

Norwegian University of Life Sciences  
Faculty of Science and Technology

Philosophiae Doctor (PhD)  
Thesis 2020:25

# Information Extraction from Large Point Cloud Data: A Deep Learning Approach

Uthenting av informasjon fra store  
punktskydatasett: En tilnærming basert  
på dyp læring

Hasan Asy'ari Arief



# Information Extraction from Large Point Cloud Data: A Deep Learning Approach

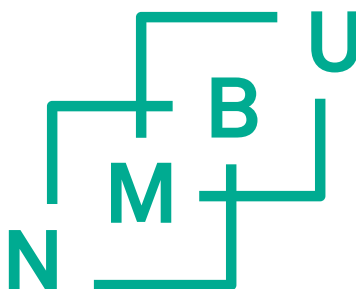
Uthenting av informasjon fra store punktskydatasett:  
En tilnærming basert på dyp læring

Philosophiae Doctor (PhD) Thesis

Hasan Asy'ari Arief

Norwegian University of Life Sciences  
Faculty of Science and Technology

Ås (2020)



## Supervision Team

### *Main supervisor*

Associate Professor Håvard Tveite  
Faculty of Science and Technology (RealTek),  
Norwegian University of Life Sciences (NMBU),  
Ås, Norway

### *Co-supervisor*

Associate Professor Ulf Geir Indahl  
Faculty of Science and Technology (RealTek),  
Norwegian University of Life Sciences (NMBU),  
Ås, Norway

### *Co-supervisor*

Head of Research Geir-Harald Strand<sup>1</sup>  
Division of Survey and Statistics,  
Norwegian Institute of Bioeconomy Research (NIBIO),  
Ås, Norway

## Evaluation Committee

### *First opponent*

Research Scientist, Dr. Johannes Kvam  
Smart Sensor Systems,  
SINTEF Digital,  
Oslo, Norway

### *Second opponent*

Professor Terje Midtbø  
Faculty of Engineering,  
Norwegian University of Science and Technology (NTNU),  
Trondheim, Norway

### *Committee coordinator*

Associate Professor Kristian Hovde Liland  
Faculty of Science and Technology (RealTek),  
Norwegian University of Life Sciences (NMBU),  
Ås, Norway

---

<sup>1</sup> Adjunct Professor at the Faculty of Science and Technology (RealTek), Norwegian University of Life Sciences (NMBU), Ås, Norway



---

# Summary

---

Recent advances in Light Detection and Ranging (LiDAR) sensors have led to an increasing amount of large scale point cloud data collections. The LiDAR sensors can capture the fine spatial details of a remote environment in a full three-dimensional perspective, thus providing huge potentials for better machine understanding of a 3D scene.

This thesis explores these potentials by providing robust and effective ways to extract information from large scale point cloud data. The study focuses on the utilization of deep learning techniques for the 3D scene understanding tasks, i.e semantic segmentation and object detection. It should be noted that the deep learning techniques were chosen mainly because the techniques simplify the generation of representative and robust features taking into account the spatial autocorrelation of input data, while often resulting in the highest prediction accuracies.

As the backbone of this thesis, the deep learning approach has shown remarkable progress in generating the highest classification accuracy for several benchmark datasets, including our in-house dataset. Our contributions to improve the quality of point cloud annotation is closely related to the improvement of the deep learning models, i.e improving the deep learning preprocessing step by using a better density sampling approach, restructuring the deep learning modules by developing our Stochastic Atrous Network (SA-NET) architecture, and refining the post-processing step of deep learning prediction by invoking spatial and spectral similarities of point cloud data, using our Atrous  $\mathcal{X}$  Conditional Random Field (A-XCRF) algorithm.

The present PhD-work started by addressing some challenging problems regarding the modelling of the 3D point cloud data, and it was completed by providing a deliverable prototype capable of generating fast and accurate point cloud annotation labels. During the research process, we have managed to develop a better solution for extracting information in the form of semantic labelling from 2D projected point cloud data. We also developed a post-processing module refining point-level classifications *directly* generated from raw point cloud data.

Finally, we developed an open-source and robust semi-automatic point cloud annotation tool, called Smart Annotation and Evaluation (SAnE). The SAnE

---

speeds up the point cloud annotation process while also offering significantly better annotation accuracy than the baseline annotation approaches.

**Keywords:** point cloud annotation, deep learning, semantic mapping, 3D object detection, land cover segmentation, autonomous vehicle application.

---

# Sammendrag

---

Utviklingen innen Light Detection and Ranging (LiDAR) sensorer har de siste årene ført til en økende innsamling av data i form av storskala punktskyer. Med LiDAR-sensorene kan man få høyoppløselige beskrivelser av objekter og miljøer i 3D. De utgjør dermed et enormt potensiale for bedre maskin forståelse av et 3D-bilde.

Denne avhandlingen utforsker dette potensialet gjennom å utvikle robuste og effektive metoder for å hente ut informasjon fra slike punktskyer. Hovedvekten er lagt på dype kunstige nevralt nettverk for tolking av 3D-bilder. Dette omfatter blant annet semantisk segmentering og objekt-deteksjon. Denne typen dyp læring er en sentral metode innenfor maskinlæring. Teknikken ble hovedsakelig valgt fordi den forenkler etableringen av representative og robuste beregningsfunksjoner samtidig som det er mulig å ta hensyn til romlig autokorrelasjon i bildene som analyseres. Metodene viser seg også ofte å gi den høyeste prediksjonsnøyaktigheten.

Metodene innenfor dyp læring som utgjør kjernen i denne oppgaven, har i gjentatte sammenlignende tester gitt svært gode resultater i form av den høyeste klassifiseringsnøyaktigheten for flere referansedatasett, inkludert vårt interne datasett. Våre bidrag til å forbedre kvaliteten i tolkingen av punktskyer er nært knyttet til forbedringen av modellene for dyp læring. Det innebærer for det første en videreutvikling av dyp læring metoder for preprosessering av data gjennom sampling med bedre tetthet. Videre har vi bidratt til restrukturering av modulene for dyp læring ved å utvikle vår SA-NET-arkitektur. For det tredje har vi forbedret etterbehandlingstrinnet i dyp læringsprediksjon ved å ta hensyn til romlige og spektrale likhetstrekk innenfor punktskyen ved å bruke vår A-XCRF-algoritme.

Dette doktorgradsarbeidet startet med å ta fatt i kjente utfordringer innen modellering av punktskyer som avbilder fenomener i 3D. Gjennom arbeidet er nye utfordringer identifisert og det er etablert en prototype for rask og nøyaktig klassifisering av elementer i punktskyer. I løpet av forskningsprosessen har vi klart å utvikle en bedre løsning for å trekke ut informasjon i form av semantisk merking fra 2D projiserte punktskydata.

Vi har også utviklet en etterbehandlingsmodul som forbedrer klassifisering av elementer direkte fra punktskyer. Til slutt utviklet vi (i form av åpen kildekode) en robust, halvautomatisk verktøy for annotering av punktskyer, kalt

---

SAnE. Med dette verktøyet kan annotasjon av punktskyer gjøres mer effektivt, samtidig som det gis betydelig bedre kommentarnøyaktighet enn ved manuelle tilnæringsmetoder.

**Nøkkelord:** punktsky, dyp læring, semantisk kartlegging, 3D-objekt-deteksjon, arealdekke-segmentering, autonome kjøretøyprogram.

---

# Ringkasan

---

Berbagai terobosan terbaru dalam teknologi Light Detection and Ranging (LiDAR) menyebabkan penggunaan dan pengumpulan data berbasis *point cloud* dalam skala besar meningkat. Sensor berbasis LiDAR dapat memberikan tampilan tiga dimensi dari sebuah objek secara utuh, sehingga berpotensi untuk meningkatkan kualitas sistem pengolahan citra secara otomatis dalam lingkungan tiga dimensi.

Disertasi ini mengeksplorasi potensi-potensi di atas dengan menyediakan berbagai terobosan yang aplikatif dan efektif dalam proses ekstraksi informasi dari data berbasis *point cloud*. Penelitian ini berfokus pada implementasi teknologi *deep learning* dalam bidang pengolahan citra digital untuk lingkungan tiga dimensi, seperti segmentasi semantik and pendeteksian objek. Pendekatan semacam ini dipilih karena teknologi *deep learning* memudahkan dan menyederhanakan penyaringan dan pemilihan fitur-fitur terbaik dari sebuah data masukan dengan memperhatikan korelasi dan kedekatan spasial dari data-data tersebut, sehingga seringkali menghasilkan prediksi dengan tingkat akurasi terbaik.

Sebagai bagian utama dari penelitian ini, teknologi *deep learning* telah memperlihatkan berbagai capaian yang signifikan dengan memberikan prediksi dengan tingkat akurasi tertinggi dari berbagai percobaan yang dilakukan, termasuk percobaan menggunakan data-data patokan yang ada. Kontribusi-kontribusi dari disertasi ini dalam rangka meningkatkan kualitas proses penyediaan anotasi terhadap data berbasis *point cloud* sangat terkait dengan terobosan yang diberikan terhadap pengembangan model *deep learning*, antara lain: (1) perbaikan terhadap tahapan pemrosesan data sebelum dimasukkan dalam proses pembelajaran mesin menggunakan pendekatan *density-sampling* dan pemahaman data masukan, (2) penyusunan ulang bagian-bagian dari arsitektur *deep learning* untuk menghasilkan arsitektur terbaik berdasarkan kondisi data yang dimodelkan (arsitektur yang ditawarkan diberi nama SA-NET) dan (3) perbaikan hasil prediksi dengan menghaluskan tingkat kekasaran hasil prediksi dengan menekankan pentingnya kedekatan spasial dan kesamaan spektral dari data berbasis *point cloud* (teknik ini diberi nama A-XCRF).

Dalam kerangka kerja (dan penelitian) yang utuh, penelitian doktoral ini dimulai dengan menjawab berbagai tantangan yang ada dalam memodelkan data tiga dimensi berbasis *point cloud*, kemudian diakhiri dengan menawarkan produk berbasis *software* kode terbuka yang mampu menghasilkan anotasi

---

data berbasis *point cloud* secara cepat dan akurat. Pertama-tama, sebuah proses penelitian dilakukan yang kemudian menghasilkan solusi terbaik dalam proses ekstraksi informasi dibidang segmentasi semantik dari data berbasis *point cloud*. Sebagai catatan, data berbasis *point cloud* yang digunakan diproyeksikan terlebih dahulu dalam bidang dua dimensi. Pada tahapan berikutnya, sebuah algorithm penghalusan hasil prediksi diusulkan untuk memperbaiki hasil prediksi segmentasi semantik yang dihasilkan langsung dari data input yang berasal dari data mentah berbasis *point cloud*. Pada tahap akhir, sebuah software berbasis kode terbuka ditawarkan. Software ini berfungsi untuk menganotasi data berbasis *point cloud* secara cepat dengan akurasi yang jauh lebih tinggi dibanding dengan proses anotasi secara manual.

**Kata kunci:** *point cloud*, *deep learning*, pendeteksian objek, segmentasi semantik, aplikasi mobil otomatis.

---

# Acknowledgements

---

This work was carried out at the Faculty of Science and Technology, Norwegian University of Life Sciences. I would like to thank the university for its generous financial support.

I would like to thank my main supervisor Dr. Håvard Tveite, who gave me a chance to become a better researcher. Through all the years of my Ph.D. career, he has guided and helped me explore the world of research passionately with patience and collaborative work while teaching me the meaning of perseverance and hard work in the process. Thank you for all your help, conversations, and advises.

I am obliged to my co-supervisors Dr. Ulf Geir Indahl and Prof. Geir-Harald Strand, who always have insightful ideas, giving me many suggestions and supports along the way of my study at the university. With his deep knowledge on the field, Dr. Ulf has always been a good sparring partner, correcting my mistakes and sharpening my understanding of the fundamental domain knowledge in the research area. With his wisdom and experience, Prof. Geir-Harald taught me more deeply about critical thinking while (always) bringing fresh perspectives on the discussions, helping shape my research outcome for the better.

I also would like to thank Dr. Ding Zhao from Carnegie Mellon University for giving me the opportunity to join the Safe AI lab for six months. Also, many thanks to all the Safe AI lab members, especially to my brother Ph.D. fellow Mansur Maturidi Arief, who made my stay (in the US) fun and productive. I wish you all the success in the world!

Tremendous gratitude goes to my former office-mates Dr. Ivar Oveland and Dr. Martina Idžanović. Guys, thank you for the friendship and I wish you all success in your careers and beyond! Also, for my (current) office-mate Ph.D. Fellow Brian Bramanto, enjoy Norway and good luck with your future!

I also would like to thank the members of the evaluation committee: Dr. Johannes Kvam, Prof. Terje Midtbø, and Dr. Kristian Hovde Liland.

A special thanks to all the members of UMINor for always welcoming me and my family warmly, making it easier for us to settle in this northern part of the world.

---

Finally, I would like to thank all my family and friends in Indonesia, and express my deepest gratitude to my wife Nurul Isma, who has been there with me from the start, through thick and thin, with support, love, and enjoyment. Last but not least, to the most important people in my life: my mother Nuraini and my father Arief Halim, who have been supporting me my whole life, I am forever in debt to you both. I would not be the person I am today without your supports. This is for both of you!

*Hasan Asy'ari Arief*

*January 2020*



---

# Contents

---

<b>Contents</b>	<b>xi</b>
<b>List of Figures</b>	<b>xiii</b>
<b>List of Tables</b>	<b>xv</b>
<b>1 Introductions</b>	<b>1</b>
1.1 Overview . . . . .	1
1.2 Research Questions and Objectives . . . . .	2
1.3 Contributions and Thesis Outline . . . . .	3
1.4 Publications . . . . .	5
<b>2 Background</b>	<b>7</b>
2.1 Datasets . . . . .	7
2.2 3D Scene Understanding . . . . .	11
2.2.1 Semantic Segmentation . . . . .	11
2.2.2 Object Detection . . . . .	12
2.3 Deep Learning for Point Cloud Data . . . . .	13
2.3.1 2D Projection . . . . .	13
2.3.2 3D Point Cloud Representation . . . . .	16
2.3.3 Semi-automatic Annotation . . . . .	17
2.4 Evaluation Metrics . . . . .	18
<b>3 Publications</b>	<b>21</b>
3.1 Semantic Mapping for 2D Projected Point Cloud . . . . .	22
3.2 Pointwise Segmentation for 3D Point Cloud Representations . . . . .	25
3.3 Deep Learning for Point Cloud Annotations . . . . .	28
3.4 Semi-automatic Point Cloud Annotation Tools . . . . .	31
<b>4 Conclusions</b>	<b>33</b>
4.1 Technical Evolution of The Thesis . . . . .	33
4.2 Limitations of our approach . . . . .	35
4.3 Future Directions . . . . .	36
4.4 Outlook . . . . .	37
<b>Bibliography</b>	<b>39</b>
	xi

## Contents

---

Appendices	47
Paper A: Land Cover Segmentation of Airborne LiDAR Data Using Stochastic Atrous Network	49
Paper B: Addressing Overfitting on Point Cloud Classification using Atrous XCRF	73
Paper C: Density-Adaptive Sampling for Heterogeneous Point Cloud Object Segmentation in Autonomous Vehicle Ap- plications	93
Paper D: SAnE: Smart Annotation and Evaluation Tools for Point Cloud Data	103

---

## List of Figures

---

2.1	Map of Norway (1:15M, UTM Zone 33N) showing the location of the Follo area. . . . .	8
2.2	Example from the Follo point cloud - forested area. . . . .	9
2.3	Map of Norway (1:15M, UTM Zone 33N) showing the location of Bergen. . . . .	10
2.4	Point-level classification from the Bergen dataset. . . . .	11
2.5	3D object detection based on PointRCNN (Shi et al., 2019). . . . .	12
2.6	Convolution operation using 2D kernel. . . . .	14
2.7	The max-pooling operation using a 2D kernel. . . . .	15
2.8	Rectifier unit (blue line). Image source: Dan Stowell (distributed under a CC0 license). . . . .	15
3.1	The early fusion SA-NET architecture. . . . .	22
3.2	The 3 by 3 atrous kernel with different number of holes, defined using the value of <i>rate</i> . . . . .	22
3.3	Residual connection using bottleneck building block. . . . .	23
3.4	Point cloud feature learning with MLP. . . . .	25
3.5	On-the-fly preprocessing on x-Conv algorithm: (a) point cloud input data with x,y,z dimension, (b) each point gather neighboring points and normalized their value, (c) final input data used for MLPs learning process. . . . .	25
3.6	Full pipeline of A-XCRF technique using the PointCNN as the main deep learning architecture. . . . .	26
3.7	The nature of point cloud data from two different domains: (a) driving scene point cloud from Velodyne-type LiDAR (b) landscape map point cloud from airborne LiDAR along with point density distributions (c-d). . . . .	28
3.8	The density-adaptive sampling pipeline for semantic segmentation for heterogeneous density point cloud. . . . .	29
3.9	Point cloud visualization of the KITTI dataset with (a) Prediction results, and (b) Ground truth label with missing object bounding box. . . . .	30
3.10	The interface of SAnE, a semi-automatic annotation tool based on one-click annotation scheme empowered with denoising point-wise segmentation approach and robust guided-tracking algorithm. . . . .	31

List of Figures

---

3.11 The results of denoising algorithm (a) before and (b) after the implementation. The algorithm enables the use of one-click annotation techniques for annotation tool. . . . . 32

---

## List of Tables

---

2.1	Class distribution of the Vaihingen 3D semantic labeling dataset. . . . .	9
3.1	The test result for 2D semantic segmentation. CRF: conditional random field; MIoU: mean intersection-over-union; MPA: mean pixel accuracies; PA: pixel accuracy. . . . .	24
3.2	A quantitative comparison between A-XCRF and other methods on the Vaihingen dataset, namely (a) ISS_7 (Ramiya et al., 2016), (b) UM (Horvat et al., 2016), (c) HM_1 (Steinsiek et al., 2017), (d) LUH (Niemeyer et al., 2016), (e) RIT_1 (Yousefflussen et al., 2017), (f) WhuY4 (Yang et al., 2018), (g) PointCNN (Li et al., 2018), and (h) A-XCRF (Arief et al., 2019b). All cells except the last two rows show the per-class F1 score. . . . .	26
3.3	The performance of each of the sampling scenario. VB: Voxel based sampling; GBU: Grid based uniform sampling; GBR: Grid based random sampling. . . . .	29
3.4	Bounding box accuracies for objects in front of the ego vehicle and objects in the whole area of the point cloud using IoU agreement between annotated bounding boxes and GT. BBOX denotes the accuracies for bounding boxes projected in the image while BEV (Bird Eye View) denotes the accuracies for bounding box from the top view of point cloud scene. *The IoU agreement between KITTI labels and GT labels is 72.65%.	32



# CHAPTER 1

---

## Introductions

---

### 1.1 Overview

The ever increasing volumes of point cloud data and the advances of automatic perception systems, has spawned new research targeting approaches to the description, analysis, and understanding of 3D scenes. This new research stream contributes to the development of automated systems using 3D data by adding realm of point cloud data modelling (including semantic mapping and object detection) to the previous focus on image analysis and hyperspectral data (Qi et al., 2017b; Li et al., 2018; Shi et al., 2019). The point cloud measurement technology can capture a very detailed (high resolution) three-dimensional image of objects and their environments. In fact, the point cloud data have already been used to generate high accuracy digital terrain models required for hazard assessment, susceptibility mapping, and detecting surface displacements (Jaboyedoff et al., 2012). There is also a substantial potential for improving the quality of our automated perception systems by using point cloud data, especially in the field of 3D scene understanding. Therefore, development and systematic testing of new methodology is required to release this potential.

Many researchers from computer vision, remote sensing, and automated systems have explored machine learning-based approaches to provide robust and accurate ways to extract (meaningful) information from the point cloud data. In fact, several *recent* dissertations from Stanford (Qi et al., 2018b), ETH Zurich (Hackel, 2018), and Imperial College London (McCormac, 2018) all addressed this issue, aiming to improve the quality of our automatic perception system for point cloud data based on different choices of deep learning architectures.

The success of deep learning for image analysis surpassing human-level performance (He et al., 2016) has attracted considerable attention in the last couple of years because of its potential for improving the quality of (automated) perception systems. In point cloud classification, the deep learning modelling approach has also shown superiority by providing the best performing classifiers for several point cloud classification benchmarks, such as 3D Shapenets (Wu et al., 2015), ScanNet (Dai et al., 2017), S3DIS (Armeni et al., 2016), and ISPRS Vaihingen 3D Labeling (Niemeyer et al., 2014).

In this thesis, we focus on a similar challenge: To provide high-quality annotation from large scale point cloud data using deep learning-based modelling techniques. We addressed the question of what the best way is to provide high accuracy

## 1. Introductions

---

annotation of point cloud data? We approached this question from three different viewpoints, (1) by contributing to the state of the art methodology for 2D image segmentation in generating semantic maps based on (2D projected) point cloud data, (2) by employing deep learning modelling techniques specifically developed for raw point cloud data handling *directly*, and (3) by combining deep learning-based modelling and minimal human-perception in our proposed semi-automatic annotation tool capable of generating fast and accurate point cloud labels (Arief et al., 2018, 2019b,a, 2020).

### 1.2 Research Questions and Objectives

As the amount of available point cloud data is growing, the efforts to extract meaningful information and accurate annotations also increases. Relying solely on humans to provide the point cloud annotations is not only expensive and time-consuming but can also result in inconsistent outcomes. Moreover, a multitude of applications requires that annotations are provided and updated in near real-time. Thus, it is a necessity to develop automatic (or semi-automatic) approaches to provide and maintain these high-accuracy point cloud annotations.

The overall goal of this thesis has been to provide an effective way to provide accurate annotations from large amounts of point cloud data. Such annotations were provided in the form of point-level classification, bounding box localization, and the combination of both.

The main research question of this thesis has been: *What are the (most) efficient and effective ways to generate high accuracy point cloud annotations?* We have tried to address and answer this question from three perspectives.

1. **2D Projections.** Based on emerging researches and high-quality results from 2D image understanding in computer vision, our first approach was to embody the techniques from this field to generate automatic segmentation based on 3D point cloud data by projecting the point cloud data into the 2D grids. This raises the questions: (1) Which state of the art techniques for image understanding is suitable for this approach, (2) How accurate are the existing techniques, and (3) How can we improve their accuracies and overcome their limitations?
2. **3D Representations.** One obvious problem with projecting point cloud data in the 2D grids is that meaningful information useful for automatic inference is lost. Therefore, the next objective of this thesis was to perform semantic segmentation *directly* from the 3D point cloud representations. This approach raises similar questions to the first approach: (1) What and (2) How accurate is the state of the art technique for generating point-wise segmentation directly from the point cloud data, and (3) How can we improve the accuracy of this technique?
3. **Semi-automatic Annotations.** Given the efficiency and the quality of the first two approaches, our last objective was to develop an annotation tool for generating high-accuracy point cloud labels, guided by automatic point-wise classification. This raises the questions: (1) How to efficiently annotate the 3D point cloud data, (2) What are the obstacles to providing



robust annotation tools for point cloud data, and (3) How can we overcome those problems?

By addressing these more detailed research questions, we have contributed towards our main objective, to provide efficient and accurate ways to generate high accuracy point cloud annotations.

### 1.3 Contributions and Thesis Outline

The contributions of this thesis have been published in several peer-reviewed papers on high-impact journals and conferences. The key papers are included in appendices A, B and C+D. The rest of the thesis is outlined as follows:

Chapter 2 provides background material. Previous work and current advances in the deep learning techniques for point cloud data are presented in this chapter. We also present the state of the art in automatic perception techniques for 3D scene understanding. The point cloud datasets used as research material in this thesis are also described here.

Our own research is described in Chapter 3. In the beginning of this thesis, we proposed a deep learning architecture for generating high quality semantic segmentation maps from a 2D projection of large amounts of point cloud data (Arief et al., 2018). Our proposal combined both Light Detection and Ranging (LiDAR)-derived features and image-based features. The results are good and applicable for updating the existing segmentation maps (NIBIO, 2018). However, the preprocessing techniques, including point cloud projections and height-above-ground (HaG) feature generations, are expensive and time-consuming.

Alleviating the complex preprocessing pipeline from our first approach (Arief et al., 2018), the XCRF algorithm was introduced (Arief et al., 2019b). In combination with the PointCNN (Li et al., 2018) we were able to simplify the generation of high accuracy semantic segmentation maps *directly* from the point cloud data. This methodology provided the highest F1-score on the tested benchmark dataset (Niemeyer et al., 2014). However, when the model derived by the proposed approach was applied to a (very) different dataset, the promising results could not be replicated, indicating a domain adaptation problem.

To overcome the domain adaptation problem and provide a robust point cloud annotation tool, we introduced SAnE (Arief et al., 2020). The SAnE uses a semi-automatic approach combining automatic perception (Arief et al., 2019a) and human involvement to obtain faster and more accurate annotation of point cloud data.

These approaches cover all the research questions stated in the previous section and contribute to the development in the fields of computer vision and remote sensing.

**Paper A (Section 3.1).** Here we propose a deep learning fusion architecture, combining LiDAR-derived features and image-based features for generating high quality land cover segmentation maps.

## 1. Introductions

---

- The proposed deep learning architecture integrates the deep atrous network architecture including the stochastic depth approach for speeding up the learning process while causing a regularization effect.
- By introducing an early fusion deep layer combining the image-based and LiDAR-derived features, we obtained more than a 5% improvement in the measured relative Mean Intersection over Union (MIoU) compared to the atrous network (Vladimir, 2018).
- By including the multi-class Intersection over Union (IoU) loss function in our implementation, the resulting model became better in handling highly imbalanced datasets, preventing against overfitting.

**Paper B (Section 3.2).** Our contribution is the development of a point cloud post-processing module, emphasizing the spatial autocorrelation of unlabeled data and neighborhood embedding to generate high-accuracy pointwise segmentation maps. Our propositions include:

- A novel technique for addressing the overfitting issue for automatic point cloud classification.
- A technique for utilizing unlabeled data to refine a validated deep learning model.
- A post processing module, A-XCRF, that can be combined with any machine learning technique to strengthen classification accuracy.
- Our approach yields the highest accuracy in term of F1-score (71.05%) for the Vaihingen 3D labelling dataset (Niemeyer et al., 2014).

**Paper C (Section 3.3) and Paper D (Section 3.4).** Finally, we have contributed to the development of robust open source point cloud annotation tools for generating fast and accurate point cloud annotation labels. Our propositions include

- A density-adaptive sampling, enabling the pointwise segmentation algorithm for heterogeneity density point cloud data.
- A denoising pointwise segmentation strategy, enabling the one-click annotation technique.
- A motion model approach using our novel guided-tracking algorithm, simplifying the frame-to-frame annotation process.
- A robust interactive open-source point cloud annotation tool for simplifying the creation of high-quality bounding box annotations.
- Annotation using our method speeded up the annotation process by a factor of 4.17 and provided annotation accuracy in term of Intersection over Union (IoU) agreements of 92.02% and 82.22% with 2D bounding box (BBOX) and Bird Eye View (BEV), respectively. A more carefully executed annotation based on the same tool even achieves +8.84% higher BEV IoU agreement than the baseline annotation accuracies.

## 1.4 Publications

The work and results described in this thesis are based on the following publications addressing several challenging problems concerning 3D point cloud annotation by classification modelling based on deep learning:

- **Paper A:** *Arief H, Strand GH, Tveite H, Indahl U. Land Cover Segmentation of Airborne LiDAR Data Using Stochastic Atrous Network. Remote Sensing. 2018 Jun 19;10(6):973.*
- **Paper B:** *Arief H, Indahl U, Strand GH, Tveite H. Addressing Overfitting on Point Cloud Classification using Atrous XCRF. ISPRS Journal of Photogrammetry and Remote Sensing (ISPRS Journal). Sept 2019. pp 90-101.*
- **Paper C:** *Arief H, Arief M, Bhat M, Indahl U, Tveite H, Zhao D. Density-Adaptive Sampling for Heterogeneous Point Cloud Object Segmentation in Autonomous Vehicle Applications. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, pp. 26-33. 2019.*
- **Paper D:** *Arief H, Arief M, Zhang G, Indahl U, Tveite H, Zhao D. SAnE: Smart annotation and evaluation tools for point cloud data. Submitted on Nov 2019.*



## CHAPTER 2

---

# Background

---

Recent advances in LiDAR has enabled higher quality 3D scene representations. This has fueled up a new research stream in 3D scene understanding, enriching the automatic perception research area, that previously focused on 2D vision and image processing (Krizhevsky et al., 2012; Simonyan and Zisserman, 2014; He et al., 2016), with point cloud data modelling and 3D scene understanding (Qi et al., 2017a,b; Li et al., 2018; Shi et al., 2019). Several datasets containing point cloud data have recently been published both for remote sensing (Niemeyer et al., 2014; Blom Geomatics AS, 2014; Hackel et al., 2017) and autonomous vehicle applications (Geiger et al., 2013; Chang et al., 2019; Waymo, 2019).

The point cloud datasets used in this thesis are described in Section 2.1. In Section 2.2, automatic perception techniques for 3D scene understanding are introduced, and then previous work and current advances in deep learning techniques for point cloud data are presented in Section 2.3.

### 2.1 Datasets

Compared to image-based datasets (Russakovsky et al., 2015; Everingham et al., 2015), point cloud datasets are significantly larger in volume. This is *mostly* because the point cloud data can capture 3D scenes better than the high-resolution images. It should be noted that the more dense the point cloud data are, the better their in-depth representation of the 3D view, resulting in larger amounts of data.

In this thesis, we used a reasonably small point cloud dataset, the Vaihingen dataset, for 3D semantic labeling (Niemeyer et al., 2014). This dataset contains a few hundred thousand data points. We also used a larger dataset, the Follo 2014 LiDAR dataset (Blom Geomatics AS, 2014), containing approximately eight billion points. Other datasets that were also used in this thesis, were the NIBIO AR5 land cover / land use maps (NIBIO, 2018), the Bergen dataset (Norwegian Map Authority, 2016), and the KITTI Vision Benchmark dataset (Geiger et al., 2013).

**Follo LiDAR 2014** (Blom Geomatics AS, 2014). The Follo dataset (Blom Geomatic AS, using a Riegl LMS Q-780, with 5 points/m<sup>2</sup>, covering 850 km<sup>2</sup>) was used in Paper A (Section 3.1) for generating land cover segmentation maps from point cloud data projected into a grid / image structure (Arief et al.,

## 2. Background

---

2018). The dataset covers Follo (around 819 km<sup>2</sup>, a part of Akershus county in Norway. The area is moderately hilly and dominated by forest with large patches of agricultural areas and small areas of settlement). See Fig. 2.1 and 2.2.

The Follo 2014 dataset has both LiDAR-derived features (X, Y, Z-coordinates, intensity, number of returns, and more), and image (RGB) features. It was stored in 1877 files (structured as tiles) in LAZ (LAS compressed files) format. Each tile covers an area of 600 m x 800 m, containing more than 2.4 million data points each.



Figure 2.1: Map of Norway (1:15M, UTM Zone 33N) showing the location of the Follo area.

**NIBIO AR5 land cover / land use map** (NIBIO, 2018). The AR5 land cover / land use maps were used in Paper A (Section 3.1) as the ground truth data for training and validating the SA-NET deep learning method (Arief et al., 2018). The dataset consists of several types of classification: land type ("arealtype" in Norwegian - a combination of land cover and land use), forest productivity, tree type, and ground conditions. We used the "arealtype" classification, with 11 classes (Ahlstrøm et al., 2019). Some of the classes did not exist or covered very small areas, so the number of classes was reduced to eight: settlement, road/transportation, cultivation/grass, forest, swamp, lake-river, ocean, and *other*.

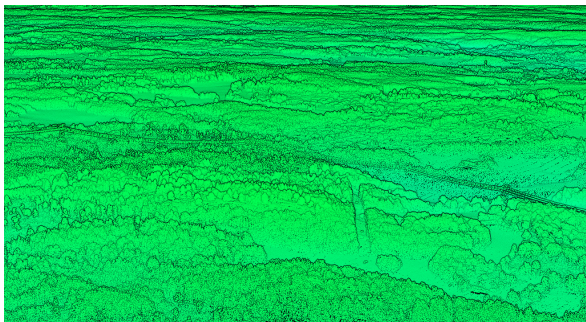


Figure 2.2: Example from the Follo point cloud - forested area.

Class	Number of Points	
	Training Data	Test Data
Powerline	546	-
Low Vegetation	180,850	-
Impervious Surfaces	193,723	-
Car	4,614	-
Fence/Hedge	12,070	-
Roof	152,045	-
Facade	27,250	-
Shrub	47,605	-
Tree	135,173	-
Total	753,876	411,722

Table 2.1: Class distribution of the Vaihingen 3D semantic labeling dataset.

