J. Sun, and S. Pu, “Rpvnet: A deep and efficient range-point-voxel fusion network for lidar point cloud segmentation,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 16 024–16 033. [Page 15.]

[50] Z. Cai and N. Vasconcelos, “Cascade R-CNN: high quality object detection and instance segmentation,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 5, pp. 1483–1498, 2019. [Page 15.]

[51] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang *et al.*, “Hybrid task cascade for instance segmentation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4974–4983. [Page 15.]

[52] B. De Brabandere, D. Neven, and L. Van Gool, “Semantic instance segmentation with a discriminative loss function,” *arXiv preprint arXiv:1708.02551*, 2017. [Page 15.]

[53] A. Newell, Z. Huang, and J. Deng, “Associative embedding: End-to-end learning for joint detection and grouping,” *Advances in neural information processing systems*, vol. 30, 2017. [Page 15.]

[54] P. Voigtlaender, M. Krause, A. Osep, J. Luiten, B. B. G. Sekar, A. Geiger, and B. Leibe, “Mots: Multi-object tracking and segmentation,” in *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 2019, pp. 7942–7951. [Page 16.]

[55] A. Athar, S. Mahadevan, A. Osep, L. Leal-Taixé, and B. Leibe, "Stem-seg: Spatio-temporal embeddings for instance segmentation in videos," in *European Conference on Computer Vision*. Springer, 2020, pp. 158–177. [Pages 16 and 24.]

[56] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6411–6420. [Pages 16 and 21.]

[57] R. Marcuzzi, L. Nunes, L. Wiesmann, I. Vizzo, J. Behley, and C. Stachniss, "Contrastive instance association for 4D panoptic segmentation using sequences of 3D lidar scans," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1550–1557, 2022. [Page 16.]

[58] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *Journal of machine learning research*, vol. 13, no. 2, 2012. [Page 17.]

[59] B. Bischl, M. Binder, M. Lang, T. Pielok, J. Richter, S. Coors, J. Thomas, T. Ullmann, M. Becker, A.-L. Boulesteix *et al.*, "Hyperparameter optimization: Foundations, algorithms, best practices and open challenges," *arXiv preprint arXiv:2107.05847*, 2021. [Page 17.]

[60] E. Hazan, A. Klivans, and Y. Yuan, "Hyperparameter optimization: A spectral approach," *arXiv preprint arXiv:1706.00764*, 2017. [Page 17.]

[61] M. Seeger, "Gaussian processes for machine learning," *International journal of neural systems*, vol. 14, no. 02, pp. 69–106, 2004. [Page 18.]

[62] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *International conference on learning and intelligent optimization*. Springer, 2011, pp. 507–523. [Pages 18 and 26.]

[63] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in neural information processing systems*, vol. 24, 2011. [Pages 18 and 27.]

[64] K. Jamieson and A. Talwalkar, "Non-stochastic best arm identification and hyperparameter optimization," in *Artificial intelligence and statistics*. PMLR, 2016, pp. 240–248. [Page 18.]

[65] S. Falkner, A. Klein, and F. Hutter, “Bohb: Robust and efficient hyperparameter optimization at scale,” in *International Conference on Machine Learning*. PMLR, 2018, pp. 1437–1446. [Pages 18 and 42.]

[66] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, “Hyperband: A novel bandit-based approach to hyperparameter optimization,” *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6765–6816, 2017. [Page 18.]

[67] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A next-generation hyperparameter optimization framework,” in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631. [Page 18.]

[68] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, “Tune: A research platform for distributed model selection and training,” *arXiv preprint arXiv:1807.05118*, 2018. [Page 18.]

[69] L. Zimmer, M. Lindauer, and F. Hutter, “Auto-pytorch: Multi-fidelity metalearning for efficient and robust autodl,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 9, pp. 3079–3090, 2021. [Page 19.]

[70] M. Lindauer, K. Eggensperger, M. Feurer, A. Biedenkapp, D. Deng, C. Benjamins, T. Ruhkopf, R. Sass, and F. Hutter, “Smac3: A versatile bayesian optimization package for hyperparameter optimization.” *J. Mach. Learn. Res.*, vol. 23, pp. 54–1, 2022. [Page 19.]

[71] D. Neven, B. D. Brabandere, M. Proesmans, and L. V. Gool, “Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8837–8845. [Page 24.]

[72] B. Wang, V. Wu, B. Wu, and K. Keutzer, “Latte: accelerating lidar point cloud annotation via sensor fusion, one-click annotation, and tracking,” in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 265–272. [Page 29.]

[73] C. Sager, P. Zschech, and N. Kühl, “labelcloud: A lightweight domain-independent labeling tool for 3D object detection in point clouds,” *arXiv preprint arXiv:2103.04970*, 2021. [Page 29.]

[74] E. Li, S. Wang, C. Li, D. Li, X. Wu, and Q. Hao, "Sustech points: A portable 3D point cloud interactive annotation platform system," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1108–1115. [Page 29.]

[75] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "Semantickitti: A dataset for semantic scene understanding of lidar sequences," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9297–9307. [Page 29.]

Appendix A

Documentation

A.1 Software Installation

I use docker to create a separated environment for my experiments, more details are written in How I set the docker environment. However, conda environment is recommended in our case. The main reason of choosing docker is to avoid CUDA version conflict and make sure the security of the host system.

A.2 Data Annotation

Raw point cloud data should be transformed into ’bin’ format, the same format used in Semantic-Kitti. For convenience, we use Kitti Annotation Tool. The file that contains poses of LiDAR scans, i.e. ’poses.txt’, is missing, since we do not have any poses information. In our case, the LiDAR is mounted on a stationary truck, so the pose of one scan is set to an identity matrix. The poses can be obtained by IMU or poses estimation methods. Note that the poses file is only used for the Kitti annotation tool. As for the calibration file, which is useful only if the images from cameras are available, can be set as identity matrix. Recording time of every scan is written in ’times.txt’, which is the name of raw point cloud data.

A.3 Network Fine-tuning

Regarding of the backbone network training, please see How to train the network. In this section, I will provide some ideas about how to fine-tune

the network. To find the best hyperparameter configuration, one way is using all the training datasets and performing HPO for months. I follow another way that saves me a lot of time by using multi-fidelity tricks. Practically, I conduct HPO on the subset of datasets and train the network within fewer iterations. In this case, the searched hyperparameters maybe the best for the limited iterations, but not necessarily the best for the full training process. To improve the network performance, I run another several hundred ieration with finetuned hyperparameters after HPO. For example, I evaluate the performance at each check point and adjust learning rate according to the performance metrics.
For future work, I suggest to conduct HPO on the full datasets, including the augmented ones, then train the network up to 500 epochs given enough time.

TRITA-EECS-EX- 2022:00