**The Vaihingen dataset for 3D semantic labeling** (Niemeyer et al., 2014). This dataset is provided by ISPRS WG II/4, and was used in Paper B (Section 3.2) both as input and label data for 3D point cloud semantic labeling (Arief et al., 2019b). It is a 3D point cloud covering a part of Vaihingen village in Germany, acquired using a Leica ALS50 system. The dataset has a point density of 6.7 points per  $\text{m}^2$ , and has nine classes provided by Niemeyer et al. (2014). The classes are powerline, low vegetation, impervious surface, car, fence/hedge, roof, facade, shrub, and tree.

The Vaihingen point cloud dataset consists of more than one million data points, divided into training and test data, containing 753,879 points and 477,722 points, respectively. The data are stored in CSV files, containing X-, Y- and Z-coordinates, intensity values, number of returns, and class-id from one of the eight classes. Along with the dataset, the ISPRS has since 2014 provided a benchmark for 3D semantic labeling to compare the state of the art techniques in this domain.

**Bergen LiDAR Dataset** (Norwegian Map Authority, 2016). The Bergen dataset was used in Paper B (Section 3.2), as a transfer learning dataset for 3D

## 2. Background

---

semantic labeling (Arief et al., 2019b). The dataset was acquired using a Riegl Q-1560 mounted on a Piper Aircraft P-31-350, and covers the Bergen region in western Norway (see Fig. 2.3). It contains 3D spatial coordinates (XYZ) and RGB values.

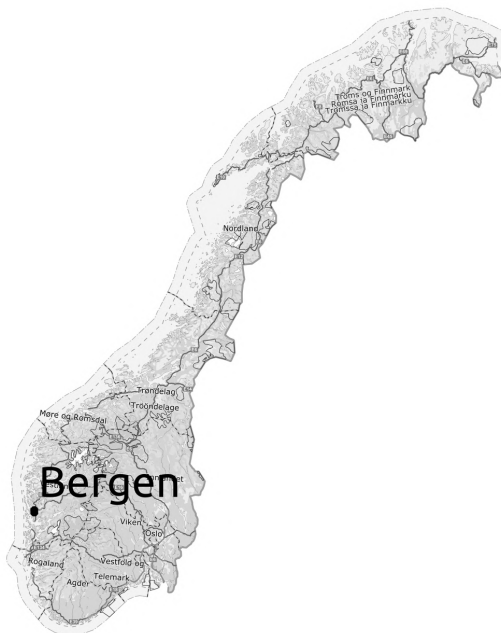


Figure 2.3: Map of Norway (1:15M, UTM Zone 33N) showing the location of Bergen.

The Bergen classification schema contains eight classes, including ground, low vegetation, medium vegetation, high vegetation, building, water, bridge, and snow/ice. We only used 100 tiles of the dataset, containing 719,762,528 data points, for the transfer learning experiment (Arief et al., 2019b).

**The KITTI vision benchmark dataset** (Geiger et al., 2013). The KITTI dataset was used both in Paper C (Section 3.3) and Paper D (Section 3.4) for 3D semantic segmentation (Arief et al., 2019a) and object detection (Arief et al., 2020), respectively. The dataset used for object detection contained 7481 scenes, and each scene has (on average) 1.3 million data points. The data points were collected using a Velodyne HDL-64E rotating 3D laser with 64 beams at 10 Hz.

The labels of the KITTI dataset is provided as bounding box locations, containing center coordinates, dimensions, rotation angle, and object-id with class reference. For the 3D semantic segmentation task, we preprocessed the data by assigning the class label of a box to all the data points that it contained. We



used the KITTI tracking dataset for object detection to show the applicability of our guided tracking algorithm proposed in Paper D (Section 3.4).

## 2.2 3D Scene Understanding

3D scene understanding can be viewed as an automatic perception of geometric structure and semantic information in a 3D scene, including the ability to recognize objects and estimate the layout of the scene. It is a broad topic in 3D computer vision, involving object recognition, layout estimation, semantic segmentation, motion estimation, and more (Qi et al., 2018b). In this thesis, we focus on two tasks, namely semantic segmentation and object detection.

### 2.2.1 Semantic Segmentation

In (2D) image space, semantic segmentation can be defined as dense pixel classification, where each pixel of the image is assigned to one class (Arief et al., 2018). Similarly, in a 3D point cloud scene, semantic segmentation is used to assign a class to each point in the scene, see Fig. 2.4. Thus, it is also called pointwise classification or point-level segmentation.

Generating a high accuracy pointwise classification is not a trivial task. Not only because it is difficult to handle large volumes of point cloud data in a computational sense, but it is also (almost) impossible to deterministically come up with a generic pattern that describes the irregular, unordered, and (sometimes) not scaled point cloud representation for a specific classification schema. Much research has been conducted to address these challenging problems, e.g. by generating geometrical features (such as sphericity, deviation angle, and planarity) and textural properties from point cloud data (Horvat et al., 2016; Steinsiek et al., 2017; Yang et al., 2018), then feeding them to a machine learning model such as K-Nearest Neighbour (Steinsiek et al., 2017) or Random Forest (Niemeyer et al., 2014) to perform the pointwise classification.



Figure 2.4: Point-level classification from the Bergen dataset.

## 2. Background

---

Data-driven feature learning, like deep learning, is also used to do pointwise classification. In fact, several 3D semantic segmentation benchmarks, such as 3D ShapeNet (Wu et al., 2015), ScanNet (Dai et al., 2017), S3DIS (Armeni et al., 2016), and the ISPRS Vaihingen 3D labelling (Niemeyer et al., 2014), show that deep learning-based techniques are (among) highest performing classifiers (Qi et al., 2017a,b; Li et al., 2018; Arief et al., 2019b), providing high accuracy semantic maps.

With the ability to provide accurate semantic mapping, the state of the art semantic segmentation techniques are frequently deployed both in the field of remote sensing and autonomous vehicle applications. In remote sensing, the techniques are used for generating land cover and land use maps (Yousefhusien et al., 2017; Yang et al., 2017; Arief et al., 2018, 2019b), road detection (Caltagirone et al., 2017; Brust et al., 2015), water body extraction (Yu et al., 2017; Kemker et al., 2018) and crop yield prediction (Payne et al., 2013; Milioto et al., 2018). For autonomous vehicle applications, they are used for environmental mapping and make up the backbone for object detection and localization tasks (Yang et al., 2019a; Shi et al., 2019; Yang et al., 2019b; Arief et al., 2020).

### 2.2.2 Object Detection

3D object detection, also called object localization, is used to determine the location of objects in 3D space, represented using bounding boxes and/or centroids. In contrast to semantic segmentation, object detection assigns a unique object-id with a corresponding class name to each object. Each object, in the 3D object detection task, will normally contain much more than one point from the point cloud dataset, see Fig. 2.5.

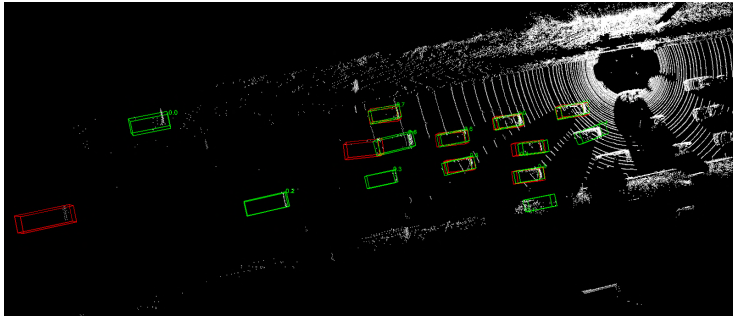


Figure 2.5: 3D object detection based on PointRCNN (Shi et al., 2019).

In remote sensing, object detection techniques are used for building extraction, tree classification, and pedestrian detection (Mnih, 2013; Du et al., 2017; Demir et al., 2018; Van Etten et al., 2018). In other fields, like virtual reality and autonomous driving, this research area plays an even more vital role as the backbone of the applications. Because of its importance, researchers have proposed many approaches to perfecting the solutions for generating high

accuracy bounding box and object location. Lahoud and Ghanem (2017) proposed the use of histogram-based point coordinates to derive a 3D bounding box location using a fully connected network. Zhou and Tuzel (2018) used voxel grids to represent unordered point cloud data and implement voxel feature encoding to perform object detection. Qi et al. (2018a) combined point cloud feature learning and image-based object detection to detect object locations in 3D space.

### 2.3 Deep Learning for Point Cloud Data

Deep learning is a term that refers to a deep layer neural network, a machine learning algorithm that has been around for decades. A Neural Network (NN) algorithm tries to replicate the way the human brain works by providing neurons and activation functions that are used to make decisions, similar to the cat's visual cortex experiments by Hubel and Wiesel (1962).

An early neural network architecture called LeNet-5 (LeCun et al., 1998) provides a foundation for modern neural networks. The LeNet modules, such as convolution layers, sub-sampling/pooling layers, activation function, and fully connected layers are still widely adopted in current neural network architectures. However, compared to the current standard and the result of other machine learning algorithms, the old neural network results were significantly lower than the present state of the art.

In 2015, the technology also reached a new level of success, a deep neural network technique started to surpass human-level performance on visual recognition challenges (He et al., 2016). And as the technology matured, the focus shifted to more complex challenges, such as semantic segmentation, object detection, and instance-aware segmentation (Chen et al., 2017).

The following chapters review the deep learning techniques applied to point cloud data, such as deep learning on 2D projected point cloud with image-based convolutional neural networks, deep learning for raw point cloud data, and deep learning for semi-automatic annotation.

#### 2.3.1 2D Projection

The deep learning approaches for 2D projected point cloud data are similar to the deep learning techniques for image segmentation. The main difference is that instead of using RGB values as input, the 2D projected point cloud data use normalized height and intensity values. It should be noted that the projected data can also use RGB values if these values are available.

Treating point clouds as image data enables the use of the Convolutional Neural Network (CNN) based approach for automatic feature learning. It is an important step, because CNN-based techniques can provide high-accuracy predictions for (image-based) semantic segmentation problems (Long et al., 2015; Ronneberger et al., 2015; Badrinarayanan et al., 2017; Chen et al., 2017).

**Convolutional Neural Network.** The CNN can be viewed as a stack of learning blocks capable of capturing various spatial correlations, while at the same time being inherently qualified to represent non-linear phenomena. CNN

## 2. Background

---

works by using three main modules, namely the convolution operation, the pooling block, and a non-linear activation function.

The convolution operation is a dot product operation between input feature maps and initially random numbers in a fixed size matrix, called kernel. The kernel convolves on top of the feature maps, generating new feature maps, hence the name convolution kernel ( $k$ ), see Fig. 2.6. The convolution operation uses a stride ( $s$ ) to define the movement of the convolution kernel on top of the feature maps; for 2D convolution, the stride is represented by two numbers defining the kernel movement on the  $X$  and  $Y$  axis. In addition to stride value and the kernel size, the convolution operation is defined by padding ( $p$ ). The value of ( $p$ ) is used to fill the empty pixels caused by the striding of the convolution kernel on top of the input feature maps.

The pooling block, on the other hand, acts as a local feature aggregation which summarizes nearby features using an agreed value. Using the maximum value of the nearby features is called max-pooling (see Fig. 2.7), while using the average value of the nearby features is called average-pooling. The pooling block not only reduces the spatial size of feature maps, which significantly lowers the usage of memory storage and computational load but also invokes an aggregation of the feature of interest and use them as the global feature representation.

Another important component of CNNs is the non-linear activation functions. Without such functions, CNN layers would become just “one big linear sandwich” which cannot represent the nonlinear phenomena that are important in many classification tasks (Minsky and Papert, 1988). CNN uses non-linear functions such as a Rectified Linear Unit (ReLU) (Dahl et al., 2013) or one of its variants. The ReLU clips all the input values that are below zero ( $< 0$ ) and outputs them as zero, while returning all other values ( $\geq 0$ ), see Fig. 2.8. An important property of a ReLU is that it outputs zero for half of the values and keep the remaining values whenever the unit is active. This property makes a ReLU favorable because it makes the model easy to optimize, while keeping the computational costs low.

**2D Semantic Segmentation.** For the semantic segmentation task, a CNN

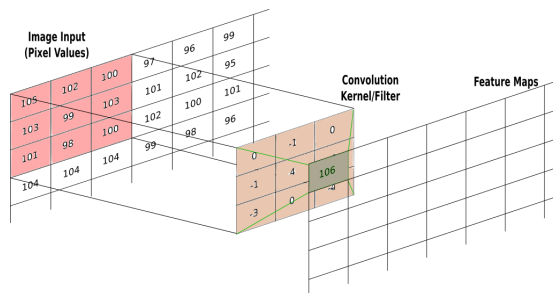


Figure 2.6: Convolution operation using 2D kernel.

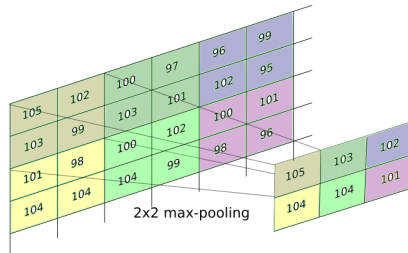


Figure 2.7: The max-pooling operation using a 2D kernel.

is equipped with upsampling modules capable of generating larger feature maps from smaller ones, such as unpooling and transposed convolution layers (Arief et al., 2018). The unpooling operation remaps the downsampled feature maps using max-indices from its downsampling procedure to recover the pre-sample feature maps (in the original spatial resolution). Max-indices contain information about which pixel index was used to represent the nearby pixels. From this information, the operation can recover the “original” spatial resolution of a feature map without losing the spatial connectivity of their previous process.

Transposed convolution, on the other hand, works by enhancing the dot product and sum operations of the upsampling kernel on a smaller feature map (with extra padding) to generate larger feature maps. This approach forces the upsampling kernel to fill in the padding values with more meaningful information, often reflected in a lower classification loss. This is because the upsampling kernel is updated during the parameter update operation, so its values will also

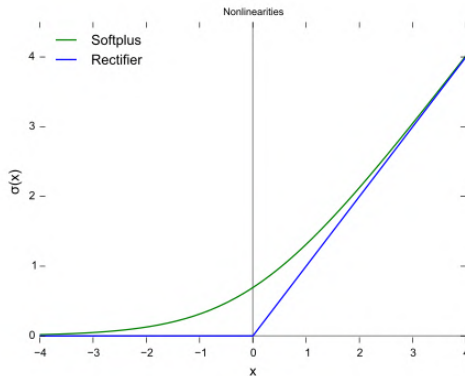


Figure 2.8: Rectifier unit (blue line). Image source: Dan Stowell (distributed under a CC0 license).

## 2. Background

---

reflect the final loss that the deep learning model generates.

A Fully Connected Network (FCN) (Long et al., 2015) is one of the few deep learning architectures that adopted upsample convolution at an early stage. The idea is that instead of crude oversampling of the feature maps, learning the oversampling procedure using the convolution operation could be more meaningful. FCN also introduces skip connections for the upsampling process to enrich and tighten the spatial connection between input data and the final prediction. In fact, an enhanced version of FCN implements the skip connection idea on all downsampled layers and then generates cascaded and hierarchical upsampled feature maps to provide a more robust and high-accuracy classifier.

In addition to the FCN, there are several deep learning architectures specifically designed for semantic segmentation. SegNet (Badrinarayanan et al., 2017) uses the stacked of pooling-unpooling layers to perform upsampling feature generation from input data. Ronneberger et al. (2015) designed u-net with multi-level skip connections, structured as the letter U, generating high accuracy semantic maps. Finally, the atrous network from deeplab (Chen et al., 2017) uses a cascade of atrous kernel to learn spatial autocorrelation from image data using convolutional layers with wider reach. It should be noted that the atrous kernel, also called dilatation kernel, is a convolution kernel with distance space (called rate) between each kernel value in a matrix space, see Chapter 3.1 for more details.

### 2.3.2 3D Point Cloud Representation

3D point cloud data can come as coordinate list (COO), containing spatial coordinates (X, Y, Z), intensity values, and sometimes RGB colors and class labels, e.g. {x, y, z, i, r, g, b, c}. However, the COO format doesn't have neighboring information among points that are important for utilizing the spatial autocorrelation, required in many automatic perception algorithms. Typical search trees, like KD-Trees (Bentley, 1975; Sproull, 1991) are deployed to identify the neighbors of points. However, compared to the neighboring search complexity in 2D images  $O(1)$ , the tree search is a very expensive operation  $O(\log(n))$ , especially when the number of points ( $n$ ) is very large.

**Feature Learning.** In 2016, Qi et al. (2017a) introduced the PointNet architecture, a deep learning model capable of generating (robust) feature representations directly from (unordered) point cloud data. This work shows that a typical Neural Network represented as Multi-Layer Perceptron (MLP), can generate a reasonable feature projection of the un-order point cloud using the T-NET architecture, resulting in a powerful point cloud classifier. A hierarchical version of PointNet, called PointNet++ (Qi et al., 2017b), provides higher accuracy than the original PointNet. It should be noted that there are other point cloud feature learning techniques proposed to address similar problems, i.e Hypervoxel (Mughees and Tao, 2017), SuperPoints (DeTone et al., 2018), and 3D Convolution (Li, 2017). However, these other learning algorithms require a more complicated pre-processing procedure than the ones that can learn (directly) from the raw point cloud data (Qi et al., 2017a,b).

**PointCNN.** Li et al. (2018) proposed the PointCNN with the xConv operator, enabling a (typical) convolutional operation to be directly applied to the

point cloud data. Similar to the PointNet approach, PointCNN uses an MLP architecture for feature learning. The main difference is that the PointCNN applies MLP by (first) embedding each point with their neighbor points (enriching feature representations for each of the input points). The PointNet ignores the neighborhood feature embedding.

**Spatial Smoothing.** Emphasizing neighbor points when building high-order feature maps can also be applied as a post-processing approach for semantic segmentation. This approach is called random field models, enforcing spatial smoothing on a neighborhood of data. The idea is that data points that are close to each other and have similar features of interest, should have the same class label. Krähenbühl and Koltun (2011) proposed the use of Conditional Random Field (CRF) similarity penalties using Gaussian filtering by treating image pixels as a fully connected graph for the random field smoothing. Niemeyer et al. (2014) also used the CRF idea to provide semantic labeling on 3D point cloud input data. Other researchers, like Zheng et al. (2015) (even) deployed a Recurrent Neural Network (CRF), treating the CRF model as a deep learning architecture.

### 2.3.3 Semi-automatic Annotation

Semi-automatic annotation is an approach to combine the automatic perception algorithms with human-based perception, also called human-in-the-loop. This is because, even though the automatic perception techniques can generate accurate predictions, (most often than not) these predictions are still nowhere (near) perfect. Meanwhile, in many modern applications, very high accuracy annotations are required to ensure safety and applicability.

(Hurl et al., 2019) shows that a better-annotated and bigger dataset contributes results in a higher quality machine learning model. It should be noted that real-world datasets are limited in size and accuracy compared to synthetic datasets, but the current synthetic datasets are not (fully) domain transferable (Arief et al., 2019b). Manual annotations, however, are (often) expensive and contain erroneous labels (Wang et al., 2019). Therefore, it is important to provide semi-automatic annotation tools capable of delivering fast and high-accuracy annotation labels.

Castrejon et al. (2017) proposed PolygonRNN, a semi-automatic annotation tool for image segmentation, leveraging polygon vertices outlining the annotated object, delivering faster and more accurate annotations. In 2018, PolygonRNN++ was published by Acuna et al. (2018), an enhanced version of PolygonRNN providing more than 10% higher accuracy and faster annotation than the original PolygonRNN.

Several annotation tools for point cloud data, that provide high accuracy point cloud labels, have also been published i.e 3D-Bat (Zimmer et al., 2019) and Latte (Wang et al., 2019). The 3D-Bat provides fully functional point cloud annotation tools from keyboard only annotation, to semi-automatic tracking interpolations (Zimmer et al., 2019). Latte, on the other hand, offers a robust annotation toolbox with smart automatic perception algorithms, like semantic segmentation with MaskRCNN (He et al., 2017), point cloud classification

## 2. Background

---

with GoogleNet (Szegedy et al., 2015), and object tracking with Kalman Filter (Welch et al., 1995).

### 2.4 Evaluation Metrics

One of the most common metrics used for pixel classification or point-level segmentation tasks is the pixel/point accuracies (PA), also called Overall Accuracy (OA). The PA has been used in many classification tasks (Long et al., 2015; Everingham et al., 2015; Russakovsky et al., 2015). For segmentation, the accuracies are commonly measured using mean pixel accuracies (MPA), mean intersection-over-union (MIoU), and the F-Measure (F1 Score). It should be noted that for segmentation problems where the total area of the classes is very different (imbalanced), the PA measure is less informative. This is because assigning all of the pixels to the largest class may result in a large PA value, even without training a model.

With  $k + 1$  being the total number of classes (including the background class) and  $p_{ij}$  denoting the number of pixels from class  $i$  assigned to class  $j$ , the accuracy measures PA, MPA, and MIoU are defined (Garcia-Garcia et al., 2017) as follows:

$$PA = \frac{\sum_{i=0}^k p_{ii}}{\sum_{i=0}^k \sum_{j=0}^k p_{ij}}, \quad (2.1)$$

$$MPA = \frac{1}{k+1} \sum_{i=0}^k \frac{p_{ii}}{\sum_{j=0}^k p_{ij}}, \quad (2.2)$$

and

$$MIoU = \frac{1}{k+1} \sum_{i=0}^k \frac{p_{ii}}{\sum_{j=0}^k p_{ij} + \sum_{j=0}^k p_{ji} - p_{ii}}. \quad (2.3)$$

The MIoU equation shows that the metric awards valid predictions ( $p_{ii}$ ) and a penalizes false negatives ( $p_{ij}$ ) and false positives ( $p_{ji}$ ).

The F-Measure has been used to better evaluate the boundary region of the predicted pixels (Badrinarayanan et al., 2017). We used the mean of the F-Measure per class to evaluate the performance of the classifiers. This metric considers both the precision ( $p$ ) and recall ( $r$ ) of the prediction results. With TP denoting the true positives, FP denoting the false positives, and FN denoting the false negatives, the F-Measure (F1 Score) is defined as:

$$p = \frac{TP}{TP + FP}, \quad (2.4)$$

$$r = \frac{TP}{TP + FN}, \quad (2.5)$$

and

$$F1Score = \frac{1}{k+1} \sum_{(i=0)}^k 2 * \frac{p * r}{p + r}. \quad (2.6)$$



## 2.4. Evaluation Metrics

---

For the sake of simplicity, we will refer to the pixel accuracy measure as PA or OA, the mean pixel accuracy as MPA, the mean intersection-over-union as MIoU, and the F-Measure as F1-Score.



## CHAPTER 3

---

# **Publications**

---

### 3. Publications

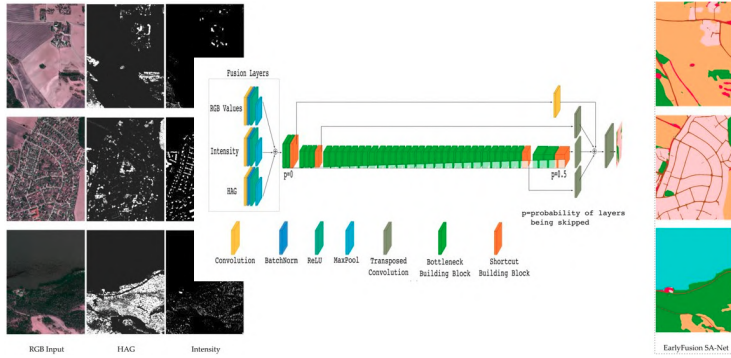


Figure 3.1: The early fusion SA-NET architecture.

### 3.1 Semantic Mapping for 2D Projected Point Cloud

For the purpose of generating high accuracy point cloud segmentation mask, the 3D point cloud data were projected onto 2D grids enabling the use of high accuracy deep learning algorithms for 2D images. During the research process, several deep learning architectures were considered, and their prediction results were compared and limitations were highlighted. Based on this work, we were able to take advantage of existing "state of the art" approaches on helping us generate high accuracy segmentation masks, by avoiding several of the classical limitations - such as high-memory consumption, (very) coarse up-sampling output, overfitting certain classes etc. It should be noted that the segmentation mask used, were low resolution maps compared to the high resolution input data, causing imperfect match on the co-registered dataset (NIBIO AR5 and Follo LiDAR data). Therefore, our resulting approach gave better predictions than the "state of the art" methods.

Confronting the dataset challenges and the existing technique limitations, we

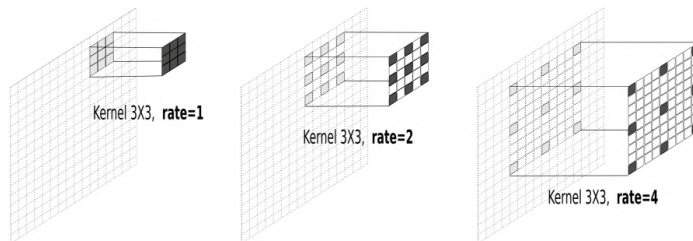


Figure 3.2: The 3 by 3 atrous kernel with different number of holes, defined using the value of *rate*.

### 3.1. Semantic Mapping for 2D Projected Point Cloud

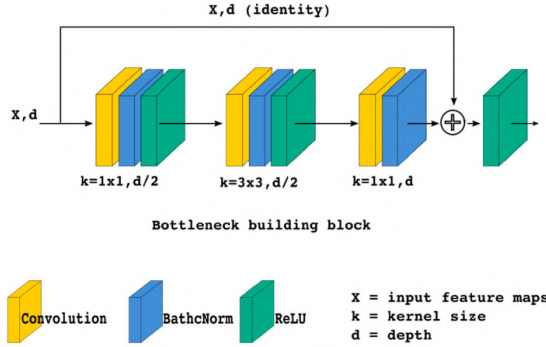


Figure 3.3: Residual connection using bottleneck building block.

ended up proposing the Stochastic Atrous Network architecture with its fusion layer (called earlyfusion SA-NET) , see Fig. 3.1. This architecture uses three enhanced deep learning modules, namely atrous kernel (Chen et al., 2017), residual layer (He et al., 2016), and stochastic learning (Huang et al., 2016).

1. The atrous kernel allows a wider (spatial) reach of a normal convolutional kernel, enabling a better (spatial) generalization while keeping the computation cost low. This approach is based on convolution kernels with holes, see Fig. 3.2, but otherwise works as usual by dot product operations, similar to the normal convolution operations in CNN.
2. The residual layer, with its shortcut connections (from the ResNet architecture), includes training of the deeper neural network layers by applying an identity information from the previous building block, see Fig. 3.3. This idea is important, because the deeper the neural network layers are located in the architecture, the risk of aggregated information loss would otherwise increase critically. Therefore, by using the residual approach, the total information loss of original input data is prevented, while allowing the deep neural network kernels to generate more efficient features.
3. The stochastic learning, on the other hand, acts as a catalyst to speed up the training process, while causing an advantageous regularization effect by randomly skipping the updating of some layers during training.

By using the SA-Net, the prediction accuracy in term of MIoU, improved by 5% compared to the original "benchmark" atrous network, providing a basis for better and improved use of LiDAR data for automatic image-based segmentation, see Table 3.1. Moreover, by inspecting the final prediction results and using local knowledge, we note that the prediction results are (actually) better than the labelled data. This indicate the usefulness of our proposal to streamline the

### 3. Publications

---

	PA	MPA	MIoU	F1
FCN-8s	93.36	69.62	64.97	73.05
SegNet	92.11	63.79	59.12	67.13
Atrous Network + CRF	90.97	61.12	56.70	63.50
Atrous Network (DeeplabV2)	92.28	67.60	62.81	70.79
<b>Earlyfusion SA-Net</b>	<b>93.96</b>	<b>73.00</b>	<b>68.51</b>	<b>75.81</b>

Table 3.1: The test result for 2D semantic segmentation. CRF: conditional random field; MIoU: mean intersection-over-union; MPA: mean pixel accuracies; PA: pixel accuracy.

maintenance workflow of the labelled data (NIBIO AR5 dataset) by efficiently directing the cartographer attention towards areas where changes/challenges are most likely to be found.

The SA-Net demonstrates the possibility of generating high accuracy segmentation maps from 2D projected point cloud data. However, the required data projection phase is clearly a very time consuming pre-processing step. For our application it required three days for a 64-core processor to perform the data projection for the Follo LiDAR data. Moreover, by projecting the 3D representations to a compact 2D grid, some information loss is inevitable, limiting the flexibility of the proposed approach in generating efficient segmentation maps.

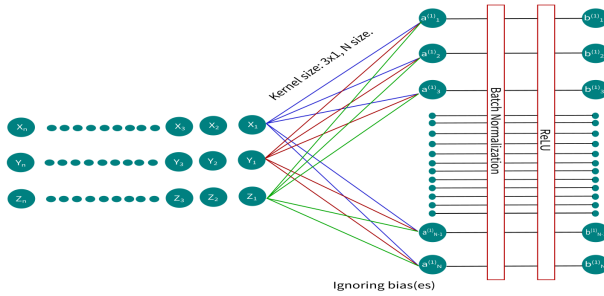


Figure 3.4: Point cloud feature learning with MLP.

### 3.2 Pointwise Segmentation for 3D Point Cloud Representations

As demonstrated in Paper A (Section 3.1), by projecting the 3D point cloud data into the 2D grids, a high accuracy segmentation map can be generated using the SA-NET. However, the preprocessing 2D-projection of the 3D point cloud data is sub-optimal and time-consuming. Following up on these issues, we decided to focus on training our high accuracy semantic segmentation maps *directly* by using the raw point cloud coordinates (x-y-z) as the (main) input data.

Similar to our first approach, we focus on the PointCNN as the baseline deep learning architecture for classifying the raw point cloud data. The PointCNN was developed from the idea of the T-NET (from the PointNet) into a more sophisticated feature learning algorithm called x-Conv. Both T-NET and x-Conv serve as feature extractors for a final Multi Layer Perceptron (MLP) block, see Fig. 3.4. The main difference between the two is that the x-Conv does (some) on-the-fly preprocessing before feeding the input data to the MLPs, by gathering and normalizing neighbouring points as the actual input features for each point, see Fig. 3.5.

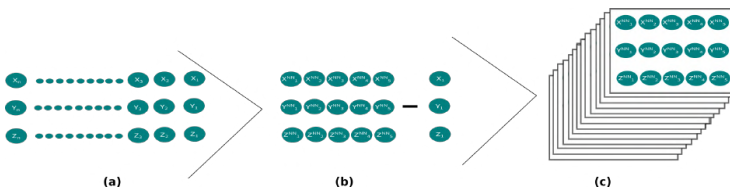


Figure 3.5: On-the-fly preprocessing on x-Conv algorithm: (a) point cloud input data with x,y,z dimension, (b) each point gather neighboring points and normalized their value, (c) final input data used for MLPs learning process.

### 3. Publications

Class	a	b	c	d	e	f	g	h
Powerline	54.4	46.1	<b>69.8</b>	59.6	37.5	42.5	61.5	63.0
Low-Veg.	65.2	79.0	73.8	77.5	77.9	<b>82.7</b>	<b>82.7</b>	82.6
Imper-Surfaces	85.0	89.1	91.5	91.1	91.5	91.4	91.8	<b>91.9</b>
Car	57.9	47.7	58.2	73.1	73.4	74.7	<b>75.8</b>	74.9
Fence	28.9	5.2	29.9	34.0	18.0	<b>53.7</b>	35.9	39.9
Roof	90.9	92.0	91.6	94.2	94.0	94.3	92.7	<b>94.5</b>
Facade	-	52.7	54.7	56.3	49.3	53.1	57.8	<b>59.3</b>
Shrub	39.5	40.9	47.8	46.6	45.9	47.9	49.1	<b>50.7</b>
Tree	75.6	77.9	80.2	83.1	82.5	82.8	78.1	<b>82.7</b>
Avg F1	55.27	58.96	66.39	68.39	63.33	69.2	69.5	<b>71.1</b>
OA	76.2	80.8	80.5	81.6	81.6	84.9	83.3	<b>85.0</b>

Table 3.2: A quantitative comparison between A-XCRF and other methods on the Vaihingen dataset, namely (a) ISS\_7 (Ramiya et al., 2016), (b) UM (Horvat et al., 2016), (c) HM\_1 (Steinsiek et al., 2017), (d) LUH (Niemeyer et al., 2016), (e) RIT\_1 (Yousefhussein et al., 2017), (f) WhuY4 (Yang et al., 2018), (g) PointCNN (Li et al., 2018), and (h) A-XCRF (Arief et al., 2019b). All cells except the last two rows show the per-class F1 score.

Our experimental results shows that the PointCNN prediction accuracy is comparable to other proposals for the benchmark dataset (the Vaihingen 3D labeling task), see Table 3.2. However, the PointCNN, as the other MLP architectures, does not utilize the spatial auto-correlation properties of the point cloud representation, which is obviously useful for modelling based on spatial data. PointCNN uses gradient descent algorithm and a cross entropy loss function in the weight optimization without utilizing spatial issues particularly. Therefore, we claim that an improvement can be made by emphasizing the importance of spatial and feature similarities between neighbors while doing the weight optimization process (based on the spatial auto correlation theorem).

Our key contribution in this work is to demonstrate that spatial auto correlation can be successfully combined with the PointCNN by using Conditional Random Field (CRF) as a post processing module. The main idea is that two points, that are (spatially) near each other and also have similar features, should belong to the same class. In the CRF graph we used the Gaussian bilateral and spatial

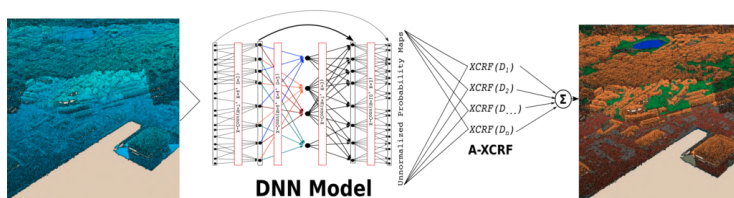


Figure 3.6: Full pipeline of A-XCRF technique using the PointCNN as the main deep learning architecture.



---

### 3.2. Pointwise Segmentation for 3D Point Cloud Representations

filter. The purpose of the bilateral filter is to invoke the feature similarities while the spatial filter is used to handle spatial similarities with the neighborhood points. We combined these modelling components into a refinement block, called A-XCRF, see Fig. 3.6.

An important property of the A-XCRF is that the module is trained with unlabelled data to introduce noise in the validated deep learning model and emphasize neighbourhood point similarities in the unlabelled data. The underlying assumption is that if the resulting model respect the neighbourhood point similarities for both training and unlabelled data, it can also produce high accuracy predictions not only for the training data but also towards the unlabelled data to prevent against overfitting. It should be noted that our experimental results show that the A-XCRF prediction accuracy is superior to the other proposals for the Vaihingen benchmark dataset, see Table 3.2.

We also tested the A-XCRF technique on the transfer learning- and domain adaptation problems to show the applicability of the resulting classifier for another dataset (the Bergen dataset) generated with a different LiDAR setting, in a different environment and landscape without (model) retraining. Interestingly, a consistent 3% improvement (in accuracy) could be achieved by using our approach for the new dataset, showing the importance of utilizing spatial auto correlation for modelling with spatial data. However, it should be noted that the improvement was limited and the resulting predictions were not at a level appropriate for production quality. This is makes sense because the Vaihingen and Bergen dataset are very different in both topography, landscape, and the number of data points. It should be noted that the number of data points for the Vaihingen training data was only 753,876, while the (tested) Bergen dataset has 719,762,528 data points (20% of the whole Bergen dataset).

### 3.3 Deep Learning for Point Cloud Annotations

The obtained accuracies of our two previous proposals seems to be the highest on their respective domains of approach. However, for the purpose of providing the most reliable way to generate high quality annotation labels for point cloud data, those proposals inherit some weaknesses, in particular for the transfer learning problem. Therefore, our next approach includes operation of humans to overcome the imperfectness of the machine-based predictions to provide fast and accurate point cloud labels.

In the present and the next section we focus on developing a robust semi-automatic annotation methodology for point cloud data on autonomous vehicle (AV) domain. The objective of the present section is to enable the automatic pointwise classification algorithms, like the PointCNN, to work on the heterogeneous point cloud densities, commonly generated by the mobile LiDAR, such as the Velodyne LiDAR for autonomous vehicle, see Fig. 3.7. Finally, in the next section, our focus is adjusted to utilizing the automatic classification approach for fully functional point cloud annotation tools.

The standard version of the PointCNN assumes that the density of the input data is homogeneous, therefore, its MLP parts can learn and generate representative features, resulting in high accuracy pointwise segmentations. However, in the AV domain, the assumption of homogeneity does not hold. There, the density of the point cloud data is higher near the ego-vehicle while decreasingly lower

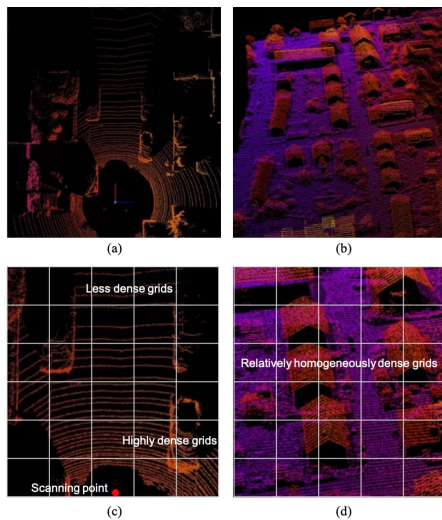


Figure 3.7: The nature of point cloud data from two different domains: (a) driving scene point cloud from Velodyne-type LiDAR (b) landscape map point cloud from airborne LiDAR along with point density distributions (c-d).

### 3.3. Deep Learning for Point Cloud Annotations

Method	<i>MPA</i>	<i>MIoU</i>
VB Block-10	0.6329	0.3369
VB Block-20	0.6999	0.5263
VB Block-30	0.8273	0.5717
Without oversampling		
GBR-Original	0.8895	0.6418
GBU-Original-XY-0.25	0.8641	0.6179
GBU-Original-XY-1	0.8731	0.6511
XY-GRID with Oversampling		
GBU-XY-grid-0.25	0.8876	0.5840
GBU -XY-grid-1	0.8983	0.6304
GBR -XY-grid-0.25	0.9152	0.6342
GBR -XY-grid-1	0.8881	0.6504
XYZ-GRID with Oversampling		
GBU-XYZ-grid-0.25	0.9026	0.6471
GBU-XYZ-grid-1	0.9040	0.6346
GBR-XYZ-grid-0.25	<b>0.9225</b>	<b>0.6816</b>
GBR-XYZ-grid-1	0.9217	0.6509

Table 3.3: The performance of each of the sampling scenario. VB: Voxel based sampling; GBU: Grid based uniform sampling; GBR: Grid based random sampling.

further away from the ego-vehicle.

To provide a robust semi-automatic point cloud annotation tool, our main contribution and stepping stone in this work is a (preprocessing) sampling approach, enabling pointwise segmentation algorithms for the AV domain. It should be noted that in our experiment, the PointCNN was used as the pointwise segmentation architecture.

The proposed sampling technique, on the other hand, is a grid-based sampling approach. The points inside each grid are over-sampled until the point density of each grid is equal to the overall average point cloud density. Thereafter, the point cloud data are randomly sampled from the complete collection of

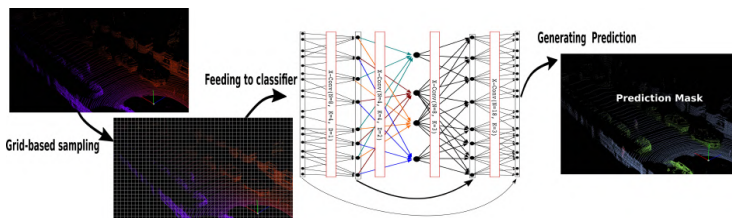


Figure 3.8: The density-adaptive sampling pipeline for semantic segmentation for heterogeneous density point cloud.

### 3. Publications

---

point cloud grids, before using them as training data for the PointCNN and the subsequent inference process. The proposed sampling approach is called density-adaptive sampling.

In Table 3.3, the experimental results show that with our proposal, the accuracies, in term of MPA and MIoU, improved by about 10% compared to the voxel-based sampling approach. In addition, some misalignment problems within the KITTI benchmark dataset (Fig. 3.9) are also pointed out, emphasizing the importance robustness required for our annotation tools to be able to generate fast and accurate bounding box labels.

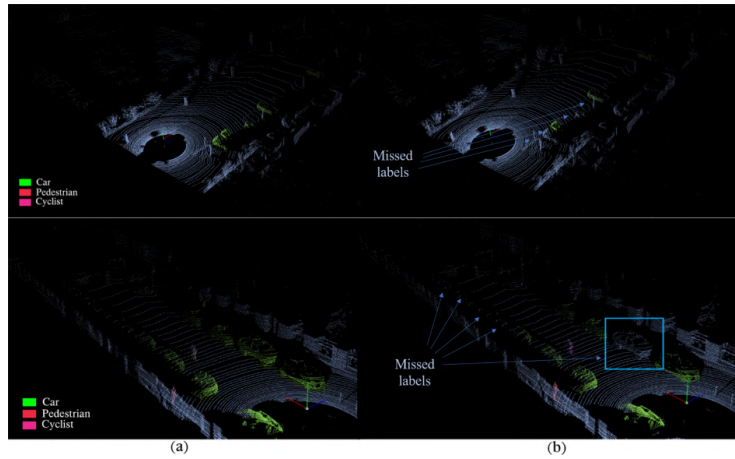


Figure 3.9: Point cloud visualization of the KITTI dataset with (a) Prediction results, and (b) Ground truth label with missing object bounding box.

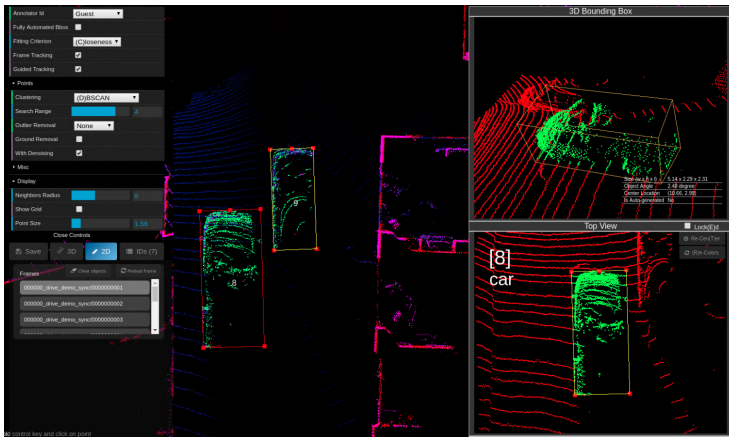


Figure 3.10: The interface of SANe, a semi-automatic annotation tool based on one-click annotation scheme empowered with denoising point-wise segmentation approach and robust guided-tracking algorithm.

### 3.4 Semi-automatic Point Cloud Annotation Tools

Starting from a workable pointwise segmentation for raw point cloud data, in this section, our focus is on developing the semi-automatic annotation tools. This is achieved by merging the automatic pointwise segmentation algorithm of the previous section with motion modelling and human based perception. While most semi-automatic annotation tools for point cloud data combine image- and LiDAR point cloud data as input, we here focus on the utilization of point cloud data alone to provide the desired high accuracy annotation labels.

Producing high accuracy bounding boxes from point cloud scene is a complicated and time-consuming task while the results are (often) inaccurate (Arief et al., 2019a; Wang et al., 2019). Here, we propose an annotation tool called SANe (Smart Annotation and Evaluation tools for point cloud data), capable of generating fast and accurate bounding box annotations, see Fig. 3.10. It should be noted that the SANe have been build on top of the Latte annotation tool by Wang et al. (2019).

Our main contributions are threefold: (1) we propose a denoising pointwise segmentation strategy enabling one-click annotation (see Fig. 3.11), (2) we expand the motion model approach with our novel guided-tracking algorithm, simplifying the frame-to-frame annotation process, and (3) we provide a robust interactive open-source point cloud annotation tool, simplifying the generation of high-quality bounding box annotations. Several enhancements were applied in the SANe, from the AI-assisted features (i.e fully automated bounding box proposal, one-click annotation, and guided-tracking algorithm) to the UI-based functionalities, such as side-view refinement, height estimation, keyboard-only annotation, object recoloring, and more.

### 3. Publications

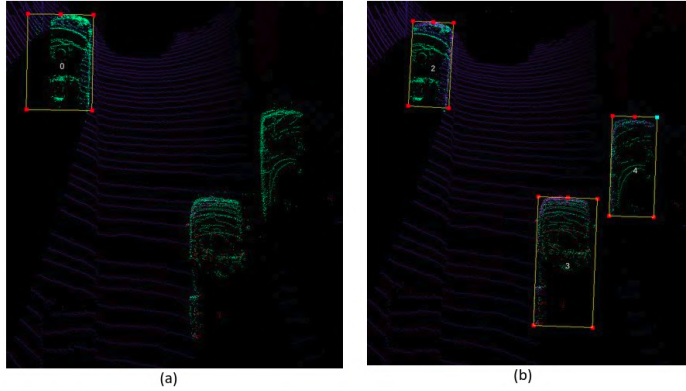


Figure 3.11: The results of denoising algorithm (a) before and (b) after the implementation. The algorithm enables the use of one-click annotation techniques for annotation tool.

	IoU value		vs Baseline		vs KITTI
	BBOX	BEV	BBOX	BEV	BBOX
Objects in $\approx 90$ -degree view					
GT	100.00	100.00	-	-	27.35
Baseline	89.49	76.35	0.00	0.00	16.84
Annot1	90.87	<b>85.19</b>	1.38	<b>8.84</b>	18.22
Annot2	<b>92.02</b>	82.22	<b>2.53</b>	5.87	<b>19.37</b>
Objects in 360-degree view					
Baseline	-	77.10	-	0.00	-
Annot1	-	<b>84.57</b>	-	<b>7.47</b>	-
Annot2	-	79.57	-	2.47	-

Table 3.4: Bounding box accuracies for objects in front of the ego vehicle and objects in the whole area of the point cloud using IoU agreement between annotated bounding boxes and GT. BBOX denotes the accuracies for bounding boxes projected in the image while BEV (Bird Eye View) denotes the accuracies for bounding box from the top view of point cloud scene.

\*The IoU agreement between KITTI labels and GT labels is 72.65%.

In table 3.4, the experimental results show that by using the SAnE, the annotation accuracy improved by about 8% in terms of MIoU in combination with considerably faster annotation times, confirming the applicability of our proposal for generating fast and accurate bounding box annotation for the point cloud data.

## CHAPTER 4

---

# Conclusions

---

In this thesis, we have presented several contributions towards the 3D scene understanding. The contributions are provided in the area of semantic segmentation, pointwise classification, and point cloud semi-automatic annotation. We have developed a novel deep learning architecture, called SA-NET. The new architecture incorporates a data fusion strategy capable of combining the point cloud derived features and image-based features to generate high accuracy land cover maps. We have also proposed a robust post-processing module, called A-XCRF, capable of invoking the spatial and spectral similarities between unordered point cloud data. The module provides a noticeable improvement of the deep learning inference output. Finally, we also made the point cloud annotation process more affordable for a wider user-community by providing an open-source and robust semi-automatic annotation tool, called SAnE. This tool is suited to generate fast and accurate point cloud labels.

### 4.1 Technical Evolution of The Thesis

This thesis is the result of research conducted during a three-year PhD study. The research process was designed to fulfill the main research objective: To provide the best way to generate fast and accurate point cloud labels. The first part of the work explored the current state of the art in the field of computer vision and remote sensing. This followed by the development of a new deep learning architecture, called earlyfusion SA-NET (Arief et al., 2018). The next part was used to develop a spatial smoothing algorithm for point cloud data, called XCRF (Arief et al., 2019b). This algorithm emphasizes the spatial and feature similarities among points on the point cloud region during inference process, resulting a better point-level prediction. The final part of the work was on developing a finished product, an open-source software, called SAnE (Arief et al., 2020). This software combines an automatic pointwise classification technique (Arief et al., 2019a) and human-based perception suited to deliver fast and accurate ground truth annotation based on the point cloud data. The main contributions are presented in the core sections of this thesis (Section 3.1, 3.2, and 3.3+3.4) and the technical evolution of our work is explained below.

**Semantic segmentation.** The work documented in Paper A (Section 3.1) started as an exploration of semantic segmentation in the field of remote sensing. The main idea is that given the projected point cloud data, we should be able to

## 4. Conclusions

---

generate high-quality land cover maps. We started by projecting the 3D point cloud data in the 2D plane and extracting the feature information by matching the projected point cloud data with a ground-truth dataset. It should be noted that some part of the point cloud data was used as a training dataset, while the rest was used to assess the accuracy of the classification. Several state of the art techniques based on the deep learning algorithms were deployed and their prediction results were compared. Finally, the SA-NET was proposed and its prediction capability was tested. It should be noted that the SA-NET was developed to overcome the limitations of other state of the art techniques in this domain, i.e sparsely prediction map, high memory consumption, and long training time.

The prediction accuracy of SA-NET was the highest among the deep learning architectures tested in this study. However, the preprocessing time needed to create the projected point cloud data was very long, making it impracticable for use with a larger point cloud scene. Moreover, we realized that testing multiple techniques is time-consuming and the prediction results are data-dependent. Results obtained with a particular technique using a test dataset may give different results from those obtained when the technique was deployed and tested by their original authors.

Therefore, in Paper B (Section 3.2), a deep learning module capable of generating and refining the point cloud segmentation maps *directly* from the point cloud data was developed, and the results were compared with an existing benchmark dataset.

**Point-level classification.** Paper B (Section 3.2) focuses on improving the existing deep learning techniques for raw point cloud data. In this part of the research, a post-processing module based on the conditional random field principle was proposed. It is called the A-XCRF algorithm and is capable of invoking feature similarity between the nearby point in the process of generating final prediction maps. The algorithm is provided as a stand-alone post-processing module (a software library) that can be applied to any machine/deep learning-based techniques.

The A-XCRF prediction results provided the highest accuracy (in terms of the F1-Score) in the benchmark dataset. However, when the full pipeline was tested on the domain adaptation problem, the promising accuracy could not be replicated. Therefore, in Paper C (Section 3.3) and Paper D (3.4), we proposed a semi-automatic annotation tool. This tool involves (minimal) human effort in the making of high accuracy annotations, thus addressing the domain adaptation problem.

**Semi-automatic point cloud annotation.** The final section of our work comprises two parts, Paper C (Section 3.3) and Paper D (Section 3.4). In Paper C (Section 3.3), a density adaptive sampling technique was proposed. This technique enables the use of the deep learning architectures (like the PointCNN) for heterogeneous density point cloud data, see Fig. 3.7. In Paper D (Section 3.4), we proposed the SAnE, a semi-automatic point cloud annotation tool. It was developed to overcome the limitations of other annotation tools, like the Latte and 3D-Bat. Experimental results show that by using the SAnE, an



annotator can create a more accurate point cloud labels while achieving faster annotation times (a speedup by a factor of 4.17).

### 4.2 Limitations of our approach

As we provided several contributions to the field of remote sensing and computer vision (in general), especially in the domain of 3D scene understanding, there are also weaknesses and shortcomings with our proposals. In this section, the details of these limitations are explained.

#### Semantic segmentation

- **Computation cost.** Modern deep learning techniques emphasize the importance of an end-to-end learning process, from the raw input data to the finalized (expected) outcome. This process not only prevents information loss during preprocessing but also alleviates the complexity of product generation. The SA-NET, however, does not embrace this idea for point cloud annotations. The result is an arduous data preprocessing pipeline demanding expensive (computational) resources.
- **Complicated solution.** As we addressed the technical problems on the existing deep learning architectures for semantic segmentation (by stacking and restructuring the up-sampling modules and atrous kernels while offering several other solutions), we failed to realize that our final (data fusion) proposal became more complicated and therefore, is difficult to deploy.

#### Point-level classification

- **Domain adaption.** Addressing the overfitting problem in a limited dataset was the objective behind our proposal in the point-level classification task, resulting in the development of the A-XCRF algorithm. This algorithm uses the unlabeled data points to introduce controlled noise into a validated deep learning model, relaxing the overfitting model of the training data. However, when applied to the unseen dataset with a (very) different landscape, the (more) generalized model failed to generate high accuracy prediction maps.
- **Two step learning.** In order to introduce the unlabeled data to a validated model, the model must first be trained and validated. This process makes our A-XCRF a two-step learning approach, resulting a complicated training pipeline.

#### Semi-automatic point cloud annotation

- **Shape representations.** Relying solely on the point cloud data to generate high-quality annotation labels requires a (perfect) point cloud object representation. Meanwhile, in many cases, objects in the point cloud scene are only represented by few points with many different shapes (U-Shape, I-Shape, dot shape, and others), making it complicated to deliver perfect annotation labels.

## 4. Conclusions

---

- **Frame rate and tracking.** The guided-tracking algorithms included in the SAnE annotation tool is based on a deterministic search algorithm aiming to locate objects within a consecutive data frame. On a high-frequency LiDAR scan, the proposed tracking algorithm can work fine, because the location of each object (in the next frame) is relatively unchanged. However, with a low frame rate, significant changes in object locations (in the next frame) are expected, resulting in an expensive and unreliable search outcome.

### 4.3 Future Directions

In a complete framework perspective, some of the limitations in Section 4.2 have been addressed and presumed better solutions have been proposed in sections 3.2, 3.3, and 3.4, consecutively. The future directions for further research and potential solutions to other limitations, that have so far not been addressed, will be discussed below.

- **Two step learning.** By treating the process of fitting the training data as a classification task and the process of fitting the unlabeled data as a generation task, the GAN training style (Goodfellow et al., 2014) can be adopted to simplify and speed up the A-XCRF training process. Even though the process of finding the state of equilibrium in GANs is relatively unstable, this approach is still more stable than separating the training process into different training pipeline, like in the A-XCRF pipeline. Therefore, it is worth exploring how effective the GAN training style is with respect to improving the stability and easing the complexity of the A-XCRF proposal.
- **Shape representations.** Several learning-based techniques have been proposed for scene completion and point cloud generation (Dai et al., 2018; Groueix et al., 2018). These techniques (individually or combined) can potentially alleviate the incomplete object representation, easing the generation of point cloud labels and the (automatic) detection of point cloud objects.
- **Frame rate and tracking.** The (proposed) deterministic approach relies on a consistent point cloud representation to detect and track objects by location. This precondition is not present in the low-frequency LiDAR scans, and a heuristic or learning-based approach could (potentially) provide a better tracking solution. The learning-based approach, like the T-NET model (Qi et al., 2017b), might be able to perform this task better, by generating consistent and trackable features from an object in different positions. By providing a suitable objective function and fast feature matching algorithm (similar to Scale Invariant Feature Transform (SIFT) algorithm by Brown and Lowe (2002)), then a fast, accurate and reliable object tracking (technique) for a low-frequency LiDAR scan, can be deployed.

## 4.4 Outlook

The 3D scene understanding capabilities, specifically for object detection and semantic segmentation tasks, have been significantly improved during the last couple of years. Meanwhile, 3D laser scanners are becoming more affordable and their use is increasing rapidly. An example of a fast-growing application area of these scanners is in autonomous vehicle. The machine vision combined with the 3D laser scanner will most likely play a major role in providing full autonomy in remote sensing, autonomous vehicles, and even in virtual reality. In remote sensing, the machine vision can provide automatic generation of high resolution semantic (land cover and land use) maps, building extraction, tree identification, crop yield prediction, and more. More strikingly, in robotic and autonomous driving, several (smart autonomy) companies have started deploying the 3D scene understanding capabilities in their (pre-market) autonomous products. Another field of applications is the virtual and augmented reality, where semantic segmentation and object detection techniques are used to transform our real and virtual world for a better and bright future.



---

## Bibliography

---

- Acuna, D., Ling, H., Kar, A., Fidler, S., 2018. Efficient interactive annotation of segmentation datasets with Polygon-RNN++, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 859–868.
- Ahlstrøm, A., Bjørkelo, K., Fadnes, K.D., 2019. Ar5 klassifikasjonssystem. NIBIO Bok .
- Arief, H.A., Arief, M., Bhat, M., Indahl, U., Tveite, H., Zhao, D., 2019a. Density-adaptive sampling for heterogeneous point cloud object segmentation in autonomous vehicle applications, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 26–33.
- Arief, H.A., Arief, M., Zhang, G., Indahl, U., Tveite, H., Zhao, D., 2020. Sane: Smart annotation and evaluation tools for point cloud data, in: Submitted to Flagship Conference for Image Processing and Computer Vision.
- Arief, H.A., Indahl, U.G., Strand, G.H., Tveite, H., 2019b. Addressing overfitting on point cloud classification using Atrous XCRF. ISPRS Journal of Photogrammetry and Remote Sensing 155, 90–101.
- Arief, H.A., Strand, G.H., Tveite, H., Indahl, U., 2018. Land cover segmentation of airborne lidar data using stochastic atrous network. Remote Sensing 10, 973.
- Armeni, I., Sener, O., Zamir, A.R., Jiang, H., Brilakis, I., Fischer, M., Savarese, S., 2016. 3d semantic parsing of large-scale indoor spaces, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1534–1543.
- Badrinarayanan, V., Kendall, A., Cipolla, R., 2017. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE transactions on pattern analysis and machine intelligence 39, 2481–2495.
- Bentley, J.L., 1975. Multidimensional binary search trees used for associative searching. Communications of the ACM 18, 509–517.
- Blom Geomatics AS, 2014. Lidar-Rapport—Follo 2014. URL: [https://hoydedata.no/LaserInnsyn/ProsjektRapport?filePath=\statkart.no\hoydedata\\_orig\vol1\119\metadata\Follo 2014\\_Projekt rapport.pdf](https://hoydedata.no/LaserInnsyn/ProsjektRapport?filePath=\statkart.no\hoydedata_orig\vol1\119\metadata\Follo 2014_Projekt rapport.pdf).

## Bibliography

---

- Brown, M., Lowe, D.G., 2002. Invariant features from interest point groups., in: BMVC.
- Brust, C.A., Sickert, S., Simon, M., Rodner, E., Denzler, J., 2015. Convolutional patch networks with spatial prior for road detection and urban scene understanding. arXiv preprint arXiv:1502.06344 .
- Caltagirone, L., Scheidegger, S., Svensson, L., Wahde, M., 2017. Fast lidar-based road detection using fully convolutional neural networks, in: 2017 IEEE intelligent vehicles symposium (iv), IEEE. pp. 1019–1024.
- Castrejon, L., Kundu, K., Urtasun, R., Fidler, S., 2017. Annotating object instances with a Polygon-RNN, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5230–5238.
- Chang, M.F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., et al., 2019. Argoverse: 3d tracking and forecasting with rich maps, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8748–8757.
- Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L., 2017. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE transactions on pattern analysis and machine intelligence* 40, 834–848.
- Dahl, G.E., Sainath, T.N., Hinton, G.E., 2013. Improving deep neural networks for lvsr using rectified linear units and dropout, in: 2013 IEEE international conference on acoustics, speech and signal processing, IEEE. pp. 8609–8613.
- Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Niebner, M., 2017. Scannet: Richly-annotated 3d reconstructions of indoor scenes. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) doi:10.1109/cvpr.2017.261.
- Dai, A., Ritchie, D., Bokeloh, M., Reed, S., Sturm, J., Niebner, M., 2018. Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4578–4587.
- Demir, I., Koperski, K., Lindenbaum, D., Pang, G., Huang, J., Basu, S., Hughes, F., Tuia, D., Raska, R., 2018. Deepglobe 2018: A challenge to parse the earth through satellite images, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE. pp. 172–17209.
- DeTone, D., Malisiewicz, T., Rabinovich, A., 2018. Superpoint: Self-supervised interest point detection and description, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 224–236.
- Du, S., Zhang, Y., Zou, Z., Xu, S., He, X., Chen, S., 2017. Automatic building extraction from lidar data fusion of point and grid-based features. *ISPRS Journal of Photogrammetry and Remote Sensing* 130, 294–307.

- Everingham, M., Eslami, S.A., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A., 2015. The pascal visual object classes challenge: A retrospective. *International journal of computer vision* 111, 98–136.
- Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., Garcia-Rodriguez, J., 2017. A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857* .
- Geiger, A., Lenz, P., Stiller, C., Urtasun, R., 2013. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research* 32, 1231–1237.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial nets, in: *Advances in neural information processing systems*, pp. 2672–2680.
- Groueix, T., Fisher, M., Kim, V.G., Russell, B.C., Aubry, M., 2018. A papier-mâché approach to learning 3d surface generation , 216–224.
- Hackel, T., 2018. Large-scale Machine Learning for Point Cloud Processing. Ph.D. thesis. ETH Zurich.
- Hackel, T., Savinov, N., Ladicky, L., Wegner, J.D., Schindler, K., Pollefeys, M., 2017. Semantic3d. net: A new large-scale point cloud classification benchmark. *arXiv preprint arXiv:1704.03847* .
- He, K., Gkioxari, G., Dollár, P., Girshick, R., 2017. Mask R-CNN, in: *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969.
- He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.
- Horvat, D., Žalik, B., Mongus, D., 2016. Context-dependent detection of non-linearly distributed points for vegetation classification in airborne lidar. *ISPRS Journal of Photogrammetry and Remote Sensing* 116, 1–14.
- Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q., 2016. Deep networks with stochastic depth, in: *European conference on computer vision*, Springer. pp. 646–661.
- Hubel, D.H., Wiesel, T.N., 1962. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology* 160, 106–154.
- Hurl, B., Czarnecki, K., Waslander, S., 2019. Precise synthetic image and lidar (presil) dataset for autonomous vehicle perception. *arXiv preprint arXiv:1905.00160* .
- Jaboyedoff, M., Oppikofer, T., Abellán, A., Derron, M.H., Loye, A., Metzger, R., Pedrazzini, A., 2012. Use of lidar in landslide investigations: a review. *Natural hazards* 61, 5–28.

## Bibliography

---

- Kemker, R., Salvasgio, C., Kanan, C., 2018. Algorithms for semantic segmentation of multispectral remote sensing imagery using deep learning. *ISPRS journal of photogrammetry and remote sensing* 145, 60–77.
- Krähenbühl, P., Koltun, V., 2011. Efficient inference in fully connected crfs with gaussian edge potentials, in: *Advances in neural information processing systems*, pp. 109–117.
- Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks, in: *Advances in neural information processing systems*, pp. 1097–1105.
- Lahoud, J., Ghanem, B., 2017. 2d-driven 3d object detection in rgb-d images, in: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4622–4630.
- LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al., 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86, 2278–2324.
- Li, B., 2017. 3d fully convolutional network for vehicle detection in point cloud, in: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE. pp. 1513–1518.
- Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B., 2018. PointCNN: Convolution on X-transformed points, in: *Advances in Neural Information Processing Systems*, pp. 820–830.
- Long, J., Shelhamer, E., Darrell, T., 2015. Fully convolutional networks for semantic segmentation, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440.
- McCormac, B.J., 2018. SLAM and deep learning for 3D indoor scene understanding. Ph.D. thesis. Imperial College London.
- Milioto, A., Lottes, P., Stachniss, C., 2018. Real-time semantic segmentation of crop and weed for precision agriculture robots leveraging background knowledge in cnns, in: *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE. pp. 2229–2235.
- Minsky, M.L., Papert, S.A., 1988. *Perceptrons: expanded edition*. MIT Press.
- Mnih, V., 2013. *Machine learning for aerial image labeling*. Citeseer.
- Mughees, A., Tao, L., 2017. Hyper-voxel based deep learning for hyperspectral image classification, in: *2017 IEEE International Conference on Image Processing (ICIP)*, IEEE. pp. 840–844.
- NIBIO, 2018. Nibio ar5 wms service. URL: <https://www.nibio.no/tema/jord/arealressurser/arealressurskart-ar5>.
- Niemeyer, J., Rottensteiner, F., Soergel, U., 2014. Contextual classification of lidar data and building object detection in urban areas. *ISPRS journal of photogrammetry and remote sensing* 87, 152–165.



- Niemeyer, J., Rottensteiner, F., Sörgel, U., Heipke, C., 2016. Hierarchical higher order crf for the classification of airborne lidar point clouds in urban areas. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences-ISPRS Archives* 41 (2016) 41, 655–662.
- Norwegian Map Authority, 2016. Laser Scan Report - Bergen Kommune 2016. Technical Report. Terratec. URL: <https://hoydedata.no/LaserInnsyn/>.
- Payne, A.B., Walsh, K.B., Subedi, P., Jarvis, D., 2013. Estimation of mango crop yield using image analysis–segmentation method. *Computers and electronics in agriculture* 91, 57–64.
- Qi, C.R., Liu, W., Wu, C., Su, H., Guibas, L.J., 2018a. Frustum pointnets for 3d object detection from rgb-d data. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition doi:10.1109/cvpr.2018.00102.
- Qi, C.R., Su, H., Kaichun, M., Guibas, L.J., 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) doi:10.1109/cvpr.2017.16.
- Qi, C.R., Yi, L., Su, H., Guibas, L.J., 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space, in: *Advances in Neural Information Processing Systems*, pp. 5099–5108.
- Qi, R.J., Guibas, L.J., Girod, B.J., Savarese, S.J., 2018b. Deep learning on point clouds for 3D scene understanding. Ph.D. thesis. Stanford.
- Ramiya, A.M., Nidamanuri, R.R., Ramakrishnan, K., 2016. A supervoxel-based spectro-spatial approach for 3d urban point cloud labelling. *International Journal of Remote Sensing* 37, 4172–4200.
- Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation, in: *International Conference on Medical image computing and computer-assisted intervention*, Springer. pp. 234–241.
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al., 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision* 115, 211–252.
- Shi, S., Wang, X., Li, H., 2019. PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–779.
- Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 .
- Sproull, R.F., 1991. Refinements to nearest-neighbor searching ink-dimensional trees. *Algorithmica* 6, 579–589.
- Steinsiek, M., Polewski, P., Yao, W., Krzystek, P., 2017. Semantische analyse von als-und mls-daten in urbanen gebieten mittels conditional random fields. *Tagungsband* 37, 521–531.

## Bibliography

---

- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1–9.
- Van Etten, A., Lindenbaum, D., Bacastow, T.M., 2018. Spacenet: A remote sensing dataset and challenge series. arXiv preprint arXiv:1807.01232 .
- Vladimir, 2018. Deeplab-resnet Rebuilt in Tensorflow. URL: <https://github.com/DrSleep/tensorflow-deeplab-resnet>.
- Wang, B., Wu, V., Wu, B., Keutzer, K., 2019. Latte: Accelerating lidar point cloud annotation via sensor fusion, one-click annotation, and tracking. arXiv preprint arXiv:1904.09085 .
- Waymo, 2019. Waymo open dataset: An autonomous driving dataset.
- Welch, G., Bishop, G., et al., 1995. An introduction to the kalman filter .
- Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J., 2015. 3d shapenets: A deep representation for volumetric shapes, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1912–1920.
- Yang, B., Wang, J., Clark, R., Hu, Q., Wang, S., Markham, A., Trigoni, N., 2019a. Learning object bounding boxes for 3d instance segmentation on point clouds. arXiv preprint arXiv:1906.01140 .
- Yang, Z., Jiang, W., Xu, B., Zhu, Q., Jiang, S., Huang, W., 2017. A convolutional neural network-based 3d semantic labeling method for als point clouds. Remote Sensing 9, 936.
- Yang, Z., Sun, Y., Liu, S., Shen, X., Jia, J., 2019b. Std: Sparse-to-dense 3d object detector for point cloud, in: The IEEE International Conference on Computer Vision (ICCV).
- Yang, Z., Tan, B., Pei, H., Jiang, W., 2018. Segmentation and multi-scale convolutional neural network-based classification of airborne laser scanner data. Sensors 18, 3347.
- Yousefhussein, M., Kelbe, D.J., Ientilucci, E.J., Salvaggio, C., 2017. A fully convolutional network for semantic labeling of 3d point clouds. arXiv preprint arXiv:1710.01408 .
- Yu, L., Wang, Z., Tian, S., Ye, F., Ding, J., Kong, J., 2017. Convolutional neural networks for water body extraction from landsat imagery. International Journal of Computational Intelligence and Applications 16, 1750001.
- Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.H., 2015. Conditional random fields as recurrent neural networks, in: Proceedings of the IEEE international conference on computer vision, pp. 1529–1537.

- Zhou, Y., Tuzel, O., 2018. Voxelnet: End-to-end learning for point cloud based 3d object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4490–4499.
- Zimmer, W., Rangesh, A., Trivedi, M., 2019. 3d bat: A semi-automatic, web-based 3d annotation toolbox for full-surround, multi-modal data streams. arXiv preprint arXiv:1905.00525 .



---

## **Appendices**

---



---

**Paper A:**  
**Land Cover Segmentation of Airborne LiDAR  
Data Using Stochastic Atrous Network**

---

**Land Cover Segmentation of Airborne LiDAR Data Using  
Stochastic Atrous Network**

Hasan Asy'ari Arief <sup>1,\*</sup>, Geir-Harald Strand<sup>1,2</sup>, Håvard Tveite<sup>1</sup>, and Ulf  
Geir Indahl<sup>1</sup>,

<sup>1</sup>*Faculty of Science and Technology, Norwegian University of Life  
Sciences, 1432 Ås, Norway*

<sup>2</sup>*Division of Survey and Statistics, Norwegian Institute of Bioeconomy  
Research, 1431 Ås, Norway*


Published on Remote Sensing (ISSN 2072-4292).





Article

# Land Cover Segmentation of Airborne LiDAR Data Using Stochastic Atrous Network

Hasan Asy'ari Arief <sup>1,\*</sup> , Geir-Harald Strand <sup>1,2</sup> , Håvard Tveite <sup>1</sup>  and Ulf Geir Indahl <sup>1</sup><sup>1</sup> Faculty of Science and Technology, Norwegian University of Life Sciences, 1432 Ås, Norway;

geir.harald.strand@nibio.no (G.H.S.); havard.tveite@nmbu.no (H.T.); ulf.indahl@nmbu.no (U.G.I.)

<sup>2</sup> Division of Survey and Statistics, Norwegian Institute of Bioeconomy Research, 1431 Ås, Norway

\* Correspondence: hasanari@nmbu.no; Tel.: +47-453-91-706

Received: 30 April 2018; Accepted: 17 June 2018; Published: 19 June 2018



**Abstract:** Inspired by the success of deep learning techniques in dense-label prediction and the increasing availability of high precision airborne light detection and ranging (LiDAR) data, we present a research process that compares a collection of well-proven semantic segmentation architectures based on the deep learning approach. Our investigation concludes with the proposition of some novel deep learning architectures for generating detailed land resource maps by employing a semantic segmentation approach. The contribution of our work is threefold. (1) First, we implement the multiclass version of the intersection-over-union (IoU) loss function that contributes to handling highly imbalanced datasets and preventing overfitting. (2) Thereafter, we propose a novel deep learning architecture integrating the deep atrous network architecture with the stochastic depth approach for speeding up the learning process, and impose a regularization effect. (3) Finally, we introduce an early fusion deep layer that combines image-based and LiDAR-derived features. In a benchmark study carried out using the Follo 2014 LiDAR data and the NIBIO AR5 land resources dataset, we compare our proposals to other deep learning architectures. A quantitative comparison shows that our best proposal provides more than 5% relative improvement in terms of mean intersection-over-union over the atrous network, providing a basis for a more frequent and improved use of LiDAR data for automatic land cover segmentation.

**Keywords:** land cover segmentation; stochastic depth atrous network; IoU loss function; airborne LiDAR data; deep learning data fusion

---

## 1. Introduction

The advancement of airborne LiDAR (light detection and ranging) technologies over the past decades have contributed significantly to the increased availability of high precision point cloud data. Thanks to the government project “Nasjonal detaljert høydemodell” [1], LiDAR data are already available for many parts of Norway through the “høydedata” website [2]. The aim of the project is to provide national coverage of high-resolution elevation data by 2019.

Easy access to high-precision LiDAR data in Norway opens an opportunity to identify and extract valuable information for many different purposes. Airborne LiDAR data have become the basis for generating a high accuracy digital terrain model, hazard assessment and susceptibility mapping, and the monitoring of surface displacements [3]. In terms of land cover classification, LiDAR data have been used to automatically generate maps for forest fire management [4] and land/water classification [5]. However, due to the complexity of the data and the difficulty of generating features of interest, most of the current techniques suffer from scaling problems when transferred to larger datasets [6].

We have investigated deep learning techniques to create a classifier for the AR5 land resource dataset [7] from Norwegian Institute of Bioeconomy Research (NIBIO). AR5 is the national

high-resolution, combined land cover and land capability dataset for Norway. Challenges with this dataset include: (a) AR5, as “ground truth” data, is an imperfect match with our LiDAR data due to the difference in resolution/level of detail; (b) the different generalization approaches and data sources that have been used to generate the AR5 classes; (c) differences in the acquisition times of AR5 and the LiDAR data; (d) point cloud data are represented as points in a specified location, which requires a fine-grain preprocessing technique to make them usable in a semantic segmentation pipeline.

One of the most common techniques for doing land cover segmentation is by engineering handcrafted features from the LiDAR data to train a machine learning algorithm for automatic detection. Guo et al. investigates 26 manually derived features from LiDAR data for the purpose of automatic classification [8], while MacFaden et al. utilizes normalized difference vegetation index (NDVI), normalized digital surface model (nDSM), and high-resolution multispectral images to do high-resolution tree canopy mapping for the same purpose [9]. Yan et al. [10] reviews a number of research projects that utilize LiDAR data for automated extraction, using three LiDAR-derived features for generating urban land cover maps [11] to several full waveform-derived features for the classification of dense urban scenes [12].

The manual engineering of features as inputs to a machine learning classifier has become common practice for generating land segmentation maps [8–12]. However, such approaches limit the ability to provide a richer representation of the data [13]. The chosen features may not be sufficient to characterize the uniqueness of a certain class or object [14], and the quality of the resulting classifier depends heavily on the feature engineering output. As each particular segmentation problem is more or less unique [15], different feature engineering approaches are often needed for different problems.

Deep learning methodologies contribute to simplifying the feature engineering process, and by using a deep multi-layer convolutional neural network (CNN) [16], the features are learned from the data during the training process. Consequently, a successfully trained CNN model can be considered as a feature extractor that both combines features in different spatial locations and takes into account the aspects of spatial autocorrelation between features.

From the era of AlexNet [17] in 2012 to the more complex deep learning architectures such as GoogleNet [18] and ResNet [19] in 2014 and 2015, respectively, deep learning has become one of the most successful machine learning techniques for approaching problems related to computer vision, including image classification, object detection, and more advanced challenges such as semantic segmentation and instance-aware segmentation. In the remote sensing community, deep learning has also been used successfully for land cover classification [20], road detection [21], and scene understanding [22].

The purpose of the present paper is to investigate and compare a collection of well-proven semantic segmentation architectures based on the deep learning approach. Based on the experiences obtained from this investigation, we propose a novel deep learning architecture for the particular land cover segmentation problem of interest. The proposed architecture involves a CNN fusion layer for merging the image-based features with height above ground information and intensity values. The output of the fusion layer provides the basis for training an atrous deep network [23] using the stochastic depth approach [24] combined with a rescaling of the derived feature map using skip connections and the learnable upsampling approach, which is similar to the technique in fully convolutional networks [25]. The final layer of our architecture includes a softmax activation function combined with an intersection-over-union (IoU) loss function [26] that both speed up the convergence process and improve the accuracy of the resulting classifier. We are able to demonstrate that the classifier resulting from our proposed architecture gains more than 5% accuracy improvement (validated) over the atrous network (DeeplabV2) [23].

In the next section, we review some of the most popular deep learning techniques for semantic segmentation. In the subsequent sections, we describe the characteristics and properties of our datasets, and the challenges that come with them. Thereafter, we report the results of our research process, where we explore some of the most well-proven deep learning architectures for semantic segmentation.

Based on our findings, we propose a novel deep learning architecture. Finally, we summarize and discuss the classification results obtained by the various architectures. Based on the conclusions drawn from our findings, we indicate future work, including possible improvements of our novel deep learning architecture and benchmarking with more traditional land cover segmentation methods. The trained model and reproducible code are available at <https://github.com/hasanari/SA-net>.

## 2. Semantic Segmentation

Semantic segmentation is essentially a dense pixel classification task, where each individual pixel on a raster grid is classified according to a predefined classification schema. Here, we will review some of the proposed and well-proven techniques in the area of computer vision for semantic segmentation since the start of the era of deep learning.

One of the most prominent architectures in this area is fully convolutional networks (FCN) [25]. This architecture gave a 20% relative improvement in classification performance on the PASCAL VOC 2012 dataset [27]. FCN has been used for many semantic segmentation tasks, ranging from medical image analysis [28] to multispectral remote sensing [29]. Motivated by the success of the FCN architecture, several alternative architectures and modifications have been proposed and demonstrated to be appropriate for these types of applications. In particular, we mention the segmentation networks (SegNet) [30], the deconvolutional networks (DeconvNet) [31], the pyramid scene parsing network (PSPNet) [32], and the atrous networks [23].

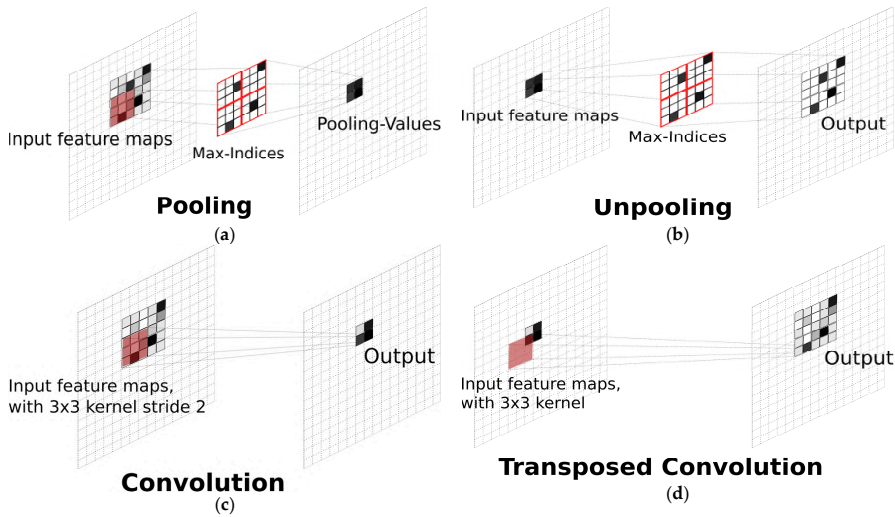
### 2.1. Characteristics of the Different Architectures

The FCN architecture offers a learning pipeline with the possibility of embedding pre-trained classification networks such as AlexNet [17] and VGGnet [33] for solving the pixel segmentation problem. More specifically, the existing deep learning architectures used to address the image classification problem can be reused with their pre-trained model, and their last fully connected layers can be retrained to obtain the complete dense pixel classifier. In addition, the FCN also includes the possibility of using the so-called skip connection method in order to deal more effectively with the coarseness of the predicted pixels.

Unlike the FCN, SegNet includes the unpooling operation [30], i.e., an upsampling technique to obtain an appropriate segmentation map in a resolution identical to the corresponding original image. Instead of directly generating pixel-wise predictions from the smallest feature maps, SegNet applies the unpooling method as a step encoder in order to regenerate larger feature maps from the resulting max indices obtained from a max pooling operation in the corresponding unpooling layer, see Figure 1a,b. Following this approach, the feature indices that have the maximum value will be preserved as dominant in the upsampling procedure, often resulting in improved classification performance. In addition to the unpooling technique, deconvolutional networks or deconvnet [31] propose a “transposed” convolution operation referred to as “deconvolution”, see Figure 1d.

Another architecture of interest to our problem is the pyramid scene parsing network (PSPNet) [32]. These architectures include a pyramid pooling module that instead of using a fixed kernel size, uses kernels of different sizes to generate pyramidal convolutional layers generating multi-size feature maps. The purpose of this approach is to help improve the classification performance by more effectively incorporating a broader spatial context for the different feature map resolutions.

For our field of application, we finally consider atrous convolution architecture [23] to be of particular interest. An important advantage of this architecture is the moderate amount of pooling layers, which prevents the extensive use of rescaling procedures. Using this approach, we can avoid extensive upsampling [34].



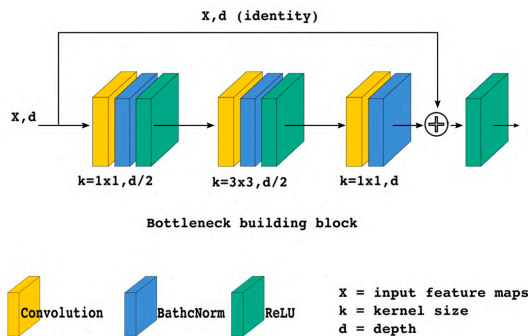
**Figure 1.** Illustration of (a) pooling, (b) unpooling, (c) convolution, and (d) transposed convolution operation. Each operation uses a  $3 \times 3$  kernel with a stride/step of two, and for any overlapping cell, a summation of the overlapping values of the cells are performed.

2.2. The Atrous Convolution Architecture

In the present study, we implemented an atrous convolution architecture that resembles the deep residual network (ResNet) [19]. The main idea of the ResNet is to pass the input feature maps of a learning block, as a form of identity reference, to the final output of its block. It is hypothesized that it is easier to optimize feature maps with residual references than unreferenced ones. According to the original ideas of the ResNet, the residual learning of a particular layer is defined as:

$$y = F(x, (W_i)) + x, \tag{1}$$

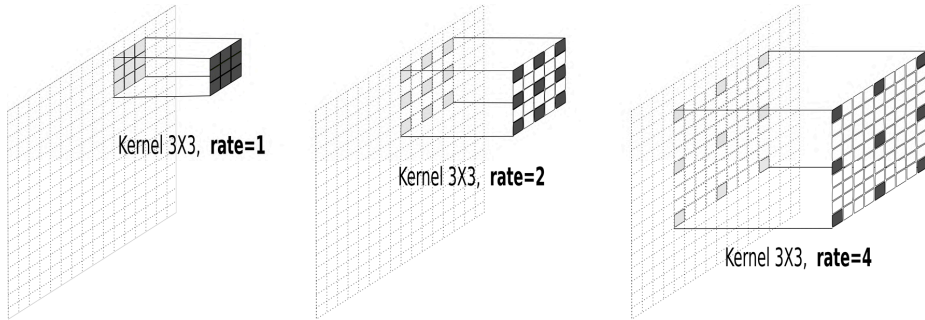
where  $x$  and  $y$  denote the representation of the input and output features of the layer, respectively.  $x$  also denotes an identity mapping from the previous block, while the function  $F(x, (W_i))$  denotes the process of convolving  $W_i$  through  $x$ , including the use of rectified linear unit (ReLU) activation functions [35]. We used a residual learning called “bottleneck” building block in our research, as shown in Figure 2.



**Figure 2.** Bottleneck building block in deep residual network (ResNet) architecture.

The particular variant of ResNet that we utilize here is called ResNet 101. This architecture consists of 101 residual layers with four main blocks. In the remaining parts of this paper, we will refer to each such block as a MainBlock. Each MainBlock has three, 4, 23, and three inner blocks, respectively. MainBlock 1 and 2 use 64 and 512 depth kernels, while MainBlock 3 and 4 use 1024 and 2048 depth kernels, respectively. In the atrous network, MainBlock 1 and 2 consist of several bottleneck building block using convolution kernels, while MainBlock 3 and 4 use the building blocks with atrous kernels.

Unlike normal convolutional layers, atrous convolutions convolve through the feature spaces using a sparse kernel including gaps that are referred to as the *rate*, as shown in Figure 3. An atrous kernel with rate one is identical to an ordinary convolution kernel. The idea of kernels with gaps is to obtain convolutions of various aggregated regions without changing the spatial resolution of the output layer. This property enables an atrous layer to combine features that are close, but are not directly spatially connected, while maintaining the spatial resolution and avoiding the use of larger kernels. This approach allows the analysis to benefit from the presence of spatial autocorrelation in the data.



**Figure 3.** Atrous kernel with rates one, two, and four. The atrous kernel with rate one is identical to a normal convolution kernel.

### 2.3. Evaluation Metrics

Various evaluation metrics can be used to determine the effectiveness of a machine learning model for predicting the unseen data. One of the most common metrics used for pixel classification problems is the *pixel accuracy* (PA). The PA has been frequently used in many classification tasks [25,27,36]. However, in the segmentation problem, the accuracies are commonly measured using mean pixel accuracies (MPA), mean intersection-over-union (MIoU), and the F-Measure (F1 Score). It should be noted that for segmentation problems where the total area of the classes is very different (imbalanced), the PA measure is less informative. This is because assigning all of the pixels to the largest class may result in a large PA value, even without training a model.

With  $k + 1$  being the total number of classes (including the background class) and  $p_{ij}$  denoting the number of pixels from class  $i$  assigned to class  $j$ , the accuracy measures PA, MPA, and MIoU are defined [37] as follows:

$$PA = \frac{\sum_{i=0}^k p_{ii}}{\sum_{i=0}^k \sum_{j=0}^k p_{ij}}, \quad (2)$$

$$MPA = \frac{1}{k+1} \sum_{i=0}^k \frac{p_{ii}}{\sum_{j=0}^k p_{ij}}, \text{ and} \quad (3)$$

$$MIoU = \frac{1}{k+1} \sum_{i=0}^k \frac{p_{ii}}{\sum_{j=0}^k p_{ij} + \sum_{j=0}^k p_{ji} - p_{ii}}. \quad (4)$$

The definition of MIoU (Equation (4)) shows that this metric considers both the valid prediction ( $p_{ii}$ ) and a penalization with respect to the false negatives ( $p_{ji}$ ) and false positives ( $p_{ij}$ ). Training an architecture based on the MIoU contributes to making the resulting model both robust in terms of good predictions and more sensitive against the bad ones.

In addition to the MIoU, we also use the F-Measure to better evaluate the boundary region of the predicted pixels [30]. We use the mean of the F-Measure per class to evaluate the performance of the classifiers. This metric considers both the precision ( $p$ ) and recall ( $r$ ) of the prediction results. With TP denoting the true positives, FP denoting the false positives, and FN denoting the false negatives, the F-Measure (F1 Score) is defined as:

$$p = \frac{TP}{TP + FP'} \quad (5)$$

$$r = \frac{TP}{TP + FN'} \quad \text{and} \quad (6)$$

$$\text{F1 Score} = \frac{1}{k+1} \sum_{i=0}^k 2 * \frac{p * r}{p+r}. \quad (7)$$

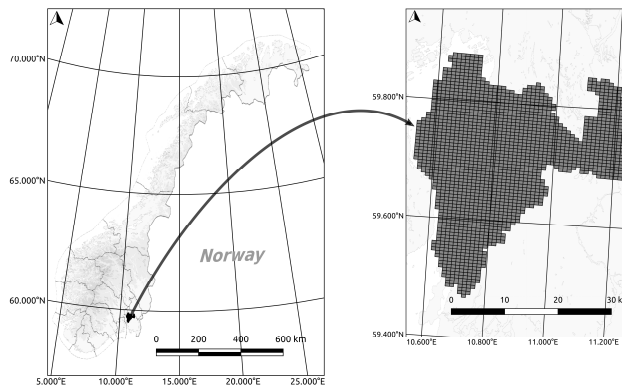
For the sake of simplicity, we will refer to the pixel accuracy measure as PA, the mean pixel accuracy as MPA, the mean intersection-over-union as MIoU, and the F-Measure as F1 throughout this paper.

### 3. Data Sources

In this section, we describe the datasets and preprocessing techniques used in our experiments prior to the deep learning training process.

#### 3.1. Study Area and Datasets

Our study area covers the Follo region, which is a part of the county of Akershus, Norway, as shown in Figure 4. Follo is located in Southern Norway, covers 819 km<sup>2</sup>, and has a relatively long coastline. It consists of moderately hilly terrain dominated by forest, with large patches of agricultural areas and smaller areas of settlement. There are also some lakes in the area.



**Figure 4.** The study area (Follo) with 1877 tiles covering an area of 819 km<sup>2</sup>.

The datasets that we have used for our research are gridded (1 m × 1 m) LiDAR point cloud data as input data, and the AR5 land resource map from NIBIO as label/ground truth data. The LiDAR data used for this research were taken from the Follo 2014 project with project number BNO14004. The project was operated by Blom Geomatics AS with a specification of 5 points/m<sup>2</sup> covering 850 km<sup>2</sup> using a Riegl LMS Q-780. The scan was done from 1030 m aboveground with a pulse repetition

frequency of 266,000 Hz and a scan frequency of 93 Hz [38]. The LiDAR data used for this project have both LiDAR-derived features (height, intensity, number of return, and more) and image-based (red–green–blue, or RGB) features. The total number of points is approximately eight billion, which is divided into 1877 tiles that are stored in LAZ (LAS compressed files) format. Most of the tiles have a size of 600 m × 800 m (600 × 800 pixels), and each tile consists of more than 2.4 million points.

The ground truth data that we have used for this project are the NIBIO AR5 land resource dataset [7]. The dataset covers Follo and consists of four types of classification, namely land type category (arealtype, a mixture of land use and land cover classification), forest productivity (skogbonitet), tree type (treslag), and ground conditions (grunnforhold). In this project, we utilize only the land type classification, which originally covers 11 classes. Due to the non-existence or very small coverage of some of the land type classes in the Follo area, we have simplified the classification task to the modeling and prediction of the following eight classes: settlement, road/transportation, cultivation/grass, forest, swamp, lake–river, ocean, and *other land types*. The *other land type* class is a combination of undefined objects, open interpretations, and other classes with limited representation in the area (below 0.5%).

The AR5 dataset was first established in the 1960s, and has been updated continuously since 1988 for the purpose of agriculture and forest management in Norway. The precision for the agricultural classes is recorded down to 500 m<sup>2</sup>. Other areas such as forest, swamp, and open areas smaller than 2000 m<sup>2</sup> are not registered, but include surrounding or neighboring classes. In addition, classes such as road and settlement have their origin in auxiliary datasets that have been merged into the AR5 dataset, thus resulting in a varying level of details (LoD) and accuracy for the included classes. These conditions represent considerable challenges when using the AR5 dataset as ground truth for classifying areal types based on the high-precision LiDAR dataset, since perfect classification results based on the image data are impossible to achieve.

### 3.2. Challenges with the AR5 Dataset

One of the most pronounced challenges with respect to learning the AR5 classes using LiDAR data is the difference in resolution. The laser data has a submeter LoD, while the AR5 dataset has a maximum resolution of 500 m<sup>2</sup>. The difference in the original resolutions clearly contributes to making the regridded ground truth class labeling data inaccurate at the 1 m<sup>2</sup> resolution of the gridded LiDAR dataset. In addition, differences pertaining to acquisition times (last time of update) also contribute to inaccuracies in the class labeling.

There is also a notable difference between the AR5 classes with respect to their internal homogeneity. E.g., a “forest” can be composed of different tree species producing heterogeneous structures in terms of their LiDAR footprints. Parts of the forest may consist of old stands, while other stands are recent clear-cuts. In between these extremes, every age group is represented within the “forest” class.

Therefore, the accuracy with respect to the prediction of the ground truth data is correspondingly limited by these shortcomings. For example, some settlement classes are actually combinations of roads, trees, and buildings. This makes it more challenging to train a classifier to successfully distinguish between these classes.

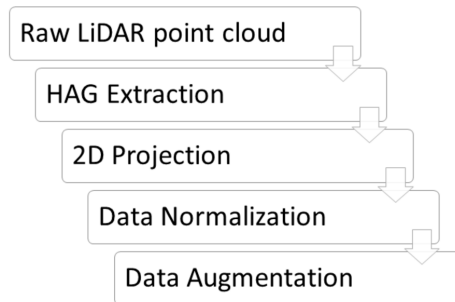
Another practical challenge is the imbalance in the relative frequencies of the different class labels. More than 50% of the Follo area has class labels marked as Forest, and about 20% of the area has class labels marked as cultivation. The remaining areas are unevenly distributed over the other six classes. This imbalance calls for the training process to focus more intensively toward modeling the underrepresented classes.

### 3.3. Preprocessing Procedures

The pipeline of data preprocessing for the current investigation can be described as follows: (1) extracting of height above ground (HAG) for the LiDAR point cloud, (2) aggregating the HAG values



into a two-dimensional (2D) grid, and (3) normalizing the feature values in the new representation. The summary of the preprocessing pipeline can be shown in Figure 5.



**Figure 5.** The preprocessing pipeline that was used for our research. It should be noted that the data augmentation was only applied to the training dataset.

Firstly, the HAG values were extracted using a HAG filter in the point data abstraction library (PDAL) toolkit [39]. The purpose of this process was to obtain a height representation that reflects the variation in the vegetation cover and the built environment.

Secondly, in the gridding step, the LiDAR data were aggregated to a  $1\text{ m} \times 1\text{ m}$  resolution grid based on their spatial location. The AR5 dataset was therefore regridded into the same  $1\text{ m} \times 1\text{ m}$  resolution as the LiDAR data, and each cell was given a (RGB) color roughly according to the style used for the NIBIO AR5 web map service (WMS) [40]. This process was the most time-consuming step, because billions of LiDAR points required individual processing before the grid cell aggregation. The processing was completed by an extensive multi-threaded application of the LibLAS library for python.

The grid size of 1 m was chosen to be consistent with the unit size of the coordinate system that covers the laser data. Consequently, an individual point was assigned to a grid cell by truncation of its coordinate values. Aggregated values for each grid cell were from the assigned points. The most frequent value in each of the RGB channels were taken as the RGB channel values of the grid cell, while for HAG and intensity, mean values were used. Using the mean value for a high-resolution grid with (on average) five points yields small corresponding standard deviations, in particular within areas that are roughly homogenous. The complete aggregation process results in a dataset containing gridded LiDAR with RGB, HAG, and intensity as features. It should be noted that we have padded the empty grid cells with zeros, similar to zero padding in CNN [19]. This requires less computation than using an interpolation algorithm, and arguably, for a deep learning architecture, it has a similar effect on a high-density point cloud ( $5\text{ points/m}^2$ ) [17].

Lastly, after completing the gridding process, we performed zero mean and unit variance normalization. The values of each channel were obtained by subtracting the channel mean and dividing by the channel standard deviation. Finally, we split the normalized tiles into sets of training (70%), validation (10%), and test data (20%), respectively.

For the training data, we performed a patching process with a sliding window for each tile. We used a patch size of  $224 \times 224$  pixels and a 168-pixel ( $3/4$ ) step size. For all of the boundary patches that were smaller than the patch size, we shifted the origin of the boundary patches back and/or up so that the patches all have the size of  $224 \times 224$  pixels. The reason for choosing a patch size of  $224 \times 224$  is that this is the most commonly applied crop size for both the VGGnet and the ResNet architecture [19]. This choice simplifies the benchmarking process, because the other deep learning architectures that we have explored in our study were designed for the  $224 \times 224$  patch size.



The resulting preprocessed dataset is a set of overlapping patches, where the pairwise overlap of horizontally and vertically neighboring patches is 1/4 (25%). The overlap contributes to reducing the impact of disturbing boundary effects in the resulting classifiers.

The successful training of a deep learning architecture requires access to a large number of training samples. Therefore, the final part of the preprocessing procedure is the data augmentation step. With image data, the data augmentation process is completed by generating additional data through flipping and rotating the original images. For each of the patches described above, we did a 90° clockwise rotation and up-down and left-right flipping. This process increases the dataset by 300%, and at the end of the preprocessing pipeline, we obtained a total of 57,036 patches that were available for the training process.

It should be noted that for the validation and test data, we omitted the sliding window and data augmentation parts. Most of the architectures can process images of any size; therefore, the validation and test accuracies were calculated by using the original LiDAR gridding (600 × 800 pixels) to simplify the inference process. We used the validation data for selecting the “best model” for each architecture, and the test data to calculate the final accuracies of the selected models.

The implementations for all of our experiments were based on the TensorFlow library [41] for deep learning developed by Google. The deep learning architectures were trained for at least 50 epochs with an ongoing updating of the classification accuracies on a desktop computer using 11GB GeForce GTX 1080 Ti graphics cards. The full setup for our training process of the various architectures took several weeks. However, the final and best model was trained to the reported performance in only two days.

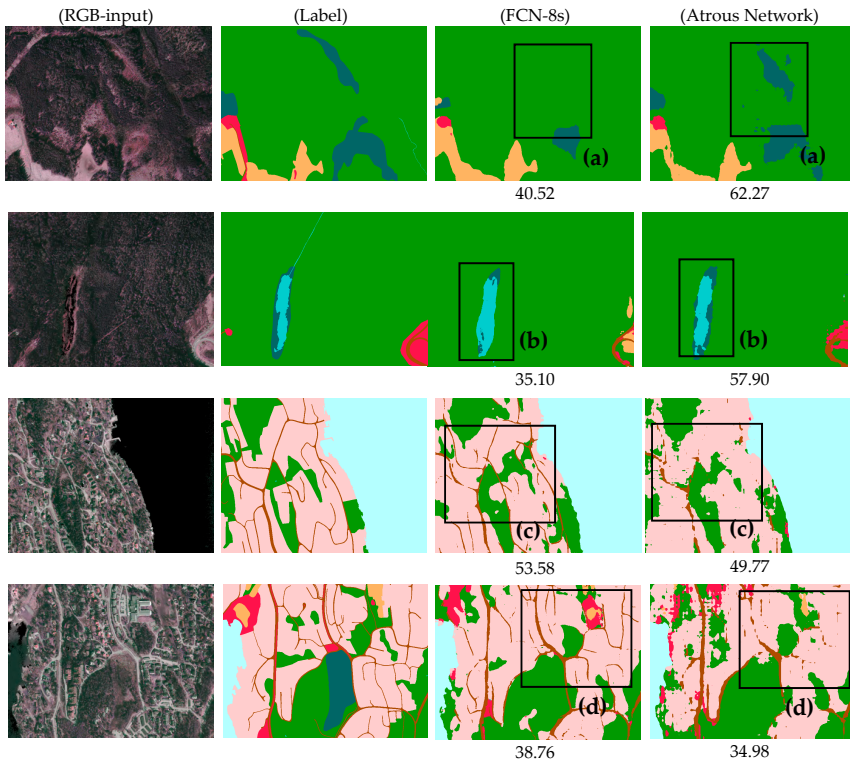
#### 4. The Research Process (Modeling Workflow)

The workflow of our research process was as follows. First, we carried out a number of experiments by training several well-known deep learning architectures for semantic segmentation with our data. The purpose of these experiments was to identify the baseline level of classification performance, and generate the required experiences to propose some of the potentially useful alternative architectures for later benchmarking. It should be noted that the initial experiments only included image-based (RGB) features, since architectures designed for using LiDAR data are hardly available.

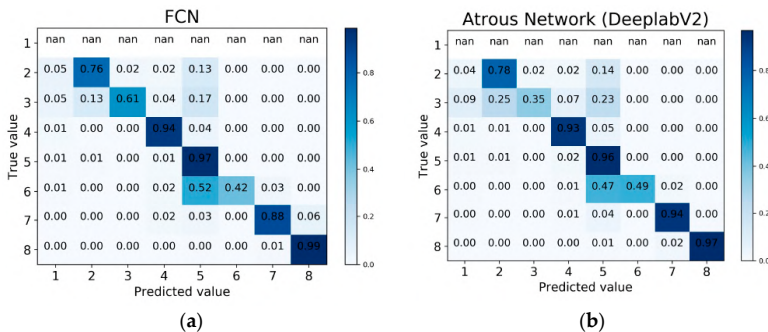
Based on the experimental results, we did an elementary qualitative analysis by inspecting the predicted output masks (Figure 6) and associated confusion matrices (Figure 7). Among our notable findings were that the classifier obtained by training an atrous network generalized considerably better than the classifier obtained by the FCN-8s architecture. In Figure 6a, the FCN-8s erroneously predicted most of the area to be green. In Figure 6b, we see that the FCN-8s generalizes too heavily for the highlighted area. We argue that this is most likely due to the effect of the small receptive field in the FCN kernel. The atrous network, on the other hand, was quite successful in predicting the most common classes, but it failed more frequently in predicting the less common ones.

We also noted that the atrous network upsampling, which utilizes a nearest neighbor technique, failed to generate a valid prediction mask. Figure 6c,d shows the insufficiency of this upsampling technique in recovering the thin and linear class of roads. We suspect that this insufficiency is closely related to the upscaling technique being part of the atrous network’s architecture.

With the ambition of preserving the advantages of the above architectures, and compensating for their limitations, we were lead to propose an alternative architecture for our classification problem. The proposed alternative architecture included atrous convolution kernels for improved generalization and the FCN-based upsampling approach to strive for predictions that are less coarse. In addition, we incorporated the softmax with IoU loss function, which is known to be effective in other architectures. We also included LiDAR-derived features such as HAG and intensity together with an effective merging technique to include the additional features, aiming at improving the classification performance of our alternative deep learning proposals.



**Figure 6.** Qualitative segmentation results from FCN-8s and atrous network. The mean pixel accuracy (MPA) value is included at the bottom of each prediction map. Classes and colors: *other land types* (red), settlement (pink), road/transportation (chocolate), cultivation/grass (orange), forest (green), swamp (dark blue), lake-river (eggshell blue), and ocean (light cyan). (a) and (b) show the ability of the atrous network to predict the most common classes better than the FCN-8s. (c) and (d) show the shortcoming of the upsampling technique in the atrous network compared to the one in the FCN-8s.



**Figure 7.** Comparison of the confusion matrix between (a) FCN-8s and (b) atrous network. The class names are (1) *other land types*, (2) settlement, (3) road/transportation, (4) cultivation/grass, (5) forest, (6) swamp, (7) lake-river, and (8) ocean. The colored bar represents the value of each cell: the higher the value, the darker the cell color.

#### 4.1. Baseline Experiments

An initial investigation based on modeling exclusively with RGB features is required to establish the benchmark classification performances associated with various existing deep learning architectures for our problem. We started out this investigation by exploring four different architectures, i.e., the DeconvNet, the SegNet, the FCN-8s, and the atrous network from DeepLab.

The first architecture that we explored was the DeconvNet, where we used Fabian's implementation [42] with five full convolutional blocks followed by five deconvolutional blocks. The architecture was trained using a mini batch gradient descent strategy by including 15 images per iteration. The training process was based on the Adam optimizer [43] with an initial learning rate of  $1 \times e^{-4}$ . By training from scratch (no utilization of a pre-trained model for this architecture), two weeks of training was insufficient for obtaining a satisfactory classification model. We therefore decided to abort the training process at that stage, and not pursue the training of this architecture any further.

The second architecture that we explored was the SegNet, where we used the Aerial Images implementation from Ørstavik [44] (called AirNet). We implemented the AirNet using five encoders and five decoders in a SegNet architecture with training based on the AdaGrad [45], including a dropout technique acting as regularization [46] to prevent against overfitting. The dropout works by ignoring a certain percentage of its neurons randomly in each epoch of the training process.

The initial learning rate for AirNet was set to  $1 \times e^{-4}$ , and the architecture was trained from scratch for 90 epochs (with an additional 40 epochs due to not utilizing a pre-trained model). The resulting model obtained an accuracy of 59.12% in terms of MIoU and 92.11% in terms of PA. It should be noted that the test accuracies for AirNet were calculated using test data with a sliding window of  $224 \times 224$  and no augmentation. This is due to the limitation of the unpooling module of this architecture to address dimensions of  $600 \times 800$ , which were the dimensions on the original test data.

The third architecture that we explored was the FCN-8s architecture. We used the FCN-8s implementation from Shekkizhar [47]. Shekkizhar's FCN-8s was built based on VGGnet architecture. We trained the model by using the Adam optimizer and fine-tuned it by using VGGnet weights obtained from MatConvNet [48]. Similar to the SegNet, the FCN-8s was also trained with regularization by using the dropout technique. The initial learning rate was set to  $1 \times e^{-4}$ , and the architecture was trained for 50 epochs. Interestingly, by using the straightforward upsampling technique of FCN-8s, a MIoU equal to 64.97% was obtained for the test data.

The last architecture that we explored was the deeplabV2 [23], which utilized an atrous convolution layer on top of a ResNet architecture; we called it the atrous network. For training this architecture, we utilized the DeepLab-ResNet architecture rebuilt in TensorFlow, as seen in Vladimir's implementation [49]. The atrous network implementation was trained with an initial learning rate of  $2.5 \times e^{-4}$  and a weight decay of 0.0005. The training process was optimized using a momentum update [50] with a momentum value of 0.9 and batch normalization [51] to reduce the internal covariate shift due to the parameters update during back propagation. The nearest neighbor technique was used to upsample the final feature maps back into the original input size, before calculating the accuracies.

Two approaches were used to calculate the test set prediction accuracies for the atrous network architecture. The first one included the post-processing conditional random field (CRF) [52], and the second one did not. It should be noted that by skipping the CRF post-processing, we obtained the better classifier, as seen in Table 1.

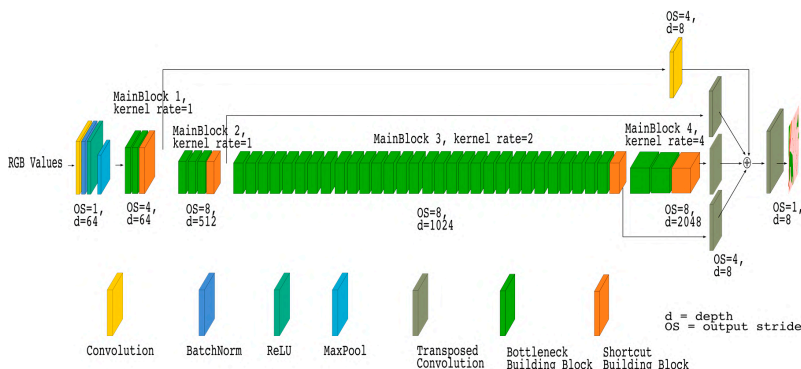
**Table 1.** The test result using image-only features. CRF: conditional random field; FCN: fully convolutional networks; MIoU: mean intersection-over-union; MPA: mean pixel accuracies; PA: pixel accuracy; SegNet: segmentation networks.

	PA	MPA	MIoU	F1
FCN-8s	93.36	69.62	64.97	73.05
SegNet	92.11	63.79	59.12	67.13
Atrous Network (DeeplabV2)	92.28	67.60	62.81	70.79
Atrous Network + CRF	90.97	61.12	56.70	63.50

#### 4.2. The FCN-Based Architectures Including IoU Loss

The results obtained by using the atrous network revealed to us some weaknesses due to the application of a non-learnable upsampling technique, as seen in Figure 6. We therefore decided to integrate the FCN upsampling technique with the atrous network’s architecture in order to try to overcome these problems.

In order to integrate the ResNet-FCN [53] and the atrous kernel, we modified the third and fourth MainBlock of the ResNet-FCN with atrous kernels with rates of two and four, respectively. The output of the last three MainBlocks were upsampled using transposed convolution to get the same size of the output as for the first MainBlock, and all of the MainBlock outputs were combined by using the addition operator. Note that before combining, the first MainBlock was updated with an additional convolution layer using a  $1 \times 1$  kernel with a depth of eight and a stride of one. Finally, the original image size was recovered using a transposed convolution, as shown in Figure 8.



**Figure 8.** Illustration of the atrous-FCN architecture.

The main idea summarized in the resulting architecture reflects the desire to obtain a model that is capable of looking widely for the appropriate feature maps by using the atrous kernels. Simultaneously, the ability of learning the upsampled feature maps is maintained by using transposed convolutions. We will refer to this architecture as the *Atrous-FCN*.

We explored the possibilities of training with both the *softmax with cross-entropy* loss function [54], and the *softmax with IoU* loss function [26]. The results were compared in a quantitative analysis. The reason for exploring the IoU type of loss function is to try to make the network more robust against bad predictions, as the IoU metric penalizes against both false positives and false negatives. With *TP* denoting the true positives, *FP* denoting the false positives, and *FN* denoting the false negatives, the original IoU is defined as:

$$IoU = \frac{TP}{FP + TP + FN}. \tag{8}$$

By using a loss function closely related to the MIoU metric, we experienced that the training was effective not only for speeding up the training process, but also for improving the classification performance.

The IoU (Equation (8)) is by definition a non-differential function that cannot be used directly with the back-propagation algorithm. Fortunately, Rahman et al. has proposed an approximation of the IoU by a differentiable function that replaces the counting operations with multiplications and additions [26]. For a binary classification problem, they proposed a formulation of the IoU by considering the ratio between intersection  $I(X)$  and union  $U(X)$ , where  $I(X)$ ,  $U(X)$ , and the IoU loss ( $L_{IoU}$ ) were defined as:

$$I(X) = \sum_{v \in V} X_v * Y_v, \quad (9)$$

$$U(X) = \sum_{v \in V} X_v + Y_v - X_v * Y_v, \text{ and} \quad (10)$$

$$L_{IoU} = 1 - \frac{I(X)}{U(X)}. \quad (11)$$

Here,  $V$  is the set of all of the pixels in the image,  $X$  is the pixel probabilities obtained by a sigmoid function, and  $Y \in \{0, 1\}^v$  are the ground-truth values.  $Y = 0$  represents the background pixel label, and  $Y = 1$  represents the object pixel label in this notation.

The implementation of the IoU loss in our application had to be done slightly differently, because our classification problem was non-binary. In our solution, we used a softmax approach to obtain the pixel probabilities for each class. Subsequently, we used the one-hot encoding to enable the binary classifiers to handle the multiclass ground-truth data. The  $L_{IoU}$  (Equation (11)) was used to calculate the losses for each class, as well as a summation of all of the losses, which was used to reflect the final loss of the network. Note that the weight-regularization loss approach [55] was not included in our implementation.

For a final comparison, the ResNet-FCN with cross-entropy loss, the ResNet-FCN with IoU loss, and the Atrous-FCN with IoU loss were all fine-tuned using the pre-trained model from DeepLab-ResNet [49]. Each network was trained for 50 epochs using a momentum update with the same initial learning rate of 0.01. The test results are shown in Table 2.

**Table 2.** The test result using FCN-based architectures.

	PA	MPA	MIoU	F1
ResNet-FCN <sup>1</sup>	92.94	68.25	63.34	71.42
ResNet-FCN <sup>2</sup>	93.07	71.44	66.01	74.12
Atrous-FCN <sup>2</sup>	92.52	72.18	66.52	74.39
SA-Net <sup>2</sup>	93.25	73.07	66.67	74.40

<sup>1</sup> Trained using cross-entropy loss function. <sup>2</sup> Trained using IoU loss function.

The results in Table 2 demonstrate the effectiveness of the IoU loss function. By only changing the loss function on a ResNet-FCN, the test accuracies improved with 3.19% on MPA and 2.67% on MIoU. In addition, the test result also confirmed the advantage of integrating the ResNet-FCN with atrous kernel. The Atrous-FCN reached the MIoU of 66.52%, which was a 3.71% improvement from the original atrous network (Table 1).

It should be noted that implementing atrous kernels on top of ResNet-FCN requires substantially more memory, which slows down the training process. This is because the atrous kernels maintain the larger feature maps of the deeper layers (output stride of eight). Therefore, convoluting through the larger feature maps and deeper architectures requires a significant amount of memory for holding the larger number of parameters. When using an 11GB GeForce GTX 1080 Ti, the Atrous-FCN required 43 min to process a single training epoch.

#### 4.3. The Stochastic Depth Extension

In order to speed up the Atrous-FCN, we therefore decided to integrate it with the stochastic depth paradigm. The resulting architecture is referred to as the *Stochastic Atrous Network (SA-Net)*.

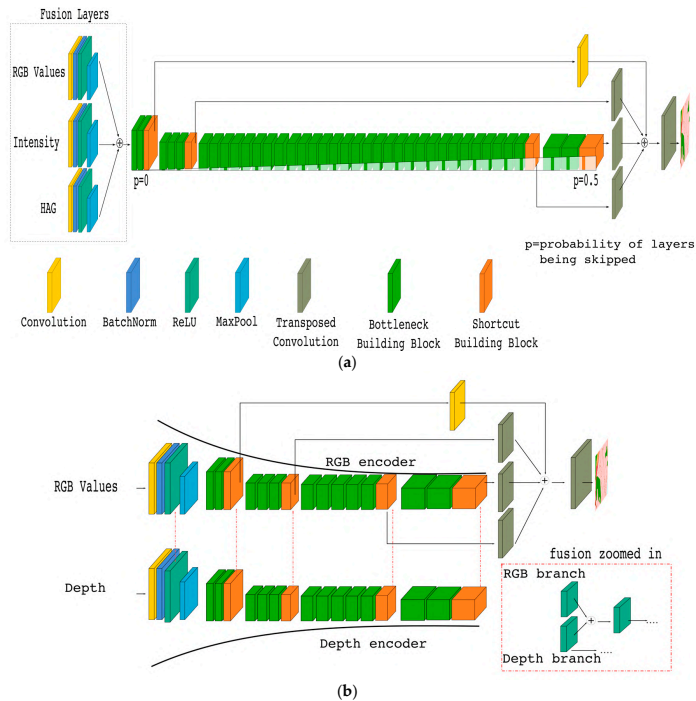
The stochastic depth paradigm comes with the idea of wanting much shorter training times. It is implemented by randomly skipping some layers in each epoch of the learning process [24]. Inclusion of the stochastic depth approach has been demonstrated both to speed up the learning process and cause an advantageous regularization effect for the training process.

In the original publication [24], stochastic depth in a residual building block of a ResNet is defined as:

$$H_l = \text{ReLU}(b_l f_l(H_{l-1}) + \text{id}(H_{l-1})), \quad (12)$$

where  $b_l$  denotes a Bernoulli random variable with values 0 or 1, and represents the active ( $b_l = 1$ ) or inactive ( $b_l = 0$ ) of the  $l$ th residual learning block, while the rest is a residual block with the ReLU activation function, which has been explained in Equation 1.  $b_l$  is controlled by another set of hyperparameters called survival probabilities and marked as  $p_l$ . This value decides the randomness degree of  $b_l$ . Stochastic depth networks commonly implement the linear decay of the value of its survival probabilities ( $p_l$ ): the deeper the layer, the smaller its probability for survival.

The integration of the stochastic depth approach with our atrous-FCN architecture is a straightforward procedure, because it is already developed for use with a ResNet architecture. The final structure of the suggested SA-Net architecture is shown in Figure 9a (at the present stage, ignore the fusion layers, except for the RGB input).



**Figure 9.** Data fusion technique based on the *Stochastic Atrous Network (SA-Net)* architecture. (a) Our proposed EarlyFusion architecture, which merges red–green–blue (RGB), intensity and height above ground (HAG) in the early convolution layers. (b) The FuseNet style architecture, which encodes RGB values and depth (HAG) using two branches of encoders, as inspired by C. Hazirbas et al.’s work [56].



We attached the stochastic depth mechanism to all of the bottleneck building blocks on Atrous-FCN by using an 0.5 linear decay. This means that the probability of a block being skipped is increased linearly up to 50% at the end of the final building block. Our training and test results show that by including the stochastic paradigm, the training time was reduced by 30%, while the test set MIoU values increased slightly to 66.67%, as seen in Table 2.

#### 4.4. The Final Extension Including More Features by Data Fusion

The purpose of data fusion in deep learning is to improve the resulting classifiers by merging more types of data (such as *intensity* and *HAG* from the LiDAR measurements) with the traditional RGB image data to get an even better representation of the problem domain. However, such merging is not always successful, and we have experienced that extending a deep learning architecture to incorporate non-RGB channels does not always lead to improved classification performance (the introduction of noisy data can sometimes corrupt the training process and damage the final classification performance).

An extension of the proposed SA-Net approach to include data fusion provides the final architecture of our investigation. The required extension from SA-Net is quite straightforward, as shown in Figure 9a. We refer to this architecture as the EarlyFusion SA-Net. The idea of data fusion in deep learning relates to the FuseNet architecture [56].

The original FuseNet architecture was built with the 16-layer VGGnet [33] using an encoder–decoder, SegNet-style architecture. The FuseNet uses two branches of encoders: one RGB encoder and one depth encoder. The output of each of the Convolution + Batch Normalization + ReLU (CBR) layers of the depth encoder is combined with the output of the corresponding CBR layers of the RGB-encoder using addition. The convolution block in a CBR consists of 64 convolution layers with  $7 \times 7$  kernels and a stride of two, followed by Batch Normalization [51] and the ReLU activation function.

Our implementation of the FuseNet is based on our SA-Net architecture (Figure 9b), where we include two branches of encoders, which were both created by using the stochastic atrous approach with different kernels. In the fusion process, we combine the output from each MainBlock of the depth-encoder branch with the RGB MainBlock output using the same fusion technique as in the original FuseNet. The decoder part was built using the FCN upsampling approach with transposed convolutions and skip connections.

The models obtained by training our implementation of the FuseNet style architecture turned out to perform below our expectations. With a big network and huge memory footprint, the best test set accuracy that we achieved was only 65.49% for MIoU and 93.38% for PA. One of the main challenges seemed to be that the separated branches of depth and RGB encoders consumed most of the graphics processing unit (GPU) memory. Due to this, the GPU was only capable of training 50 layers, compared with the 101 layers in the SA-Net architecture.

The experiences from the FuseNet style approach actually helped us in specifying the EarlyFusion SA-Net architecture (Figure 9a). Instead of creating separated encoders for different types of input data, we realized that using only one encoder might be advantageous. The solution was to assemble three CBR blocks for individually processing the RGB, Intensity, and HAG data. Each CBR block was equipped with its own kernel, and the outputs from the blocks were combined by addition into the fusion layers. Before taking the combinations, pooling layers using  $3 \times 3$  kernels with a stride of two were included at the end of each CBR block (to reduce the feature maps to have an output stride of four). The resulting fused inputs were then integrated with the SA-Net architecture, as shown in Figure 9a. Based on this approach, we managed to reduce the number of required blocks in the encoders, allowing the incorporation of additional types of input data.

Our EarlyFusion SA-Net approach was trained using a combination of RGB, Intensity, and HAG features (all normalized by using zero mean and unit variance). All of the networks were trained with the IoU loss function and an incorporation of momentum update for optimization [50] with an initial learning rate of 0.001.

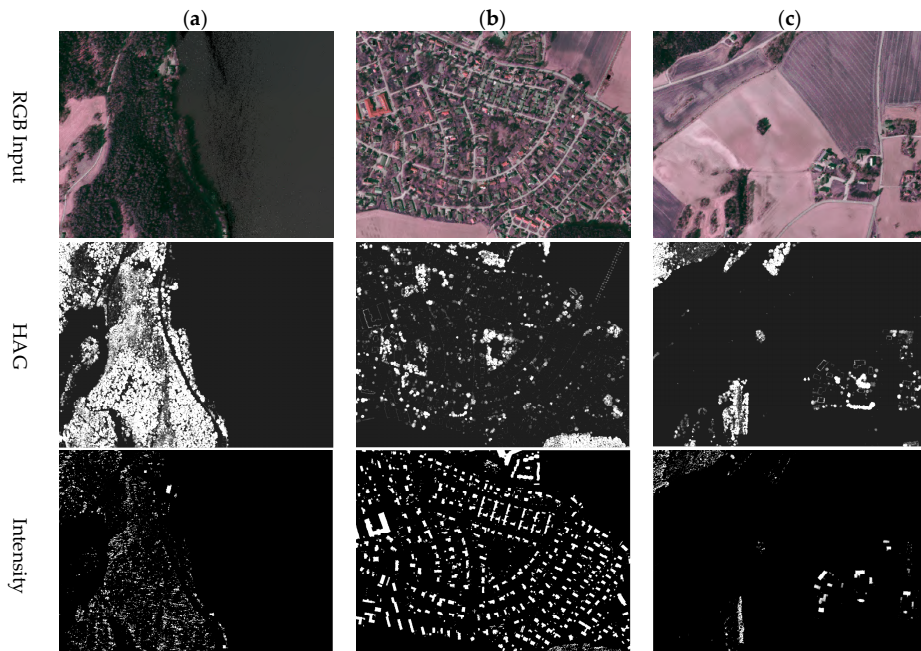
The models were fine-tuned with a modified pre-trained model inspired by the pre-trained FuseNet model described in C. Hazirbas et al.'s work [56]. The non-RGB kernels were initialized by using the mean value of the pre-trained RGB kernel (the values for the channels in the RGB kernel were summarized and divided by three, as the RGB has three input channels). This initialization strategy was applied for both the HAG and the Intensity convolution kernel. Each model was trained for 50 epochs, and the results are shown in Table 3.

**Table 3.** The test results for the data fusion approach using SA-Net-based architectures. F1: F-Measure.

	Features			Metrics			
	RGB	HAG	Intensity	PA	MPA	MIoU	F1
FuseNet style architecture <sup>1</sup>	v	v		93.38	72.08	65.49	73.53
Earlyfusion SA-Net	v	v		<b>94.02</b>	72.52	67.40	75.02
Earlyfusion SA-Net	v		v	91.80	69.80	63.78	72.09
Earlyfusion SA-Net		v	v	91.38	67.59	62.39	71.31
Earlyfusion SA-Net	v	v	v	93.96	<b>73.00</b>	<b>68.51</b>	<b>75.81</b>

<sup>1</sup> Based on SA-Net with ResNet 50.

Using the same types of input (RGB and HAG), our EarlyFusion approach and the Fusetnet style architecture obtained MIoU values of 67.40% and 65.49%, respectively. Incorporating intensity, our model achieved the highest MIoU values of 68.51%, which is a 5.7% relative improvement from the original atrous network (Table 1). The qualitative results of our EarlyFusion SA-Net is shown in Figure 10.



**Figure 10.** Cont.



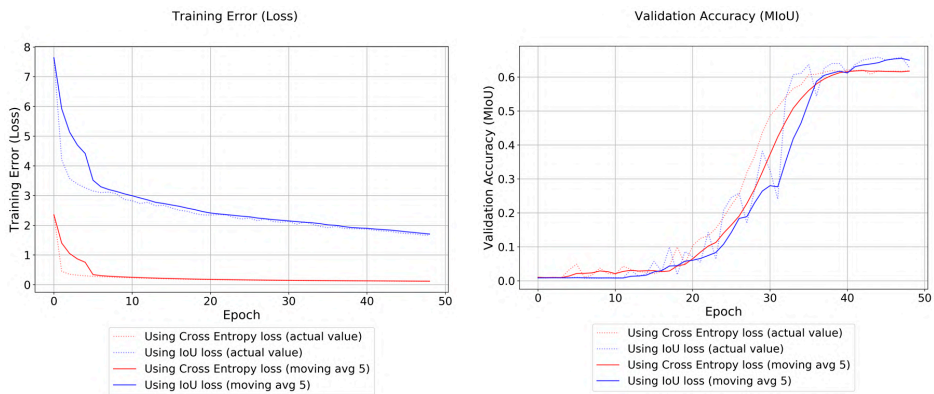


**Figure 10.** Final prediction results from our EarlyFusion SA-Net with RGB, HAG, and Intensity. The color legend is presented in the caption of Figure 6. The figures cover and visualize different types of areas, such as (a) forest and ocean, (b) settlement and road, and (c) cultivation and open land.

## 5. Discussion

### 5.1. Interesting Insights

We consider the most interesting finding of the present research project to be the significant contribution of the IoU loss function in increasing accuracies. By changing the loss function from cross-entropy loss to IoU loss, the test accuracies increased by more than 3%. It should be noted that the training error (loss values) of the IoU loss decreased steeper than for cross-entropy, as seen in Figure 11. We assume that this is not just due to the different scaling of loss values, but also due to a reflection of a better error representation.



**Figure 11.** Training errors and validation accuracies from the utilization of the cross-entropy and IoU loss functions on the deep residual network (ResNet)-FCN architecture. The training errors were calculated as the mean loss for all of the iterations in every epoch of a training process, while the validation accuracies were calculated at the beginning of each epoch using the validation data.

Looking through the behavior of the IoU loss in Figure 11, the accuracy of the ResNet-FCN could most likely be improved by additional training time. With 50 epochs of training time, the loss values

were still decreasing steadily, and the accuracies were increasing steeply even in the last training iterations. It should be noted that we stopped at 50 epochs to make the training iterations comparable with the number of training epochs executed for the other methods.

Another interesting finding was concerned with the stochastic depth paradigm. In our experience, the SA-Net with the stochastic depth approach protects against overfitting considerably better than the Atrous-FCN. Figure 12 shows the seemingly superiority of the Atrous FCN training accuracy (MIoU) over SA-Net, which is due to considerable overfitting. This was confirmed by the validation accuracies. The highest peak on the training accuracies (MIoU) for Atrous-FCN and SA-Net were 77.25% and 73.16%, respectively, and the validation accuracies for MIoU were 66.52% and 66.67%, respectively.

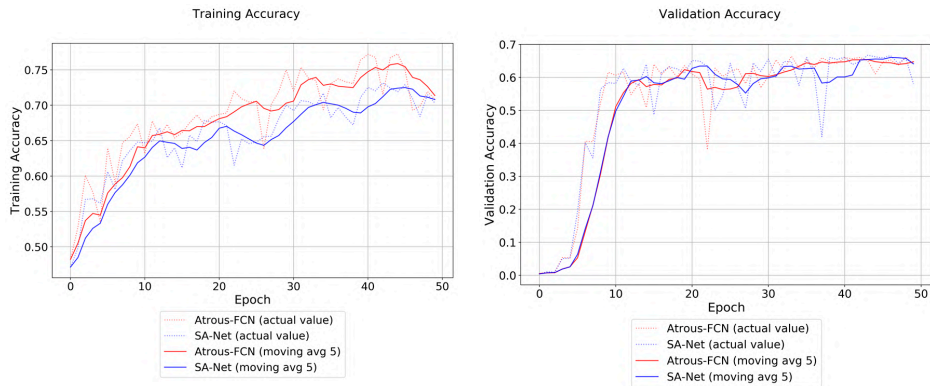


Figure 12. Training and validation accuracies for the Atrous-FCN and SA-Net.

SA-Net's ability to resist overfitting opens an opportunity in simplifying the training process. For limited training data, one can rely on the stochastic depth strategy to prevent overfitting and train the model with all of the data, including the validation data. With additional regularizations, such as weight regularization and dropout, this approach could contribute to even better predictions in terms of test set accuracies.

Another of our findings was related to the use of HAG and Intensity features to improve the prediction accuracies. The test set results in Table 3 shows a negative effect from combining RGB with Intensity only, reducing the test set MIoU values from 66.67% to 62.39%. LiDAR intensity is mainly determined by the reflectance characteristic of the reflector object, which shows the strength of the reflection of light [57]. This is similar to the optical sensors that produce RGB image data. This similarity amplifies the information contained in the RGB data, and also amplifies the noise in the data, resulting in a poor classifier.

In contrast, combining HAG with RGB improved the classification accuracies. HAG contains elevation information. This type of information is not captured in the image data, and our Earlyfusion architecture successfully manages to combine this into an improved classifier.

LiDAR data contain not only HAG and intensity features, but also Z-values, number of returns, flight angle, flight direction, and point classification. It is also possible to generate simple handcrafted features such as sparsity, planarity, linearity, and many more [13]. Incorporating those features with our fusion approach could potentially improve the performance of a resulting classifier.

## 5.2. Ground Truth Refinement

The different resolution/LoD of the LiDAR and the NIBIO AR5 data makes it difficult to obtain a good quantitative performance for the final model. Actually, we cannot expect (or aim for) a complete

match between the classification and the ground truth. A certain mismatch will always be present due to both temporal and geometrical differences between the input data and the data used as ground truth.

The qualitative assessments of the results from our best model show that the prediction map in many locations actually may be more correct than the ground truth map, because the prediction identifies areas where the land cover or land use has changed after the last incremental maintenance of the land resource map. This could make the proposed model a potentially powerful tool for a guided incremental update of the AR5 ground truth dataset, making it possible to produce higher-resolution agricultural maps.

Inspecting the final predictions in Figure 10 and using local knowledge, it is actually fair to claim that our prediction results are better than the labeled data. The image data in Figure 10b,c show that our classifier can correctly predict some roads and forest patches that have been ignored in the label data. Figure 10a also shows that the classifier can predict the existence of small settlement and cultivation classes, even if they have been removed from the label data as a result of the cartographic generalization process.

Our model can be used during the incremental update of AR5 by taking advantage of the probability prediction maps. The probability prediction maps are the output from the final softmax layer of the architecture. They show the confidence in the classification. A reasonable hypothesis is that locations classified differently by our model and the label data are likely to represent real changes, in particular if the probability of the prediction is high. This hypothesis has to be subject to another study. If it is correct, then our method can be used to streamline the maintenance of the AR5 dataset by efficiently directing the attention of the cartographers toward the areas where changes are most likely to be found.

Taking this use of the model one step further, human inspection or a trained regression classifier could be used to refine the thresholds and provide better trained weights. This approach could simplify the refinement process and increase the potential for full automation of the maintenance of the land resource map.

## 6. Conclusions

In this paper, several novel architectures for predicting land cover segmentation using airborne LiDAR data is presented. The main contribution of our research is the development of a scalable technique for doing dense-pixel prediction that incorporates image-based features and LiDAR-derived features, to update a generalized land resource map in Norway. With the aim to understand the behavior of ground-truth data constructed from different sources and with varying resolution of the label classes, we managed to develop a deep learning architecture (the EarlyFusion SA-Net) that is not only capable of predicting generalized classes, but also identifying the less common ones. In the preprocessing procedures of our work, we projected the three-dimensional (3D) laser data to a two-dimensional (2D) representation, and used RGB, HAG, and Intensity as the features of interest. As for the classifier, SA-Net was introduced, which is a deep learning architecture using atrous kernels on a ResNet-FCN architecture using the stochastic depth technique. The atrous kernel is robust for feature generalization based on spatial autocorrelation, the ResNet-FCN provides a simple yet powerful learnable upsampling technique, and the stochastic depth with its layer regularization is not only helpful for speeding up the training process, it is also capable of improving the prediction performance. The IoU loss was also implemented, and was shown to increase the accuracies for our particular dataset. In addition, an Earlyfusion layer proved helpful for combining image-based features and LiDAR-derived features.

Experiments were carried out on the Follo 2014 LiDAR dataset with the NIBIO AR5 as ground-truth data. Comparisons were made with other deep learning-based architectures for segmentation, namely DeconvNet, SegNet, FCN-8s, and the atrous network (DeeplabV2). Experimental results show that our proposed architectures were better than the other architectures. Accuracy, when calculated using MPA and MIoU, increased by more than 5%.

With the increasing availability of LiDAR data and the advancement of LiDAR technology, more and more high-quality LiDAR data can be accessed with ease. That and the maturing of deep learning technology can significantly simplify the automatic generation of high-resolution land cover, land-use, and land-change maps. Our research provides a scalable and robust way of transforming LiDAR data to a more meaningful map of information. Further improvement may be achieved by additional training with a deeper atrous layer, by implementing stochastic depth and fusion, or a combination of both. Improvements could also be achieved by adding more features of interest from the LiDAR data and by skipping the validation and increasing the use of stochastic procedures with additional regularization.

**Author Contributions:** H.A.A. wrote the paper, processed the data, performed the experiments, and proposed the architectures. G.H.S. provided the AR5 dataset, contributed to the discussion, and revised the paper. H.T. supervised the study and revised the paper. U.G.I. oversaw the research, provided the statistical analysis, and revised the paper.

**Acknowledgments:** We would like to thank J. Wu for providing the code for the ResNet-FCN. We also would like to thank all the Github authors, whose code we used for experiments. Norwegian Mapping Authority and NIBIO are acknowledged for providing the LiDAR data and the AR5 dataset, respectively. We gratefully acknowledge all the anonymous reviewers as well as their constructive notes and reviews, through which the manuscript was enriched and improved.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Norwegian Mapping Authority. Nasjonal Detaljert Høydemodell. Available online: <https://www.kartverket.no/Prosjekter/Nasjonaldetaljert-hoydemodell/> (accessed on 4 February 2018).
2. Norwegian Mapping Authority. Høydedata. Available online: <https://hoydedata.no/LaserInnsyn/> (accessed on 5 February 2018).
3. Jaboyedoff, M.; Oppikofer, T.; Abellán, A.; Derron, M.-H.; Loye, A.; Metzger, R.; Pedrazzini, A. Use of lidar in landslide investigations: A review. *Nat. Hazards* **2012**, *61*, 5–28. [[CrossRef](#)]
4. Koetz, B.; Morsdorf, F.; Van der Linden, S.; Curt, T.; Allgöwer, B. Multi-source land cover classification for forest fire management based on imaging spectrometry and lidar data. *For. Ecol. Manag.* **2008**, *256*, 263–271. [[CrossRef](#)]
5. Morsy, S.; Shaker, A.; El-Rabbany, A.; LaRocque, P.E. Airborne multispectral lidar data for land-cover classification and land/water mapping using different spectral indexes. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2016**, *3*. [[CrossRef](#)]
6. L'heureux, A.; Grolinger, K.; Elyamany, H.F.; Capretz, M.A. Machine learning with big data: Challenges and approaches. *IEEE Access* **2017**, *5*, 7776–7797. [[CrossRef](#)]
7. Bjørdal, I.; Bjørkelo, K. *Ar5 Klassifikasjonssystem—Klassifikasjon av Arealressurser*; Norsk Institutt for Skog og Landskap: Ås, Norway, 2014.
8. Guo, B.; Huang, X.; Zhang, F.; Sohn, G. Classification of airborne laser scanning data using jointboost. *ISPRS J. Photogramm. Remote Sens.* **2015**, *100*, 71–83. [[CrossRef](#)]
9. MacFaden, S.W.; O'Neil-Dunne, J.P.; Royar, A.R.; Lu, J.W.; Rundle, A.G. High-resolution tree canopy mapping for new york city using lidar and object-based image analysis. *J. Appl. Remote Sens.* **2012**, *6*, 063567. [[CrossRef](#)]
10. Yan, W.Y.; Shaker, A.; El-Ashmawy, N. Urban land cover classification using airborne lidar data: A review. *Remote Sens. Environ.* **2015**, *158*, 295–310. [[CrossRef](#)]
11. Zhou, W. An object-based approach for urban land cover classification: Integrating lidar height and intensity data. *IEEE Geosci. Remote Sens. Lett.* **2013**, *10*, 928–931. [[CrossRef](#)]
12. Guo, L.; Chehata, N.; Mallet, C.; Boukir, S. Relevance of airborne lidar and multispectral image data for urban scene classification using random forests. *ISPRS J. Photogramm. Remote Sens.* **2011**, *66*, 56–66. [[CrossRef](#)]
13. Grilli, E.; Menna, F.; Remondino, F. A review of point clouds segmentation and classification algorithms. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.* **2017**, *42*, W3. [[CrossRef](#)]
14. Poux, F.; Hallot, P.; Neuville, R.; Billen, R. Smart point cloud: Definition and remaining challenges. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2016**, *4*, 119. [[CrossRef](#)]
15. Barnea, S.; Filin, S. Segmentation of terrestrial laser scanning data using geometry and image information. *ISPRS J. Photogramm. Remote Sens.* **2013**, *76*, 33–48. [[CrossRef](#)]

16. Garcia-Gasulla, D.; Parés, F.; Vilalta, A.; Moreno, J.; Ayguadé, E.; Labarta, J.; Cortés, U.; Suzumura, T. On the behavior of convolutional nets for feature extraction. *arXiv* **2017**.
17. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet Classification with Deep Convolutional Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–8 December 2012; pp. 1097–1105.
18. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
19. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas Valley, NV, USA, 26 June–1 July 2016; pp. 770–778.
20. Castelluccio, M.; Poggi, G.; Sansone, C.; Verdoliva, L. Land use classification in remote sensing images by convolutional neural networks. *arXiv* **2015**.
21. Caltagirone, L.; Scheidegger, S.; Svensson, L.; Wahde, M. Fast lidar-based road detection using convolutional neural networks. *arXiv* **2017**.
22. Brust, C.-A.; Sickert, S.; Simon, M.; Rodner, E.; Denzler, J. Convolutional patch networks with spatial prior for road detection and urban scene understanding. *arXiv* **2015**.
23. Chen, L.-C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A.L. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv* **2016**.
24. Huang, G.; Sun, Y.; Liu, Z.; Sedra, D.; Weinberger, K.Q. Deep networks with stochastic depth. In *European Conference on Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 646–661.
25. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3431–3440.
26. Rahman, M.A.; Wang, Y. Optimizing intersection-over-union in deep neural networks for image segmentation. In *International Symposium on Visual Computing*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 234–244.
27. Everingham, M.; Eslami, S.A.; Van Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A. The pascal visual object classes challenge: A retrospective. *Int. J. Comput. Vis.* **2015**, *111*, 98–136. [[CrossRef](#)]
28. Tran, P.V. A fully convolutional neural network for cardiac segmentation in short-axis mri. *arXiv* **2016**.
29. Maggiori, E.; Tarabalka, Y.; Charpiat, G.; Alliez, P. Convolutional neural networks for large-scale remote-sensing image classification. *IEEE Trans. Geosci. Remote Sens.* **2017**, *55*, 645–657. [[CrossRef](#)]
30. Badrinarayanan, V.; Kendall, A.; Cipolla, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv* **2015**.
31. Noh, H.; Hong, S.; Han, B. Learning deconvolution network for semantic segmentation. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1520–1528.
32. Zhao, H.; Shi, J.; Qi, X.; Wang, X.; Jia, J. Pyramid scene parsing network. *arXiv* **2016**.
33. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**.
34. Wojna, Z.; Ferrari, V.; Guadarrama, S.; Silberman, N.; Chen, L.-C.; Fathi, A.; Uijlings, J. The devil is in the decoder. *arXiv* **2017**.
35. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), Haifa, Israel, 21–24 June 2010; pp. 807–814.
36. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. [[CrossRef](#)]
37. Garcia-Garcia, A.; Orts-Escolano, S.; Oprea, S.; Villena-Martinez, V.; Garcia-Rodriguez, J. A review on deep learning techniques applied to semantic segmentation. *arXiv* **2017**.
38. Blom Geomatics AS. Lidar-Rapport—Follo 2014. Available online: [https://hoydedata.no/LaserInnsyn/ProjektRapport?filePath=%5C%5Cstatkart.no%5Choydedata\\_orig%5Cvol1%5C119%5Cmetadadata%5CFollo%202014\\_Projektrapport.pdf](https://hoydedata.no/LaserInnsyn/ProjektRapport?filePath=%5C%5Cstatkart.no%5Choydedata_orig%5Cvol1%5C119%5Cmetadadata%5CFollo%202014_Projektrapport.pdf) (accessed on 18 June 2018).
39. Andrew, B.; Brad, C.; Howard, B.; Michael, G. Pdal: Point Cloud Data Abstraction Library. Release 1.6.0 ed. 2017. Available online: [https://2017.foss4g.org/post\\_conference/PDAL.pdf](https://2017.foss4g.org/post_conference/PDAL.pdf) (accessed on 18 June 2018).

40. NIBIO. Nibio ar5 Wms Service. Available online: <https://www.nibio.no/tema/jord/arealressurser/arealressurskart-ar5> (accessed on 22 May 2018).
41. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv* **2016**.
42. Bormann, F. Tensorflow Implementation of “Learning Deconvolution Network for Semantic Segmentation”. Available online: <https://github.com/fabianbormann/Tensorflow-DeconvNet-Segmentation> (accessed on 24 January 2018).
43. Kingma, D.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**.
44. Ørstavik, M. Segnet-Like Network Implemented in Tensorflow to Use for Segmenting Aerial Images. Available online: <https://github.com/mathildor/TF-SegNet> (accessed on 24 January 2018).
45. Duchi, J.; Hazan, E.; Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* **2011**, *12*, 2121–2159.
46. Srivastava, N.; Hinton, G.E.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
47. Shekkizhar, S. Tensorflow Implementation of Fully Convolutional Networks for Semantic Segmentation. Available online: <https://github.com/shekkizh/FCN.tensorflow> (accessed on 7 December 2017).
48. Vedaldi, A.; Lenc, K. Matconvnet: Convolutional neural networks for matlab. In *Proceedings of the 23rd ACM International Conference on Multimedia*; ACM: New York, NY, USA, 2015; pp. 689–692.
49. Vladimir. Deeplab-resnet Rebuilt in Tensorflow. Available online: <https://github.com/DrSleep/tensorflow-deeplab-resnet> (accessed on 7 December 2017).
50. Sutskever, I.; Martens, J.; Dahl, G.; Hinton, G. On the importance of initialization and momentum in deep learning. In *Proceedings of the International Conference on Machine Learning*, Atlanta, GA, USA, 16–21 June 2013; pp. 1139–1147.
51. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning*, Lille, France, 6–11 July 2015; pp. 448–456.
52. Krähenbühl, P.; Koltun, V. Efficient inference in fully connected crfs with gaussian edge potentials. In *Proceedings of the Advances in Neural Information Processing Systems*, Granada, Spain, 17–12 December 2011; pp. 109–117.
53. Wu, J. Resnet + Fcn (Tensorflow Version) for Semantic Segmentation. Available online: <https://github.com/wkcn/resnet-fcn.tensorflow> (accessed on 24 January 2018).
54. Dunne, R.A.; Campbell, N.A. On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function. In *Proceedings of the 8th Aust. Conf. on the Neural Networks*, Melbourne, Melbourne, Australia, 30 June–3 July 1997; p. 185.
55. Ng, A.Y. Feature selection,  $l_1$  vs.  $l_2$  regularization, and rotational invariance. In *Proceedings of the Twenty-First International Conference on Machine Learning*; ACM: New York, NY, USA, 2004; p. 78.
56. Hazirbas, C.; Ma, L.; Domokos, C.; Cremers, D. Fusetnet: Incorporating depth into semantic segmentation via fusion-based cnn architecture. In *Asian Conference on Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 213–228.
57. Song, J.-H.; Han, S.-H.; Yu, K.; Kim, Y.-I. Assessing the possibility of land-cover classification using lidar intensity data. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2002**, *34*, 259–262.



---

**Paper B:**  
**Addressing Overfitting on Point Cloud  
Classification using Atrous XCRF**

---

**Addressing Overfitting on Point Cloud Classification using  
Atrous XCRF**

Hasan Asy'ari Arief <sup>1,\*</sup>, Ulf Geir Indahl<sup>1</sup>, Geir-Harald Strand <sup>1,2</sup>, and  
Håvard Tveite<sup>1</sup>

<sup>1</sup>*Faculty of Science and Technology, Norwegian University of Life  
Sciences, 1432 Ås, Norway*

<sup>2</sup>*Division of Survey and Statistics, Norwegian Institute of Bioeconomy  
Research, 1431 Ås, Norway*

Published on ISPRS Journal of Photogrammetry and Remote Sensing  
(ISSN 0924-2716).





# Addressing Overfitting on Point Cloud Classification using Atrous XCRF

Hasan Asy'ari Arief<sup>a</sup>, Ulf Geir Indahl<sup>a</sup>, Geir-Harald Strand<sup>a,b</sup>, Håvard Tveite<sup>a</sup>

<sup>a</sup>Faculty of Science and Technology, Norwegian University of Life Sciences, Ås, 1432, Norway

<sup>b</sup>Division of Survey and Statistics, Norwegian Institute of Bioeconomy Research, Ås, 1432, Norway

---

## Abstract

Advances in techniques for automated classification of point cloud data introduce great opportunities for many new and existing applications. However, with a limited number of labelled points, automated classification by a machine learning model is prone to overfitting and poor generalization. The present paper addresses this problem by inducing controlled noise (on a trained model) generated by invoking conditional random field similarity penalties using nearby features. The method is called Atrous XCRF and works by forcing a trained model to respect the similarity penalties provided by unlabeled data. In a benchmark study carried out using the ISPRS 3D labeling dataset, our technique achieves 85.0% in term of overall accuracy, and 71.1% in term of F1 score. The result is on par with the current best model for the benchmark dataset and has the highest value in term of F1 score. Additionally, transfer learning using the Bergen dataset, without model retraining, was also performed. Even though our proposal provides a consistent 3% improvement in term of accuracy, more work still needs to be done to alleviate the generalization problem on the domain adaptation and the transfer learning field.

*Keywords:* Point cloud classification, Overfitting problem, Conditional Random Field

---

## 1. Introduction

The increased availability of high precision point cloud data, including airborne LiDAR (light detection and ranging) data, has opened interesting possibilities for many applications such as generating digital elevation models (Podobnikar and Vrečko, 2012), creating land use and land cover maps (Arief et al., 2018), 3D building reconstruction (VosDeepselman et al., 2001), and scene understanding in large dynamic environments (Zhao et al., 2010).

Improving the visual quality and accuracy of automated point cloud classification is an important topic in the computer vision, remote sensing and photogrammetry research communities. Many methods have been proposed to address this issue. Interesting examples includes the SVM (Support Vector Machine) based point cloud classification for urban areas (Mallet et al., 2008), the Random Forest combined with CRF (Conditional Random Field) approach (Niemeyer et al., 2014) for building detection, and the CNN (Convolutional Neural Network) approach for 3D semantic labeling (Yang et al., 2017; Yousefhusien et al., 2017; Zhao et al., 2018).

The use of DNN (Deep Neural Network) for point cloud classification has attracted considerable attention

in the last couple of years because of its potential of improving the quality of the automated classification task. The quantitative results from several point cloud classification benchmark datasets, such as 3D Shapenets (Wu et al., 2015), ScanNet (Dai et al., 2017), S3DIS (Armeni et al., 2016), and ISPRS Vaihingen 3D Labeling (Niemeyer et al., 2014) show that the majority of high performance classifiers for all these datasets are based on some choice of DNN model, such as MCNN (Zhao et al., 2018), PointNet++ (Qi et al., 2017), and PointCNN (Li et al., 2018).

An easy and yet robust implementations of DNN for point cloud data is known as the PointCNN (Li et al., 2018). The PointCNN uses a so-called  $\mathcal{X}$ -Transformation to allow a convolution operator to work directly within the point cloud data. In contrast to other methods such as voxel-based methods (Maturana and Scherer, 2015) and raster based methods (Zhao et al., 2018), the PointCNN reduces significantly both the required amount of time for preprocessing and the memory usage. Such advantages are important for real-time classification of point cloud data.

One of the main challenges of PointCNN and other DNN based models is that when only relatively limited size datasets are available, they are highly vulnerable to

overfitting. This is because such models usually include several millions of parameters, and robustly fitting such amounts of parameters requires a large number of training data. With the proposed Atrous XCRF method we obtain a novel way to overcome the overfitting problem by inducing controlled noise when training a DNN based classifier. The method works by retraining a validated model using unlabeled test data. The training supervision is directed by utilizing the hierarchical structure of the CRF penalty procedure (Krähenbühl and Koltun, 2011). In our experiment with the ISPRS 3D labeling benchmark dataset, we get an Overall Accuracy (OA) of 85.0% and an F1-Score of 71.1% (Niemeyer et al., 2014).

In addition, we also enriched our experiments with transfer learning challenge (Pan and Yang, 2009) using Bergen dataset (Norwegian Map Authority, 2018), i.e by predicting point-wise segmentation on a dataset that: (1) has different class distribution, (2) taken from different area and landscape, (3) acquired using different sensor settings, and (4) predicted without model retraining. As expected, the prediction accuracy is poor, however, a consistent 3% improvement can be noticed provided by our Atrous XCRF technique, from 87.2% to 90.2% in term of OA and from 17.1% to 20.1% in term of F1-score.

The present paper is organized as follows: In section 2, we provide a brief review of DNN, PointCNN, and CRF modelling. We also explain the XCRF and our proposed Atrous XCRF method for handling the overfitting problem. In the following section, we describe the experiment, including the data source, preprocessing procedure, training strategies, and results analysis. Thereafter, we discuss the limitations and characteristics of our proposed method. Finally, we provide the conclusions and indicate potential improvements of our novel technique.

## 2. Methodology

### 2.1. Brief review of DNN

Deep neural networks (DNN) or Deep Learning is an extension of a classical two-layer neural network (Blum and Rivest, 1989) using more (and wider) layers with some important enhancements. In contrast to the classical networks, the architecture of DNNs typically includes up to thousands of layers and up to millions of parameters which are normally trained using the complete architecture with an end-to-end fashion to achieve the best possible classification performance (LeCun et al., 2015).

The version of DNN currently most popular for image classification tasks is the CNN (LeCun et al., 1995). The CNNs extend the classical neural network principles into an extraordinary powerful classifier (Srivastava et al., 2014). They involve three basic operations, namely, convolutions, pooling operations, and non-linear activation functions (Schmidhuber, 2015).

A convolution operation is essentially a collection of dot product operations which allow a number of parameters (organized as a set of corresponding kernels) to aggregate on top of the feature maps provided by the previous layer to create the input to the subsequent layer. The convolution operations makes it possible for a CNN to capture spatial autocorrelation phenomena in the data into the resulting CNN model.

The pooling operation steps in CNN modelling is necessary for reducing the spatial size of the input feature map. Taking the maximum value of the features to represent the derived combined features is usually referred to as the max-pooling. In addition to reducing the spatial size of the feature map (e.g. reducing the need for memory storage and computation), the pooling operations also create a spatially generalised representation of the data.

Another important aspect of CNN modelling is the non-linear activation functions associated with the computational nodes in the network. It is there to obtain non-linearity in the transformations between the subsequent layers of the network. If omitted, a CNN (in fact any Neural Network) could be collapsed into a single linear transformation incapable of modelling the massively nonlinear phenomena present in most practical applications (Minsky and Papert, 1988). Most applications of CNNs use (some version) of a Rectified Linear Unit (ReLU) (Dahl et al., 2013) as their non-linear activation functions.

A CNN model is trained by the use of gradient descent methodology (Recht et al., 2011), which calculates the contribution of weight kernels towards the final loss value. The parameter update of a CNN model is based on the chain rule using the backpropagation algorithm (LeCun et al., 2015). The cross-entropy loss function (De Boer et al., 2005) is used to measure the precision of the trained CNN model, and reflects the degree of correspondence between the true and predicted class labels.

### 2.2. Brief review of the PointCNN

The convolution operation of CNNs is efficient for capturing the spatially-local correlations from regular and densely gridded datasets, such as images. The

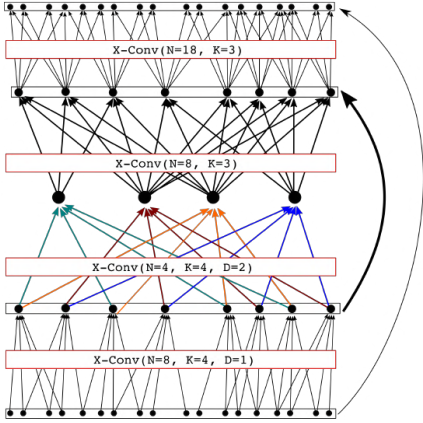


Figure 1: PointCNN architecture for point level prediction.

PointCNN introduces a modified version of the convolutions used in CNNs which we will refer to as the  $\mathcal{X}$ -Conv. The  $\mathcal{X}$ -Conv modifies the ordinary CNN convolution to work for irregular and unordered datasets such as point cloud data (Li et al., 2018). Because the  $\mathcal{X}$ -Conv can be used directly with irregular data, the need for preprocessing is significantly reduced.

Similar to a CNN convolution, the  $\mathcal{X}$ -Conv includes the calculation of inner products (element-wise product and summation operations) between feature maps and the convolution kernels. The  $\mathcal{X}$ -Conv takes into consideration the information of neighbour points among the features of interests, and finally transforms these features by a Multi-Layer Perceptrons (MLP) (Atkinson and Tatnall, 1997).

The  $\mathcal{X}$ -Conv operation is described in Algorithm 1. Here  $K$  denotes the number of neighboring points,  $p$  denotes the input point,  $\mathbf{P}$  denotes the  $K$  neighboring points, and  $\mathbf{F}$  denotes the previous feature representations of the  $K$  neighboring points.

$\mathcal{X}$ -Transformation, on the other hand, is in the lines 4-6 of the Algorithm 1 which transform the canonical point representation ( $\mathbf{P}'$ ) into  $K \times K$  matrix using MLP operation hence the term transformation. It should be noted that  $\mathcal{X}$  is dependent on the order of the input points because it will permute  $\mathbf{F}_*$  according to the specific input order. We refer the interested reader to the PointCNN paper (Li et al., 2018).

The PointCNN for point cloud segmentation is stacked into several Conv-DeConv blocks (Noh et al., 2015) using the U-Net architectural design (Ron-

---

**Algorithm 1**  $\mathcal{X}$ -Conv Operator, taken from (Li et al., 2018)

---

**Input:**  $\mathbf{K}, p, \mathbf{P}, \mathbf{F}$

**Output:**  $F_p$

- Features projected, or aggregated, into a representative point  $p$ .
  - 1:  $\mathbf{P}' \leftarrow \mathbf{P} - p$ 
    - Move  $\mathbf{P}$  to a local coordinate system with  $p$  as origo.
  - 2:  $\mathbf{F}_\delta \leftarrow MLP_\delta(\mathbf{P}')$ 
    - Individually lift each neighbor point into  $C_\delta$  dimensional space.
  - 3:  $\mathbf{F}_* \leftarrow [\mathbf{F}_\delta, \mathbf{F}]$ 
    - Concatenate  $\mathbf{F}_\delta$  and  $\mathbf{F}$ ,  $\mathbf{F}_*$  is a  $K \times (C_\delta + C_1)$  matrix.
  - 4:  $\mathcal{X} \leftarrow MLP(\mathbf{P}')$ 
    - Learn the  $K \times K$ ,  $\mathcal{X}$ -Transformation matrix.
  - 5:  $\mathbf{F}_X \leftarrow \mathcal{X} \times \mathbf{F}_*$ 
    - Weigh and permute  $\mathbf{F}_*$  with the learnt  $\mathcal{X}$ .
  - 6:  $F_p \leftarrow \text{Conv}(\mathbf{K}, \mathbf{F}_X)$ 
    - Finally, typical convolution between  $\mathbf{K}$  and  $\mathbf{F}_X$ .
- 

neberger et al., 2015). Similar to the U-Net, the Conv blocks are used to generate the global feature maps by maintaining local connectivity, while the DeConv blocks are used to propagate the global features into point level predictions. For the PointCNN, both the Conv and the DeConv blocks involves the  $\mathcal{X}$ -Conv operation but with a different number of points and receptive fields. Similar to the U-Net design, the output from the previous Conv block is forwarded not only to the next Conv block but also to the corresponding DeConv block, see Fig. 1 for details ( $K$  denotes the number of nearby points,  $N$  denotes the number of output representative, and  $D$  denotes the atrous distance). It should be noted that the PointCNN also includes dropout regularizing the Fully Connected (FC) layer to improve the accuracy of the resulting classifier (Srivastava et al., 2014).

### 2.3. Brief review of the Fully Connected CRF

A Conditional Random Field (CRF) is a probabilistic graphical model often used for sequence segmentation and labeling, capable of relaxing the strong independence assumptions of a graph model (Lafferty et al., 2001). A fully connected CRF (Krähenbühl and Koltun, 2011) is a variant of CRF that applies on a fully connected graph. For example, if the fully connected CRF is implemented on an image with its probability maps, the conditional penalty for a pixel in the image is con-

ditioned by the similarities and distances between that pixel and all the other pixels of the image.

The CRF can be characterized as representing a Gibbs distribution and for a fully connected CRF the corresponding Gibbs energy  $E(x)$  is defined as:

$$E(x) = \sum_i \psi_u(x_i) + \sum_{i < j} \psi_p(x_i, x_j), \quad (1)$$

where  $x$  denotes the label assignment (of the entire image),  $i$  and  $j$  denotes pixel locations,  $\psi_u(x_i)$  denotes the unary potential on pixel  $i$  (the unary potential is the result of an independent classifier) and  $\psi_p(x_i, x_j)$  denotes the pairwise potential between the labels  $x_i$  and  $x_j$  and is defined as:

$$\psi_p(x_i, x_j) = \mu(x_i, x_j) \sum_{m=1}^K w^{(m)} k^{(m)}(f_i, f_j), \quad (2)$$

$k^{(m)}(f_i, f_j)$  is a Gaussian kernel (Paris and Durand, 2006) that calculates the similarity between the feature vectors  $f_i$  and  $f_j$  for the pixels  $i$  and  $j$ , respectively. For multi-class classification, the Gaussian kernel is implemented as contrast sensitive two-kernel potentials using a weighted ( $w^{(m)}$ ) Gaussian filter (Tomasi and Manduchi, 1998). The pairwise potential is also weighted by the compatibility function denoted as  $\mu(x_i, x_j)$  using a Potts model (Krähenbühl and Koltun, 2011).  $\mu(x_i, x_j)$  penalizes nearby similar pixels that have different labels. A fully connected CRF is trained using iterative Mean Field Approximation, and the Gaussian filterings are computed using the permutahedral lattice (Adams et al., 2010), a high dimensional filtering algorithm. The details are explained in the above mentioned paper describing the fully connected CRF (Krähenbühl and Koltun, 2011).

A variant of a fully connected CRF, implemented by using convolution operations and trained end to end using a DNN principal, is the so-called CRF Recurrent Neural Network (CRF-RNN) (Zheng et al., 2015). The iterative CRF mean field operation of the CRF-RNN is structured as a stack of CNN layers, and the Gaussian filter is implemented using the permutahedral lattice where the filter coefficients convolves the weighted kernels on the lattice space. The mean-field iteration takes a weighted sum of the previous filter outputs for each class label, corresponding to a 1 by 1 convolution on every class label. The compatibility transform can also be seen as the convolution of a Potts model with the outputs calculated in the previous step. Finally, the update operation of a unary potential is obtained by adding the pairwise potentials and the current unary potential together. The updated outputs are then used as

the new unary potentials. The described operations are organized into a Recurrent Neural Network (RNN) architecture (Mikolov et al., 2010), so that gradient descent can be used to update the weighted Gaussian kernel and the CRF compatibility matrix.

#### 2.4. Training the artificial labels using Atrous XCRF

The Atrous XCRF (A-XCRF) can be explained as a variant of CRF-RNN, which has the same properties of calculating the pairwise similarities and penalizing according to the predictions. The main difference between the two is that the A-XCRF does not require a permutahedral lattice structure. The pairwise penalty of A-XCRF is implemented using a hollow matrix and one-hot encoding of the predicted label, and the method is used to refine a trained DNN model.

The X term in XCRF is associated with the X-Transformation in the PointCNN which utilizes the nearby points to create features of interest and uses atrous (Arief et al., 2018) indices for point selections, more explanation about the X-Transformation can be found in subsection 2.2. For point data, the atrous approach (see Fig. 2) means that the selected indices are not necessarily close to each other but closest by some number of intermediate (unselected) points, see (Arief et al., 2018) for a detailed explanation of atrous indices for raster / grid data.

While CRF-RNN is fully connected, XCRF is not fully connected and only considers the specified  $K$  number of nearby points. The intuition justifying its application is that a patch of point cloud data can be spread out in a very large region, and the points being far apart should not influence each other very much. By ignoring such distant pairs of points, we gain substantial savings computationally and in memory consumption. The XCRF is outlined in Algorithm 2.  $\mathbf{P}$ ,  $\mathbf{U}$ ,  $\mathbf{F}$  denote the matrices containing input points ( $\mathbf{P}$ ), current unaries potential ( $\mathbf{U}$ ), and existing features for each of the points ( $\mathbf{F}$ ), respectively.  $K$ ,  $D$ ,  $r$ ,  $I$  denote the number of nearby points ( $K$ ), an atrous distance between point indices ( $D$ ), the number of update iterations ( $r$ ), and nearby point indices ( $I$ ), respectively.

In line 1, for point  $\mathbf{p}$  in  $\mathbf{P}$ ,  $P_I$  gathers the indices of  $K$  nearby points according to the specified atrous distance ( $D$ ) from a list of the indices of the  $K \times D$  nearest points, sorted on the distance from  $\mathbf{p}$  in  $\mathbf{I}$ . Line 2-3, the similarity penalties between a point and its  $K$  neighbours is calculated using the Gaussian bilateral and spatial filters denoted  $B_f$  and  $S_f$ , respectively. These filter are defined as:

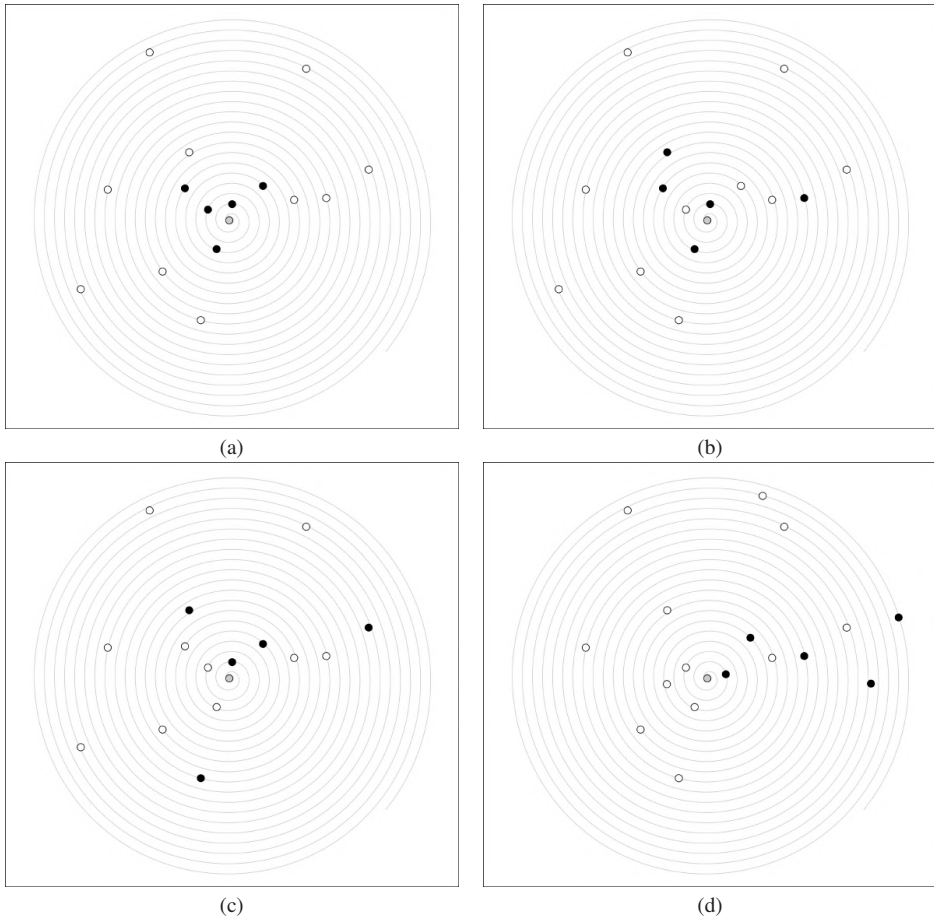


Figure 2: Atrous indices for point selections with  $K=5$ , the selected points are marked as black dots with the center point marked as grey, (a)  $D=1$ , (b)  $D=2$ , (c)  $D=3$ , and (d)  $D=4$ .

---

**Algorithm 2** XCRF Algorithm

---

**Input:**  $\mathbf{P}, \mathbf{U}, K, I, r, \mathbf{F}, D$ **Output:**  $U_1$ 

- Updated unary potentials respecting the  $K$  points similarity
  - 1:  $P_I \leftarrow \text{gather}(\mathbf{I}, D, K)$ 
    - For a point  $\mathbf{p}$  in  $\mathbf{P}$ , gather the  $K \times D$  nearest point indices  $\mathbf{I}$ , sorted on increasing distance, and skip  $D$  points for each gathered point index ( $P_I$ ).
  - 2:  $\mathbf{D}_{calc} \leftarrow \text{distance}(\mathbf{P}, \mathbf{F}, P_I)$ 
    - Calculate the euclidean distances between  $\mathbf{p}$  and  $\mathbf{P}[P_I]$ , and the distances between the feature values of  $\mathbf{F}_p$  and  $\mathbf{F}[P_I]$ .
  - 3:  $\mathbf{B}_f, \mathbf{S}_f \leftarrow \text{Gaussian}(\mathbf{D}_{calc})$ 
    - Implement the Gaussian filtering (Krähenbühl and Koltun, 2011) on  $\mathbf{D}_{calc}$ , with  $\mathbf{B}_f$  for the bilateral filter and  $\mathbf{S}_f$  for the spatial filter.
  - 4:  $\mathbf{G}_w \leftarrow \mathbf{B}_f \times \mathbf{W}_b + \mathbf{S}_f \times \mathbf{W}_s$ 
    - Passing the Gaussian weights ( $\mathbf{W}_b$  and  $\mathbf{W}_s$ ) on the previous outputs, as weighted Gaussian ( $\mathbf{G}_w$ ).
  - 5:  $\mathbf{U}_1 \leftarrow \mathbf{U}$ 
    - Duplicate original Unary ( $\mathbf{U}$ ).
  - 6: **while**  $i \leq r$  **do**
    - Update iteration as RNN, range( $r$ ).
  - 7:  $\mathbf{U}_s \leftarrow \text{softmax}(\mathbf{U}_1)$ 
    - Normalize the unary potential with the softmax function (LeCun et al., 2015).
  - 8:  $\mathbf{W}_u \leftarrow \text{OneHot}(\mathbf{U}_s) * \mathbf{W}_c$ 
    - Calculate hollow weighted unaries by using the dot product of the one hot encoding of the  $\mathbf{U}_s$  and a hollow weighted matrix ( $\mathbf{W}_c$ ).
  - 9:  $\mathbf{U}_G \leftarrow \mathbf{U}_s \times \mathbf{G}_w$ 
    - Pass the normalized unary to the weighted Gaussian output.
  - 10:  $\mathbf{U}_p \leftarrow \mathbf{U}_G * \mathbf{W}_u$ 
    - Calculate the pairwise penalty as a dot product of the weighted Gaussian and the compatibility hollow matrix.
  - 11:  $\mathbf{U}_1 \leftarrow \mathbf{U} - \mathbf{U}_p$ 
    - Update the unary values with the pairwise penalty, and after  $r$  iterations, return  $\mathbf{U}_1$  as the new unaries.
  - 12:  $i = i + 1$
  - 13: **end while**
- 

$$B_f = \exp\left(-\frac{|p_i - p_j|^2}{2\theta_\alpha^2} - \frac{|I_i - I_j|^2}{2\theta_\beta^2}\right), \text{ and} \quad (3)$$

$$S_f = \exp\left(-\frac{|p_i - p_j|^2}{2\theta_\gamma^2}\right) \quad (4)$$

where,  $p_i$  denotes the spatial ( $x, y, z$ )-coordinates of point  $p_i$  and  $I_i$  denotes the feature vector of  $p_i$ .  $\theta_\alpha, \theta_\beta$ , and  $\theta_\gamma$ , are the normalizing constants for the euclidean distances.  $B_f$  and  $S_f$  act as similarity penalties because their values increase as the associated euclidean similarities decrease (dissimilar features) and decrease as the euclidean similarities increase. In other words, the penalties are larger for nearby and similar points and smaller for more remote or dissimilar points. In step 4 of the XCRF-algorithm, the Gaussian outputs are weighted with kernels ( $\mathbf{W}_b$  and  $\mathbf{W}_s$ ) being updated in the training process of the complete architecture. In step 5, the original unaries are duplicated (to be updated in step 6-end). In steps 6, the unary potentials and the similarity penalties are combined to update the pairwise unary potentials as a RNN iteration, see also (Zheng et al., 2015). The effect of step 6 is that the original unary potential are recursively updated using the weighted Gaussian filters and the similarity label penalties using the hollow compatibility matrix.

The Gaussian filters in the XCRF algorithm yields a sharpening of the edge between two dissimilar points based on their normalized euclidean similarity distances. The weighting coefficients determine the amount of penalization according to the similarities. The hollow matrix and the one hot encoding output, on the other hand, works by penalizing label differences with weighted penalties and does not penalize equal labels, similar to the Potts model of the fully connected CRF (Krähenbühl and Koltun, 2011). XCRF can therefore be seen as a strict penalty procedure that is particularly sensitive to nearby similar points having different labels.

The proposed A-XCRF block builds on the XCRF algorithm, but includes a modification to work as a refinement block for DNN architectures, similar to the CRF-RNN. The main difference is that, in addition to the recurrent structure, the Atrous XCRF requires multiple  $D_s$  (atrous distances between point indices) to implement the hierarchical structure of the XCRF, see Fig. 3.

With  $U_{final}$  denoting the final unary values,  $n$  denoting the number of different  $D_s$  in  $D$ , and  $P_s$  denoting the collection of XCRF parameters as described for Algo-



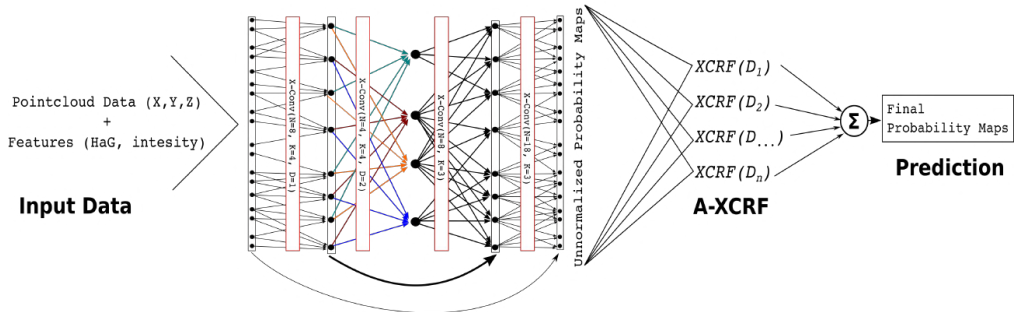


Figure 3: Implementation design of a full A-XCRF architecture.

gorithm 2, the A-XCRF is simply defined as:

$$U_{final} = \sum_{i=1}^n XCRF(Ps, D_i). \quad (5)$$

A two-step process is required to train the A-XCRF. The first step is seeking the best possible (in the validated sense) model for a DNN architecture by training the model with a split-validation approach. Inclusion of the split-validation part is important to prevent against harmful overfitting.

The second step includes training of the XCRF parameters against the validated model obtained in the first step. The training is carried out using the unseen data (unlabeled test data). The labels for these data are the predicted labels of the validated model, called artificial labels, see Fig. 4. The quality of the artificial labels obviously depends on the accuracy of the validated model from the first step.

The underlying idea of the second step is to introduce noise in the validated model by invoking the strict pairwise penalties of XCRF on the unseen data. The validated model is therefore forced to respect the XCRF penalties when fitted to the training data. By using this approach, one can invoke the differences between the calculated DNN probabilities obtained before and after they have passed through the A-XCRF refinement block.

Invoking the XCRF as a controlled noise generator can be highly useful when training a DNN model. This is particularly the case for PointCNN architectures where features generated from nearby points are limiting the potential use of data augmentations such as rotation and scaling. The introduction of a controlled

amount of noise acts as a regularizer on the DNN model that helps to overcome the ordinary limitations of point cloud data augmentation.

In order to maintain the accuracy of a validated model, the training process in the second step successively swaps between the two datasets (the training data and the unseen data with its artificial labels). The cross entropy loss function is used for training with both datasets. The parameter update using the backpropagation algorithm, on the other hand, works differently for the two datasets. For the training data, the backpropagation updates both the DNN kernels and the XCRF kernels. However, for the unlabeled data, the algorithm only updates the DNN kernels while the XCRF kernels are kept fixed, see Fig 4. By this approach, the updating of the DNN kernels works well with the training data, and at the same time respects the pairwise penalties of XCRF associated with the unseen data.

The trained model, including both the DNN- and A-XCRF parts of the architecture, makes up the resulting point classifier. The classifier works by passing the unary outputs from the DNN to the A-XCRF block for calculating the final unary potentials or the non-normalized prediction maps. A softmax normalization of the prediction maps generates the final class probability maps and the index of the highest class probability identifies the predicted class label for a particular point in the point cloud.

### 3. Experiments and Results

#### 3.1. Benchmark Dataset

The proposed method was evaluated using an open benchmark dataset, the ISPRS 3D labeling dataset

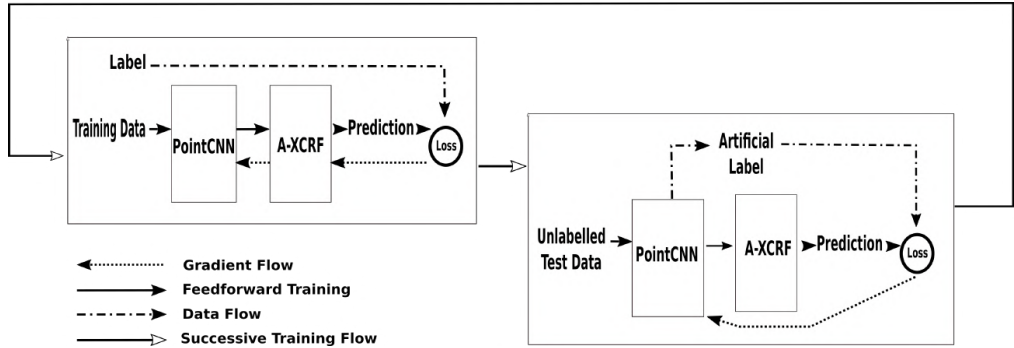


Figure 4: Training flow for the PointCNN with A-XCRF using labeled and unlabeled data.

Class	Number of Points	
	Training Data	Test Data
Powerline	546	-
Low Vegetation	180,850	-
Impervious Surfaces	193,723	-
Car	4,614	-
Fence/Hedge	12,070	-
Roof	152,045	-
Facade	27,250	-
Shrub	47,605	-
Tree	135,173	-
Total	753,876	411,722

Table 1: Class distribution of the Vaihingen 3D labeling dataset.

(Cramer, 2010). The labeling was provided by Niemeyer (Niemeyer et al., 2014). The dataset is an Airborne Laser Scanning (ALS) dataset acquired using a Leica ALS50 system with mean flying height 500 m above Vaihingen village in Germany. The dataset has a median point density of 6.7 points/m<sup>2</sup> and has nine classes for the 3D labeling task (powerline, low vegetation, impervious surfaces, car, fence/hedge, roof, facade, shrub, and tree). The dataset was divided into two parts, the training data and the test data with 753,876 points and 411,722 points, respectively. Both parts contained spatial coordinates (XYZ), intensity values and the number of returns for each point. Table 1 shows the class distribution for the training data.

The benchmark dataset was evaluated using the Overall Accuracy (OA) and the F1 score. OA is the percentage of points that were correctly classified, ignoring incorrect classifications. The F1 score is the harmonic average of precision and recall. The F1 score is more sen-

sitive to the unbalanced class distribution, as observed in the training data.

The F1 score is defined as follows:

$$\text{precision} = \frac{TP}{TP + FP}, \quad (6)$$

$$\text{recall} = \frac{TP}{TP + FN}, \text{ and} \quad (7)$$

$$F1 \text{ Score} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}, \quad (8)$$

where  $TP$ ,  $FP$  and  $FN$  denote True Positive, False Positive, and False Negative, respectively.

### 3.2. PointCNN and XCRF parameter setting

PointCNN works by training a batch of point blocks at once. It is therefore necessary to slice the dataset into point blocks before the model can be trained. For this purpose, 25m by 25m splitting blocks were used. The choice of block size was based on the point density of the dataset and the fact that PointCNN resamples and trains 2048 points in one forward pass (training iteration).

The training data was first sliced using 100m by 100m blocks (ignoring the Z axis), resulting in 12 blocks, each block containing between 25 000 and 120 000 points. Of these blocks, 80% were used for training and the remaining for validation, ensuring that there is no overlap between the training and validation data. Each of the 100m by 100m blocks were then sliced using 25m by 25m splitting blocks. Re-slicing were also performed by moving the initial slicing block by 12.5m (half sliced block) overlapping all the edges of the previous sliced blocks, hence increase the number of blocks and data points by repetition. The slicing process produced 286



and 44 blocks for the training and validation data, respectively. The number of points per 25m by 25m block varied between 1300 and 9000 points. The test data was also sliced using the same process, and produced 119 blocks of test data. It should be noted that the spatial coordinates for the points were transformed to local coordinate systems with origin at the center of the block the point belongs to.

For every training batch, 2048 points per block were randomly sampled by PointCNN. For blocks that have less than 2048 points, points were resampled with replacement. One 11GB GeForce GTX 1080 Ti graphics card was used for training, and the batch size was set to only six, due to capacity limitations of the GPU. It should be noted that increasing the batch size could improve the results, because PointCNN uses Batch Normalization (Ioffe and Szegedy, 2015) to reduce the internal covariate shift of the DNN parameters, and the technique performs better for a bigger batch size. Unless otherwise mentioned, the Tensorflow (Abadi et al., 2016) DNN library was used, and the training process started with a learning rate of 0.005, reducing it by 20% for every 5000 iterations, with a minimum learning rate of  $1e-6$ .

The XCRF weight parameters were initialized to one, assuming an equal contribution from each parameter toward the final loss. The compatibility matrix was also initialized to one, but multiplied with a zero diagonal matrix to produce a hollow matrix. The Gaussian filter parameters ( $\theta_\alpha$ ,  $\theta_\beta$ , and  $\theta_\gamma$ ) were initialized using a grid search, see (Krähenbühl and Koltun, 2011) for Gaussian filter initialization strategies. It should be noted that the filter parameters decide the size of the areas to be considered when calculating the pairwise similarities.

The XCRF uses five iterations ( $r$ ) to update the unary potentials both for the XCRF parameters update and for generating the final predictions. The A-XCRF is implemented on six hierarchies, with  $D$  equal to 1, 2, 3, 4, 8 and 16, respectively. For all of the levels a  $K$  (number of neighboring points) value of 64 was used, therefore the farthest neighboring point that is considered for the sixth level of the A-XCRF is the 1024th point ( $K=64$  by  $D=16$ ).

### 3.3. Training Strategies and Results

In order to test our proposed method, we trained the original PointCNN and the PointCNN with A-XCRF separately and analyzed the results to quantify the improvements. For the remainder of this paper, we use term "PointCNN" for the original PointCNN and "A-XCRF" for the PointCNN with Atrous XCRF. Both techniques were trained using  $x$ ,  $y$ ,  $z$  and two additional

Class	PointCNN	CRF-RNN	A-XCRF
Powerline	61.5	<b>68.5</b>	63.0
Low Vegetation	<b>82.7</b>	81.7	82.6
Impervious Surfaces	91.8	<b>92.1</b>	91.9
Car	<b>75.8</b>	75.7	74.9
Fence/Hedge	35.9	38.2	<b>39.9</b>
Roof	92.7	93.5	<b>94.5</b>
Facade	57.8	58.8	<b>59.3</b>
Shrub	49.1	50.1	<b>50.8</b>
Tree	78.1	79.5	<b>82.7</b>
Average F1	69.5	70.9	<b>71.1</b>
OA	83.3	83.6	<b>85.0</b>

Table 2: OA and F1 scores for PointCNN based techniques on the test data of the Vaihingen Dataset. All cells except the last two rows show F1 scores.

features: 1) Height Above Ground (HaG) generated using TerraScan, and 2) intensity, both of these additional features were normalized to the range  $[-0.5, 0.5]$ .

PointCNN and A-XCRF were trained using the split-validation dataset and the training was concluded after 10 consecutive training epochs without any change in the validation accuracies. We also trained the PointCNN with CRF-RNN using the same dataset and stop procedure. The PointCNN with CRF-RNN was trained end-to-end, as recommended in the CRF-RNN paper (Zheng et al., 2015).

Table 2 shows a comparison of the results on the test data for A-XCRF, PointCNN and PointCNN with CRF-RNN (CRF-RNN in the table). Fig. 5 shows the confusion matrices of PointCNN and A-XCRF. It can be noticed that A-XCRF slightly improves the OA on PointCNN for most of the classes (7 out of 9 classes). Similar to the PointCNN, A-XCRF has a problem with classifying the Powerline points. This is because the atrous approach of both models create holes in the group of neighbouring points while gathering nearby points, and this does not work well for linear features such as powerlines. CRF-RNN calculates similarity using all the nearby points, which seems to be a better strategy for such features.

Based on Table 2, it is clear that A-XCRF only offers a slight (between 1 and 2 percent) improvement in performance compared to the other techniques. However, A-XCRF is used for post-processing, and could be used as an extension to other machine learning-based classifiers, making it interesting as a tool for improving the quality of any classifiers. Fig. 6 shows the classification maps and error map produced by the A-XCRF.

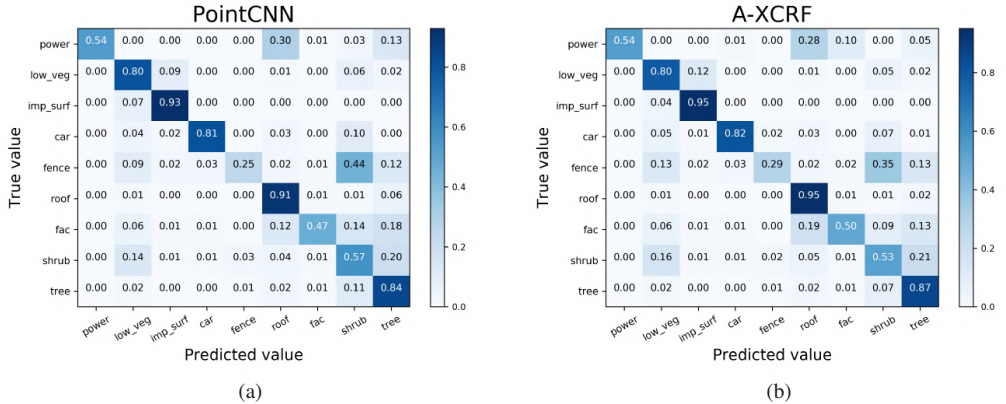


Figure 5: Comparison of the confusion matrices of (a) PointCNN and (b) A-XCRF.

### 3.4. ISPRS and Benchmark Results

A quantitative comparison between our method (A-XCRF) and other methods tested on the Vaihingen dataset are shown in Table 3. ISS\_7 (Ramiya et al., 2016) applied a supervoxel method on the point cloud data using the voxel cloud connectivity algorithm; then the local connectivity of the supervoxels were used to generate segmented objects; and finally, the generated segments were classified using machine learning techniques. UM (Horvat et al., 2016) used a combination of the point cloud attributes, textural properties and geometrical attributes generated using morphological profiles, and trained the features using One-vs-One (OvO), a multiclass machine learning technique. HM\_1 (Steinsiek et al., 2017) used k-nearest neighbors (KNN) to select neighboring points and eigenvalue-based features to generate geometrical features at the point level, then conducted the contextual classification using CRF. The LUH (Niemeyer et al., 2016) applied hierarchical higher order CRF by using two independent CRFs on the points and segments level (clustered points), respectively. It should be noted, that the majority of the techniques listed in Table 3 did not explain their stop procedure or their use of validation data. This information would be helpful when trying to replicate their result as well as when testing how A-XCRF would behave as a post-processing step for these methods.

RIT\_1 (Yousefhussein et al., 2017) and WhuY4 (Yang et al., 2018), are DNN based techniques for point cloud classification. RIT\_1 used a 1D-fully convolutional network with terrain-normalized points and spectral data. WhuY4 used a multi-scale CNN on the point to raster

representations, utilizing geometrical features such as planarity, sphericity, and variance of deviation angles, in addition to the HaG and intensity values.

It should be noted that there is another DNN method called NANJ2 (Zhao et al., 2018) but is not shown in Table 3. This is because it ignored the powerline class (Zhao et al., 2018), therefore the accuracy and the average F1 cannot be directly compared with the results from other methods. The NANJ2 generated deep features learned from the height, intensity, and roughness attributes and trained the features using multi-scale CNN.

### 3.5. Sensitivity Analysis

The A-XCRF technique uses several variables as input parameters, including the number of neighbors ( $K$ ) and the number of XCRF layers ( $n$ ) with their structure of atrous distances ( $D$ ). Table 4 shows the sensitivity of those variables with respect to the test accuracy.

The number of neighbor points considered for calculating the point similarity matters the most with respect to the OA. With  $K=8$ , the accuracy increases with the number of XCRF layers ( $n$ ), with a corresponding increase in the number of neighbor points to be considered. A too large  $K$  becomes harmful for the refinement quality, as demonstrated for  $K=128$ . In addition, with the increase of XCRF layers ( $n \geq 6$ ) with  $K=128$  comes the Out of Memory (OOM) problem. This is due to the large matrices ( $K$  by  $K$  by  $n$ ) that need to be stored in GPU memory for gradient update purposes. For this sensitivity analysis,  $K=64$  with  $n=6$  gives the most effective number of neighbor points (and XCRF structure)

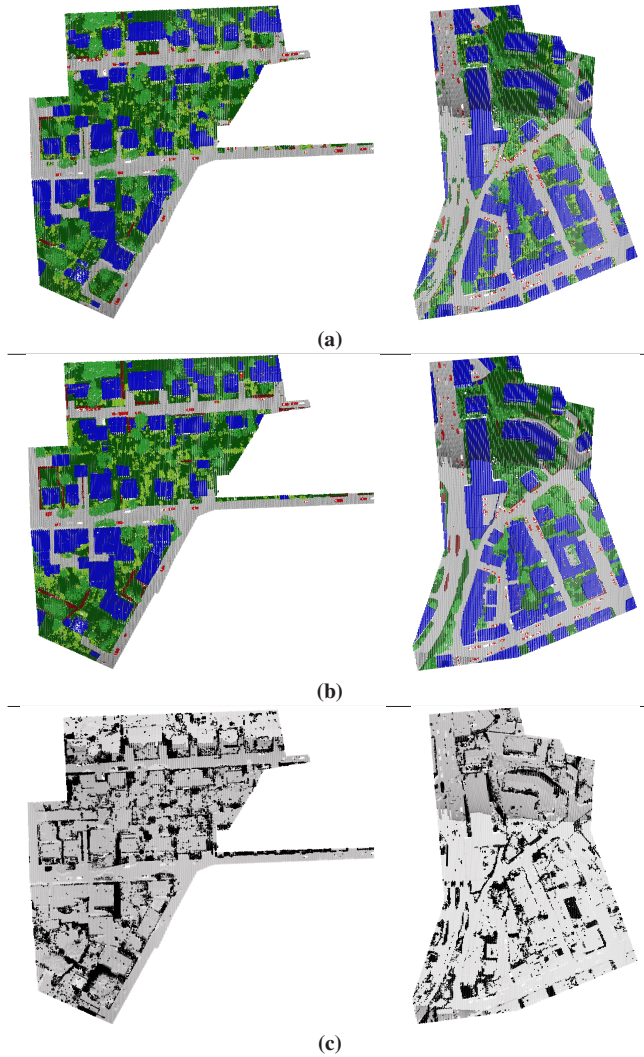


Figure 6: Classification map generated by (a) the A-XCRF method and (b) Vaihingen label test set, and (c) error map. Classes and colors: power line (orange), low vegetation (dark green), impervious surface (light gray), car (red), fence/hedge (dark red), roof (blue), facade (blue navy), shrub (green yellow), and tree (lime green).

Class	ISS.7	UM	HM.1	LUH	RIT.1	WhuY4	A-XCRF
Powerline	54.4	46.1	<b>69.8</b>	59.6	37.5	42.5	63.0
Low Vegetation	65.2	79.0	73.8	77.5	77.9	<b>82.7</b>	82.6
Impervious Surfaces	85.0	89.1	91.5	91.1	91.5	91.4	<b>91.9</b>
Car	57.9	47.7	58.2	73.1	73.4	74.7	<b>74.9</b>
Fence/Hedge	28.9	5.2	29.9	34.0	18.0	<b>53.7</b>	39.9
Roof	90.9	92.0	91.6	94.2	94.0	94.3	<b>94.5</b>
Facade	-	52.7	54.7	56.3	49.3	53.1	<b>59.3</b>
Shrub	39.5	40.9	47.8	46.6	45.9	47.9	<b>50.7</b>
Tree	75.6	77.9	80.2	83.1	82.5	82.8	<b>82.7</b>
Avg F1	55.27	58.96	66.39	68.39	63.33	69.2	<b>71.1</b>
OA	76.2	80.8	80.5	81.6	81.6	84.9	<b>85.0</b>

Table 3: A quantitative comparison between A-XCRF and other methods on the Vaihingen dataset. All cells except the last two rows show the per-class F1 score.

for implementing the AXCRF technique to generate a high accuracy classifier.

	$n=1$	$n=2$	$n=6$	$n=8$
$K=8$	83.71	83.68	84.45	84.69
$K=64$	84.50	84.70	84.97	82.65*
$K=128$	84.38	83.68	82.64*	82.68*

Table 4: Sensitivity analysis of A-XCRF input variables in terms of OA on the Vaihingen test data. \*Only calculated for inference without retraining due to the OOM problem during retraining.

### 3.6. Transfer Learning

We also include a transfer learning experiment using the Bergen dataset (Norwegian Map Authority, 2018). The idea is to show the applicability of the model (trained on the Vaihingen dataset) and the A-XCRF module for another dataset without retraining, and with a different LiDAR setting and environment (Pan and Yang, 2009).

The Bergen dataset is an ALS dataset acquired using a Riegl VQ-1560i mounted on a Piper Aircraft P-31-350 flying over the Bergen region in western Norway. The project was handled by Terratec and the data is co-registered with the R-G-B channels. The data are classified, including the ground class, low vegetation, medium vegetation, high vegetation, building, water, bridge and snow/ice (Norwegian Map Authority, 2018).

Due to the different classification schemes, we only use the roof class of the Vaihingen dataset to predict the building class of the Bergen dataset. We trained the PointCNN using the  $X$ - $Y$ - $HaG$  features of the Vaihingen training data, and tested the trained model using 20% of the Bergen dataset (100 files containing 719,762,528

data points). For A-XCRF we also used the  $R$ - $G$ - $B$  features of the Bergen dataset to calculate the point similarities, and trained the second-step of A-XCRF using only the roof class with unlabelled Bergen data.

Table 5 shows the quantitative results of PointCNN and A-XCRF on the Bergen dataset. A consistent 2-3% improvement in terms of accuracy is achieved. In addition Fig. 7 shows how the PointCNN prediction maps are improved when using the A-XCRF refinement technique.

It should be noted that even though a slight improvement can be noticed, the results are not yet of production quality. The quality of the probability maps provided by PointCNN for the Bergen dataset is poor. This is understandable because PointCNN has never been trained on the Bergen dataset. It was trained on the Vaihingen dataset, and there are many differences between these two datasets, including topography, size of trees, etc. In addition, being a DNN technique, PointCNN thrives with a large number of data points for training. In this case, the number of data point used for training and testing differs by a few order of magnitudes, significantly deteriorating the quality of the resulting classifier.

The A-XCRF, on the other hand, works on refining the probability maps by increasing or decreasing the probability value of a point belong to a certain class depending on the feature similarity of that particular point compared to its neighboring points. When the probability maps provided for A-XCRF is poor, the refinement results will also be limited, hence the quantitative results shown in Table 5.

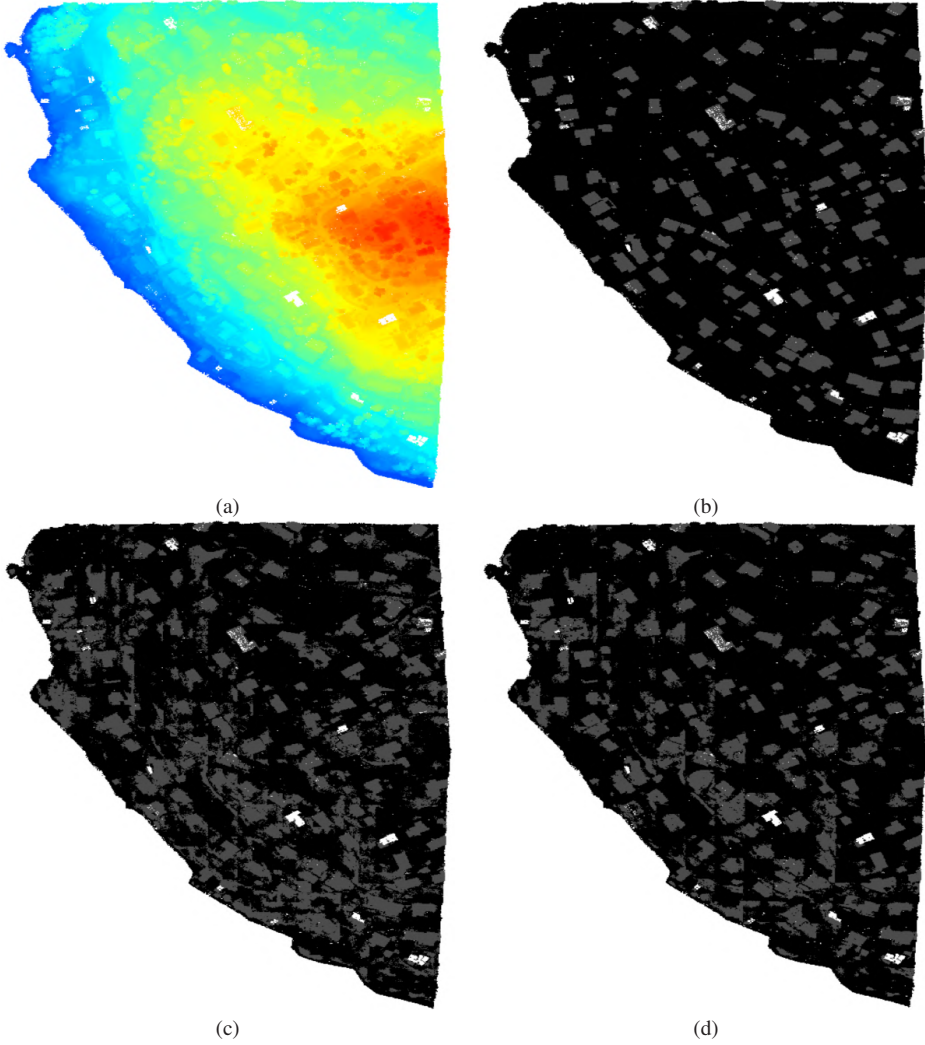
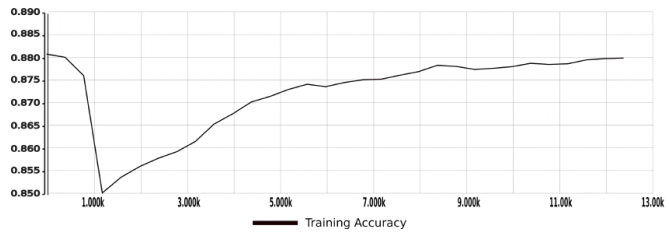
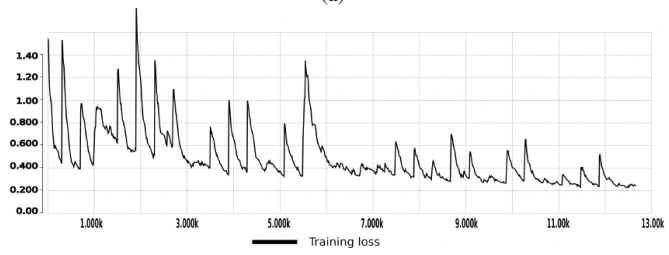


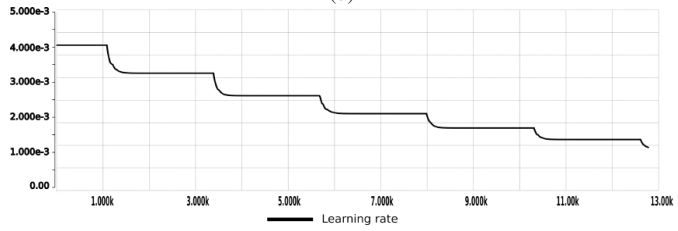
Figure 7: Top view of the segmentation maps on one of the files from the Bergen dataset (file id: 32-1-467-147-74). (a) point cloud data ( $X$ - $Y$ - $HaG$ ) colored by elevation, (b) label data using only building class, (c) the PointCNN prediction maps, and (d) the A-XCRF prediction maps.



(a)



(b)



(c)

Figure 8: Training summaries of A-XCRF, includes (a) Training accuracy, (b) Training loss, and (c) Learning rate.



	Precision	Recall	OA	F1-Score
PointCNN	09.5	83.7	87.2	17.1
A-XCRF	11.5	78.4	<b>90.2</b>	<b>20.1</b>

Table 5: The performance for transfer learning on 20% of the Bergen dataset.

## 4. Discussion

### 4.1. Limitations

The improvement offered by A-XCRF was about 2%, which may not be considered as very significant in terms of accuracy improvement. However, the use of artificial labels and XCRF similarity penalties to re-train a validated model is a novel idea and provides a basis for a future research direction.

In modelling with DNN architectures, parameter update and gradient descent works by tracking the gradient flows of the parameters and updating the parameters based on the accumulative loss caused by those parameters. Consequently, the distances between feature vectors are not emphasized in DNN modeling. Invoking the similarity penalties between features by using the CRF technique as a post-processing procedure to improve classification accuracy is not a new idea (Krähenbühl and Koltun, 2011; Chen et al., 2018; Zheng et al., 2015). However, this paper extends the idea by using unlabeled data with similarity penalties to improve the results, and the improvement is confirmed in Table 2 and 3.

Geometrical features calculated from the nearby  $K$  points can be used to strengthen the point similarity penalties. Sphericity, planarity, and deviation angle variance, proposed in (Yang et al., 2018), are examples of geometrical features that can be generated and used as additional features for the XCRF.

Including the geometrical features when calculating the pairwise similarity penalties may help the XCRF to provide better classification results on classes that have a linear geometrical shape, such as Powerline, Facade, and Fence/Hedge. Fig. 5 shows the unsatisfactory classification accuracies of those particular classes. Including geometrical features, such as planarity and sphericity, that are capable of detecting linearity from neighbouring points in the point cloud could contribute to overcome this problem. However, more research is needed to define a well-suited similarity penalty formula for such geometrical features.

The Gaussian kernel defined in Eq. 3 in (Krähenbühl and Koltun, 2011) includes a contrast sensitive two-kernel potential combining a bilateral and a spatial filter, that give higher penalties for smaller distances. This approach does not seem to be very well suited for stacking

many different features with different characteristics, because stacking them in the mentioned kernel seems to reduce the impact on the final similarity penalties. In other words, adding many different features in the Gaussian kernel are likely to both increase dissimilarity values and reduce the pairwise penalties. More work seems to be required to derive similarity formulas that works better with generated and dissimilar geometrical features.

### 4.2. End-to-end Atrous XCRF

When used as a post-processing module, the A-XCRF seems to be well suited to improve the prediction quality of machine learning based classifiers. However, the two loss functions of A-XCRF are making it more complicated to formulate the method as an end-to-end training process. This is because both loss functions update the same kernel (DNN parameters) and distributing the losses in an end-to-end fashion would result in intractable parameter updates.

Fig. 8 shows the training summaries of A-XCRF, including (a) the loss values, (b) the validation accuracy, and (c) the learning rate. The spikes in the training loss occur after every training epoch on the unseen data. The loss values were not decreasing monotonously, but display a nice decreasing trend. One notable phenomenon is that after the first learning rate decay, the validation accuracies demonstrate a relatively consistent improvement. This indicates that learning rate treatment and initialization strategy is important when training the A-XCRF.

End-to-end training and parameter update could simplify the A-XCRF learning process. Generative Adversarial Network (GAN) (Goodfellow et al., 2014) train both of GANs losses using minimax game theory (Salimans et al., 2016), with finding an equilibrium as the objective function. Using a GAN training style, we can set up both of the A-XCRF loss functions as the same minimax objective function and perform parameter update using the gradient flow and parameter update in the GAN architecture. Although both models are unstable and hard to train, A-XCRF with the minimax game algorithm could potentially be an end-to-end architecture with a better accuracy.

## 5. Conclusions

In this paper, a novel technique for addressing the overfitting issue for automatic classification of point cloud data is presented. The main contribution of our research is the proposal of an XCRF training algorithm

and an A-XCRF layer for training, utilizing the unlabeled part of a dataset to improve model accuracy. To address the overfitting behavior of DNN based models, we introduced a method that is not only capable of using similarity values between features of interest, but also induces a controlled noise in the validated model. In the preprocessing procedures of our work, we sliced the point cloud data using voxel blocks and used HaG and Intensity as the features of interest. PointCNN was used as a classifier. PointCNN is a DNN architecture that uses the  $X$ -Transformed technique to consume irregular and unordered data points using a convolution-like operator. For the post-processing, A-XCRF was used, which can be viewed as a DNN layer that forces the DNN models to respect the similarity penalties given by the unseen data. A-XCRF is a stack of XCRF modules that penalizes nearby similar points that have the dissimilar predicted label with a strict penalty procedure using a hollow compatibility matrix.

Experiments were carried out using the ISPRS 3D labeling benchmark dataset. Comparisons were made with PointCNN and PointCNN with CRF-RNN. In addition, a comparison with other techniques that has been tested on the benchmark dataset was presented. Experimental results show that our proposed technique was better than the other proposals in term of average F1 Score (71.1%) and the overall accuracy was on par with the current best proposal.

Experiments with the transfer learning challenge was also performed using the Bergen dataset. Even though the final accuracy is poor, our proposal consistently provides 3% improvement both in term of OA and F1-Score, from 87.2% to 90.2% and from 17.1% to 20.1%, respectively.

Further improvement may be achieved by introducing geometrical features in the XCRF algorithm to better handle classes with a linear geometrical shape and doing end-to-end training of A-XCRF layer using a GAN style architecture.

## Acknowledgement

We would like to thank all the Github authors, whose code we have used for our experiments, especially Li et al. (2018) for providing the PointCNN source code, the code we have used the most. The Vaihingen dataset was provided by the German Society for Photogrammetry, Remote Sensing and Geoinformation (DGPF) (Cramer, 2010) and the Bergen dataset was provided by the Norwegian Mapping Authority (Norwegian Map Authority, 2018). We also gratefully acknowledge all

the anonymous reviewers for their constructive notes and reviews.

## References

- Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al., 2016. Tensorflow: A system for large-scale machine learning, in: 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16), pp. 265–283.
- Adams, A., Baek, J., Davis, M.A., 2010. Fast high-dimensional filtering using the permutohedral lattice. *Computer Graphics Forum* 29, 753762. doi:10.1111/j.1467-8659.2009.01645.x.
- Arief, H., Strand, G.H., Tveite, H., Indahl, U., 2018. Land cover segmentation of airborne lidar data using stochastic atrous network. *Remote Sensing* 10, 973.
- Armeni, I., Sener, O., Zamir, A.R., Jiang, H., Brilakis, I., Fischer, M., Savarese, S., 2016. 3d semantic parsing of large-scale indoor spaces, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1534–1543.
- Atkinson, P.M., Tatnall, A.R., 1997. Introduction neural networks in remote sensing. *International Journal of remote sensing* 18, 699–709.
- Blum, A., Rivest, R.L., 1989. Training a 3-node neural network is np-complete, in: Advances in neural information processing systems, pp. 494–501.
- Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L., 2018. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence* 40, 834–848.
- Cramer, M., 2010. The dgpf-test on digital airborne camera evaluation—overview and test design. *Photogrammetrie-Fernerkundung-Geoinformation* 2010, 73–82.
- Dahl, G.E., Sainath, T.N., Hinton, G.E., 2013. Improving deep neural networks for lvcsvr using rectified linear units and dropout, in: 2013 IEEE international conference on acoustics, speech and signal processing, IEEE, pp. 8609–8613.
- Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M., 2017. Scannet: Richly-annotated 3d reconstructions of indoor scenes, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5828–5839.
- De Boer, P.T., Kroese, D.P., Mannor, S., Rubinstein, R.Y., 2005. A tutorial on the cross-entropy method. *Annals of operations research* 134, 19–67.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial nets, in: Advances in neural information processing systems, pp. 2672–2680.
- Horvat, D., Žalik, B., Mongus, D., 2016. Context-dependent detection of non-linearly distributed points for vegetation classification in airborne lidar. *ISPRS Journal of Photogrammetry and Remote Sensing* 116, 1–14.
- Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.
- Krähenbühl, P., Koltun, V., 2011. Efficient inference in fully connected crfs with gaussian edge potentials, in: Advances in neural information processing systems, pp. 109–117.
- Lafferty, J.D., McCallum, A., Pereira, F.C.N., 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data, in: Proceedings of the Eighteenth International Conference on Machine Learning, Morgan Kaufmann Pub-



- lishers Inc., San Francisco, CA, USA. pp. 282–289. URL: <http://dl.acm.org/citation.cfm?id=645530.655813>.
- LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. *nature* 521, 436.
- LeCun, Y., Jackel, L., Bottou, L., Cortes, C., Denker, J.S., Drucker, H., Guyon, I., Muller, U.A., Sackinger, E., Simard, P., et al., 1995. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective* 261, 276.
- Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B., 2018. Pointcnn: Convolution on x-transformed points, in: *Advances in Neural Information Processing Systems*, pp. 820–830.
- Mallet, C., Bretar, F., Soergel, U., 2008. Analysis of full-waveform lidar data for classification of urban areas. *Photogrammetrie Fernerkundung Geoinformation* 5, 337–349.
- Maturana, D., Scherer, S., 2015. Voxnet: A 3d convolutional neural network for real-time object recognition, in: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE. pp. 922–928.
- Mikolov, T., Karafát, M., Burget, L., Černocký, J., Khudanpur, S., 2010. Recurrent neural network based language model, in: *Eleventh annual conference of the international speech communication association, INTERSPEECH 2010*. pp. 1045–1048.
- Minsky, M.L., Papert, S.A., 1988. *Perceptrons: expanded edition*. MIT Press.
- Niemeyer, J., Rottensteiner, F., Soergel, U., 2014. Contextual classification of lidar data and building object detection in urban areas. *ISPRS journal of photogrammetry and remote sensing* 87, 152–165.
- Niemeyer, J., Rottensteiner, F., Soergel, U., Heipke, C., 2016. Hierarchical higher order crf for the classification of airborne lidar point clouds in urban areas. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences-ISPRS Archives* 41 (2016) 41, 655–662.
- Noh, H., Hong, S., Han, B., 2015. Learning deconvolution network for semantic segmentation, in: *Proceedings of the IEEE international conference on computer vision*. pp. 1520–1528.
- Norwegian Map Authority, 2018. *Laser Scan Report - Bergen Kommune 2018*. Technical Report. Terratec. URL: <https://hoydedata.no/LaserInnsyn/>.
- Pan, S.J., Yang, Q., 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 1345–1359.
- Paris, S., Durand, F., 2006. A fast approximation of the bilateral filter using a signal processing approach, in: *European conference on computer vision*, Springer. pp. 568–580.
- Podobnikar, T., Vrečko, A., 2012. Digital elevation model from the best results of different filtering of a lidar point cloud. *Transactions in GIS* 16, 603–617.
- Qi, C.R., Yi, L., Su, H., Guibas, L.J., 2017. Pointnet++: Deep hierarchical feature learning on point sets in a metric space, in: *Advances in Neural Information Processing Systems*, pp. 5099–5108.
- Ramiya, A.M., Nidamanuri, R.R., Ramakrishnan, K., 2016. A supervoxel-based spectro-spatial approach for 3d urban point cloud labelling. *International Journal of Remote Sensing* 37, 4172–4200.
- Recht, B., Re, C., Wright, S., Niu, F., 2011. Hogwild: A lock-free approach to parallelizing stochastic gradient descent, in: *Advances in neural information processing systems*, pp. 693–701.
- Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation, in: *International Conference on Medical image computing and computer-assisted intervention*, Springer. pp. 234–241.
- Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., 2016. Improved techniques for training gans, in: *Advances in neural information processing systems*, pp. 2234–2242.
- Schmidhuber, J., 2015. Deep learning in neural networks: An overview. *Neural networks* 61, 85–117.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1929–1958.
- Steinsiek, M., Polewski, P., Yao, W., Krzystek, P., 2017. Semantische analyse von als-und mls-daten in urbanen gebieten mittels conditional random fields. *Tagungsband* 37, 521–531.
- Tomasi, C., Manduchi, R., 1998. Bilateral filtering for gray and color images. *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)* doi:10.1109/iccv.1998.710815.
- VosDeepselman, G., Dijkman, S., et al., 2001. 3d building model reconstruction from point clouds and ground plans. *International archives of photogrammetry remote sensing and spatial information sciences* 34, 37–44.
- Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J., 2015. 3d shapenets: A deep representation for volumetric shapes, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 1912–1920.
- Yang, Z., Jiang, W., Xu, B., Zhu, Q., Jiang, S., Huang, W., 2017. A convolutional neural network-based 3d semantic labeling method for als point clouds. *Remote Sensing* 9, 936.
- Yang, Z., Tan, B., Pei, H., Jiang, W., 2018. Segmentation and multi-scale convolutional neural network-based classification of airborne laser scanner data. *Sensors* 18, 3347.
- Yousefhussein, M., Kelbe, D.J., Ientilucci, E.J., Salvaggio, C., 2017. A fully convolutional network for semantic labeling of 3d point clouds. *arXiv preprint arXiv:1710.01408*.
- Zhao, H., Liu, Y., Zhu, X., Zhao, Y., Zha, H., 2010. Scene understanding in a large dynamic environment through a laser-based sensing, in: *2010 IEEE International Conference on Robotics and Automation, IEEE*. pp. 127–133.
- Zhao, R., Pang, M., Wang, J., 2018. Classifying airborne lidar point clouds via deep features learned by a multi-scale convolutional neural network. *International Journal of Geographical Information Science* 32, 960–979.
- Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.H., 2015. Conditional random fields as recurrent neural networks, in: *Proceedings of the IEEE international conference on computer vision*. pp. 1529–1537.



---

**Paper C:**  
**Density-Adaptive Sampling for Heterogeneous  
Point Cloud Object Segmentation in  
Autonomous Vehicle Applications**

---

**Density-Adaptive Sampling for Heterogeneous Point Cloud  
Object Segmentation in Autonomous Vehicle Applications**

Hasan Asy'ari Arief<sup>1,2</sup>, Mansur Maturidi Arief<sup>2</sup>, Manoj Bhat<sup>2</sup>, Ulf Geir Indahl<sup>1</sup>, Håvard Tveite<sup>1</sup>, and Ding Zhao<sup>2</sup>

<sup>1</sup>*Faculty of Science and Technology, Norwegian University of Life Sciences, 1432 Ås, Norway*

<sup>2</sup>*Mechanical Engineering Department, Carnegie Mellon University, Pittsburgh, PA, USA*

Published on The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (2019).



# Density-adaptive Sampling for Heterogeneous Point Cloud Object Segmentation in Autonomous Vehicle Applications

Hasan Asy'ari Arief  
Norwegian University of Life Sciences  
Ås, 1432, Norway  
hasanari@nmbu.no

Manoj Bhat  
Carnegie Mellon University  
5000 Forbes Avenue, Pittsburgh, PA, USA  
mbhat@andrew.cmu.edu

Håvard Tveite  
Norwegian University of Life Sciences  
Ås, 1432, Norway  
havard.tveite@nmbu.no

Mansur Maturidi Arief  
Carnegie Mellon University  
5000 Forbes Avenue, Pittsburgh, PA, USA  
marief@andrew.cmu.edu

Ulf Geir Indahl  
Norwegian University of Life Sciences  
Ås, 1432, Norway  
ulf.indahl@nmbu.no

Ding Zhao  
Carnegie Mellon University  
5000 Forbes Avenue, Pittsburgh, PA, USA  
dingzhao@cmu.edu

## Abstract

*Robust understanding of the driving scene is among the key steps for accurate object detection and reliable autonomous driving. Accomplishing these tasks with a high level of precision, however, is not trivial. One of the challenges come from dealing with the heterogeneous density distribution and massively imbalanced class representation in the point cloud data, making the crude implementation of deep learning architectures for point cloud data from other domains less effective. In this paper, we propose a density-adaptive sampling method that can deal with the point density problem while preserving point-object representation. The method works by balancing the point density of pre-gridded point cloud data using oversampling, and then empirically sample points from the balanced grid. Using the KITTI Vision 3D Benchmark dataset for point cloud segmentation and PointCNN as the classifier of choice, our proposal provides superior results compared to the original PointCNN implementation, improving the performance from 82.73% using voxel-based sampling to 92.25% using our proposed density-adaptive sampling in terms of per class accuracy.*

## 1. Introduction

The ever-growing availability of large-scale point cloud data and easy access to affordable Light Detection and

Ranging (LiDAR) sensors have opened new streams of research in the computer vision field in recent years [10]. Instead of having to estimate distances between objects in a scene from images in order to reconstruct a 3D scene, researchers can now achieve the same objective by leveraging hundreds of thousands of point coordinates. PointNet [3] and PointCNN [13] are two popular deep learning models for learning the features of a scene from raw point cloud data, achieving an appealing performance of 83.7% for PointNet and 86.14% for PointCNN in terms of part-averaged IoU when applied on the ShapeNet Parts dataset [13]. Due to the novelty of its approach in accounting for the spatial information in the point cloud data, PointNet has commonly been used as the backbone of many recently developed deep learning models for raw point cloud data [17, 21].

Point cloud datasets usually contain an extremely large number of points, preventing deep learning models to include all the data points simultaneously. In that setting, it is common practice that only a few samples are considered in one learning iteration. The point sampling procedure in PointNet++, for example, is carried out by first partitioning the point cloud data into several voxels, and then performing random sampling from each of the voxels to construct training sets with the maximum number of points that is permissible by the architecture [18]. This approach presumes that the point density in each voxel is homogeneous. While the homogeneous point density distribution setting

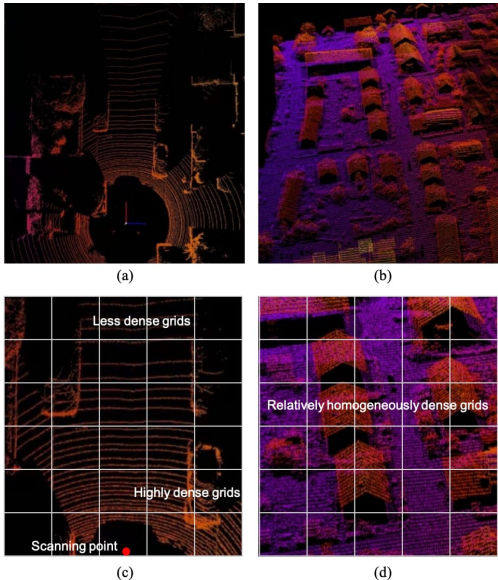


Figure 1. The nature of point cloud data from two different domains: (a) driving scene point cloud from Velodyne-type LiDAR (b) landscape map point cloud from airborne LiDAR along with point density distributions (c-d).

usually holds true for some applications, e.g. geomapping with point clouds from airborne LiDAR [1] or object scanning with point clouds from close-range scanning LiDAR [5], it is severely violated in the context of autonomous vehicle (AV) applications. In this context, the point cloud data are collected using a LiDAR with rotating beams (such as the Velodyne LiDAR) and the AV being in the center of the point clouds, thus the further the distance is, the less dense the points become (see for instance the Kitti dataset [9]). Because of this heterogeneity, the standard sampling technique with the voxelization method is not suitable. Fig. 1 illustrates the problem. In this figure, point cloud data from the Kitti Velodyne lidar (for a driving scene) and from the ISPRS Vaihingen airborne LiDAR (for a mapping application) are presented along with the top-view voxelization grids that are commonly used in training deep learning models.

The main contribution of our work is the use of a density-adaptive sampling method for improving the classification accuracy for heterogeneous point clouds with highly imbalanced class representation, especially when constructing the training set with an intention of balancing the point density distribution of the point cloud during the training process.

Other approaches have been studied in the context of learning from imbalanced datasets [2, 11]. We argue that obtaining point samples from grids of point cloud data with homogeneous point density distributions provides a good training set that is suitable with deep learning architectures utilizing raw point cloud data as input. In the experiment, we investigate the performance of various sampling strategies and observe that a class of sampling methods that are density-adaptive, i.e. taking into account the distribution of point density into the sampling scheme, yields a superior result compared to other sampling methods, including the crude voxelization-based sampling that is implemented in several existing deep learning architectures [13]. The experimental results using the Kitti 3D dataset show a significant improvement, from 82.73% using the original sampling to 92.25% using our proposed density-adaptive sampling, in terms of per class accuracy. We note that the performance of the proposed method is compared with accuracy metrics (true positive rate), rather than mIoU score, because our model discovered more objects compared to what the Kitti ground truth semantic labels provide, hence computing the mIoU score unfairly exacerbates our performance. A more detailed discussion will be provided in Section 5.

In what follows, we provide some background information describing the context of the challenges of work and review some related works. In Section 3, we present our method and the metrics used for evaluation. In Section 4, we describe the experiment settings and the Kitti dataset and show our results. In Section 5, we discuss our findings and finally we present the conclusion in Section 6.

## 2. Related Work

In this section, we provide an overview of the nature of point cloud data and an overview of selected work related to semantic segmentation using raw 3D point cloud data and sampling methods for learning models.

### 2.1. Point cloud and LiDAR

The growing interest analyzing point cloud data that capture spatial features of complex 3D scenes have become a trend in various fields, including mapping, object reconstruction, driving navigation, etc. A point cloud is a set of points in 3D space, each described by  $x$ ,  $y$ ,  $z$  coordinate values that are usually collected by LiDAR sensors. The sensor sends out pulses of beams at high frequency and calculates object distances based on the time for the pulses to return. By sweeping a region of interest with the LiDAR beams and combining the information from all points in a scene, usually in the order of hundreds of thousands or even a few million points, a data collection effort using LiDAR can provide accurate spatial information of complex scenes quite efficiently.

The point cloud data inherit unique characteristics, e.g. in terms of the distribution of point density in various sub-regions or grids of the scene or object of interest, depending on the nature of the scene of interest and the types of LiDAR utilized [16, 20]. Point cloud data from Airborne LiDAR, for instance, which is widely used for mapping geographical areas from a top-view, generally have a relatively even point density in each grid of the scene. Point clouds obtained from a close-range portable 3D object scanner LiDAR will have a point density distribution according to the surface captured by the scanning process. In contrast, point cloud data collected from Velodyne-type LiDARs, which is prevalent in AV applications with 360-degree surrounding environment as the scene of interest, will have highly heterogeneous point density in its grids. Fig. 1 shows the difference of point density distribution for different types of point cloud data.

## 2.2. Deep learning for point cloud segmentation

On its own, obtaining accurate semantic segmentation label from scenery data has less meaningful practical uses. However, semantic segmentation has been shown to be a good processing mechanism for object detection on point cloud data [6, 15]. That is, the scene segmentation task, which often is implemented as a supervised learning scheme, can be coupled with a classification or clustering method to build an accurate and efficient object detection pipeline or end-to-end framework. Hence, achieving a good performance for semantic segmentation on point cloud is a good intermediate objective. [21] provides a good review of the various frameworks used for semantic segmentation for point cloud data.

On a high level, these frameworks can be divided into several classes based on the input data that are exploited in the training of the deep learning models. Some frameworks [4, 14, 22] use an image projection or voxel representation of the point cloud data making it ready for the convolutional operations in the deep architectures. PointNet++ and PointCNN use the raw point cloud data directly which to some extent should have the advantage of being able to exploit the spatial information from the point cloud data. In these frameworks, the training set is constructed via sampling method. In the original PointNet++ implementation, the sampling is done by partitioning the scene of interest into several overlapping partitions by some distance metric, from which the local features are extracted. The local features are then clustered into larger units in a hierarchical fashion to capture the features of the whole scene [18]. PointCNN uses point cloud data and learns by utilizing the so-called X-Conv operations [13]. Similar to the convolution operation in ConvNet [12], X-Conv includes the calculation of inner products of transformed point cloud data and convolution kernels. The learning process relies on the

Multi-Layer Perceptron (MLP) algorithm and uses a Unet-like architecture [19] to do point-level segmentation. The implementation of both methods on the ShapeNet data set have yielded appealing results: 85.1% for PointNet++ and 86.14% for PointCNN. These methods, however, have not obtained the same performance when applied on point cloud data from Velodyne LiDAR, such as the Kitti dataset.

## 2.3. Sampling methods

Selection of the training data for training a deep learning model is always a critical task with respect to future good generalization. The large number of point clouds in a scene combined with the heterogeneous point density distribution and the implementation of complex deep learning architectures necessitate an efficient sampling scheme that is capable of selecting a good set of points that are informative for training purposes. In the machine learning literature, this problem is closely related to learning from imbalanced classes. Researchers have proposed the use of various sampling strategies, including random sampling, oversampling, undersampling, and stratified sampling, to achieve certain criteria that balance the training set [2, 7]. Interested readers are referred to [11] and [7] for more extended discussions about training set optimization for deep learning models.

## 3. Methodology

Our proposed method uses density-adaptive sampling that can be achieved by performing oversampling in grid cells where the number of points is below a certain threshold. In this section, we will elaborate how this density-adaptive sampling scheme is implemented as part of semantic segmentation framework to assist the learning models learn the most scene features from the training data.

### 3.1. Semantic segmentation framework

Semantic segmentation for point cloud data is essentially a point-wise classification task, where each point in the point cloud is classified into the class-object the point belongs to. In the AV context, point cloud segmentation is utilized to assign object class (such as car, pedestrian and cyclist) to each point and to use the segmented points for generating object bounding boxes. We used a step-wise semantic segmentation pipeline for Velodyne-based point cloud data by implementing a density-adaptive sampling technique in the preprocessing of the input data. We then employed PointCNN feature learning to generate probability maps. See Fig. 2 for an illustration of the framework.

### 3.2. Sampling method

To address the density problem, we utilize density-adaptive sampling method. The key idea is that feature learning using more balanced training sets will ease the

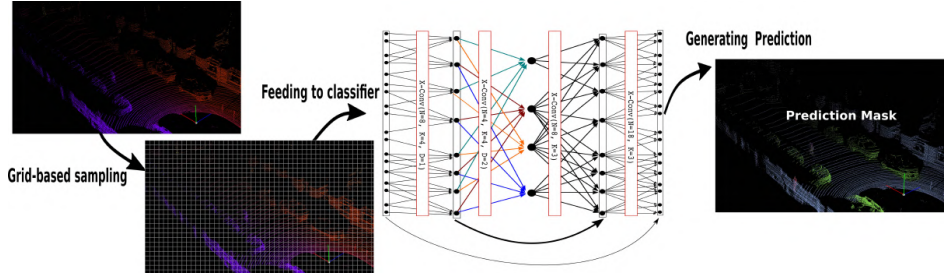


Figure 2. The proposed pipeline for semantic segmentation with PointCNN as the classifier of choice.

learning process of the deep learning kernels. In that sense, the density-adaptive sampling aims to amplify the likelihood that features from scenes with fewer points will be considered in the learning iterations. Density-adaptive sampling scheme can be achieved by using grid-based uniform sampling. This sampling method calculates the average point density ( $apd$ ) on pre-gridded 3D point cloud data. The point density in each grid will then be normalized by over-sampling points within the grid cell to make the point density equal to the value of  $apd$ . Finally, an empirical or uniform sampling technique (without replacement) is applied to the normalized-density-grid to select the points used for training the deep learning model.

### 3.3. Classification algorithm

In this study, we use PointCNN as the classifier of choice. The novelty of PointCNN in learning from point cloud data is the use of nearby points as features for a point of interest. It uses point cloud as input represented as  $(x, y, \text{ and } z)$  coordinates along with a scalar denoting the intensity value for each point. With these inputs, PointCNN enriches the features by exploiting the spatial auto-correlation of nearby points. Technically, PointCNN employs hierarchical convolution; this feature is similar to the well-known pooling layer in ConvNet. The hierarchical convolution of PointCNN aggregates information from neighborhood feature maps to fewer points by applying the X-Conv operation recursively, clusters nearby points as the feature representation of the point of interest using a  $K$ -nearest neighbor algorithm, and projects the clustered points into the local coordinate system with the point of interest being the center of the cluster. After a series of transformations based on the PointCNN pipeline using MLP coupled with X-Conv operators and higher-dimensional projections, the segmentation class for each point is acquired.

### 3.4. Evaluation metric

For a class-imbalanced data set, such as AV point cloud data where the environment class by far dominates the in-

teresting classes (such as car, pedestrian, cyclist, etc.), only a few alternative metrics will be appropriate for evaluating the segmentation task. This is the case because metrics such as the overall accuracy performance are meaningless. By just ignoring building a classifier and instead using the simple rule of assigning all points into environment class would give good results in terms of the overall accuracy. To avoid this issue, we use Mean of Per-class Accuracy ( $MPA$ ) metric as a notion of accuracy. The simultaneous comparison of true positive ( $TP$ ) rate as correct prediction and false negative ( $FN$ ) combined with false positive ( $FP$ ) rate as wrong prediction is a common practice. Hence, we will also consider the Mean Intersection over Union ( $MIOU$ ). The calculation of each metric is as follow, with  $k$  being the total number of classes and  $p_{ij}$  being the number of points of class  $i$  but classified as class  $j$ .

$$MPA = \frac{1}{k} \sum_{i=0}^{k-1} \frac{p_{ii}}{\sum_{j=0}^{k-1} p_{ij}}, \text{ and} \quad (1)$$

$$MIOU = \frac{1}{k} \sum_{i=0}^{k-1} \frac{p_{ii}}{\sum_{j=0}^{k-1} p_{ij} + p_{ji} - p_{ii}}. \quad (2)$$

## 4. Data and Experiment

We use the KITTI Vision Benchmark 3D data set [8] for our experiment, which records point clouds of driving scenes from Karlsruhe, Germany. The Kitti point cloud data set is collected using a Velodyne HDL-64E rotating 3D laser scanner, collecting data with 64 beams at 10 Hz. For our experiment purposes, we downloaded the 3D point cloud data set and labels from the Kitti website, totaling 29 GB in size. The data set contains 7481 scenes from an ego car viewpoint located in the center of the scenes. On average, each scene has 1.3 million points.

The labels provided are in the form of bounding box coordinates and class label (e.g. car, pedestrian, cyclist, etc.) for each of the boxes. We treat any points outside the bounding boxes for the pedestrian, car, and cyclist classes



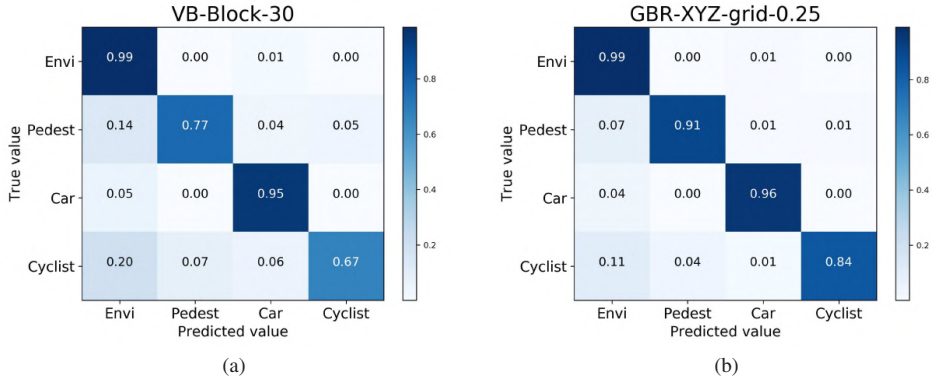


Figure 3. The accuracy performance (per-class accuracy) for the trained model using (a) the original voxel-based sampling and (b) the best-performing density-adaptive sampling.

Class	Number of Points (Percentage)	
	Training Data	Validation Data
Environment	604,128,641 (98.46%)	273,993,242 (98.44%)
Pedestrian	561,147 (0.091%)	271,529 (0.10%)
Car	8,705,256 (1.418%)	3,990,825 (1.43%)
Cyclist	181,974 (0.029%)	87,222 (0.03%)

Table 1. The distribution of class in the data set.

as environment, so most of the points belong to the environment class (around 98% of all the points). The distribution of classes is shown in Table 1. Moreover, the point density is decreasing with distance. To set the stage, we preprocessed the data by assigning the class label of the box to all points it contains, so we obtained point-wise labels. We then used 5,145 scenes for training and 2,336 scenes for validation purposes.

#### 4.1. Experimental setup

Each PointCNN model was trained using one 11 GB GeForce GTX 1080 Ti graphics card with eight-point blocks per batch, following the capacity limitation of the GPU. The Tensorflow version of PointCNN was used as the training code and environment. Unless otherwise noted, initial learning rate for all models is 0.005 with 20% learning rate decay for every 5,000 iterations and was trained for 125,000 iterations. In order to force the model to recognize objects during training for such an imbalance data set, the weighted penalty (for loss calculation) for the environment class was set to 0.1, and to 1 for other classes. For calculat-

Method	<i>MPA</i>	<i>MIoU</i>
VB Block-10	0.6329	0.3369
VB Block-20	0.6999	0.5263
VB Block-30	0.8273	0.5717
Without oversampling		
GBR-Original	0.8895	0.6418
GBU-Original-XY-0.25	0.8641	0.6179
GBU-Original-XY-1	0.8731	0.6511
XY-GRID with Oversampling		
GBU-XY-grid-0.25	0.8876	0.5840
GBU-XY-grid-1	0.8983	0.6304
GBR-XY-grid-0.25	0.9152	0.6342
GBR-XY-grid-1	0.8881	0.6504
XYZ-GRID with Oversampling		
GBU-XYZ-grid-0.25	0.9026	0.6471
GBU-XYZ-grid-1	0.9040	0.6346
GBR-XYZ-grid-0.25	<b>0.9225</b>	<b>0.6816</b>
GBR-XYZ-grid-1	0.9217	0.6509

Table 2. The performance of each of the sampling scenario.

ing the validation accuracy, the weighted penalty for all the classes was set to 1. The highest *MPA* and *MIoU* score for the validation data are then used to evaluate and compare the prediction accuracy for all models in the experiment.

#### 4.2. Sampling scenario

We test our hypothesis by training the PointCNN with different sampling methods. The benchmark is the voxel-based sampling (the original PointCNN sampling package). We also include grid-based uniform sampling and grid-based random sampling for comparison. Each of the methods sample (without replacement) 10,000 points per block

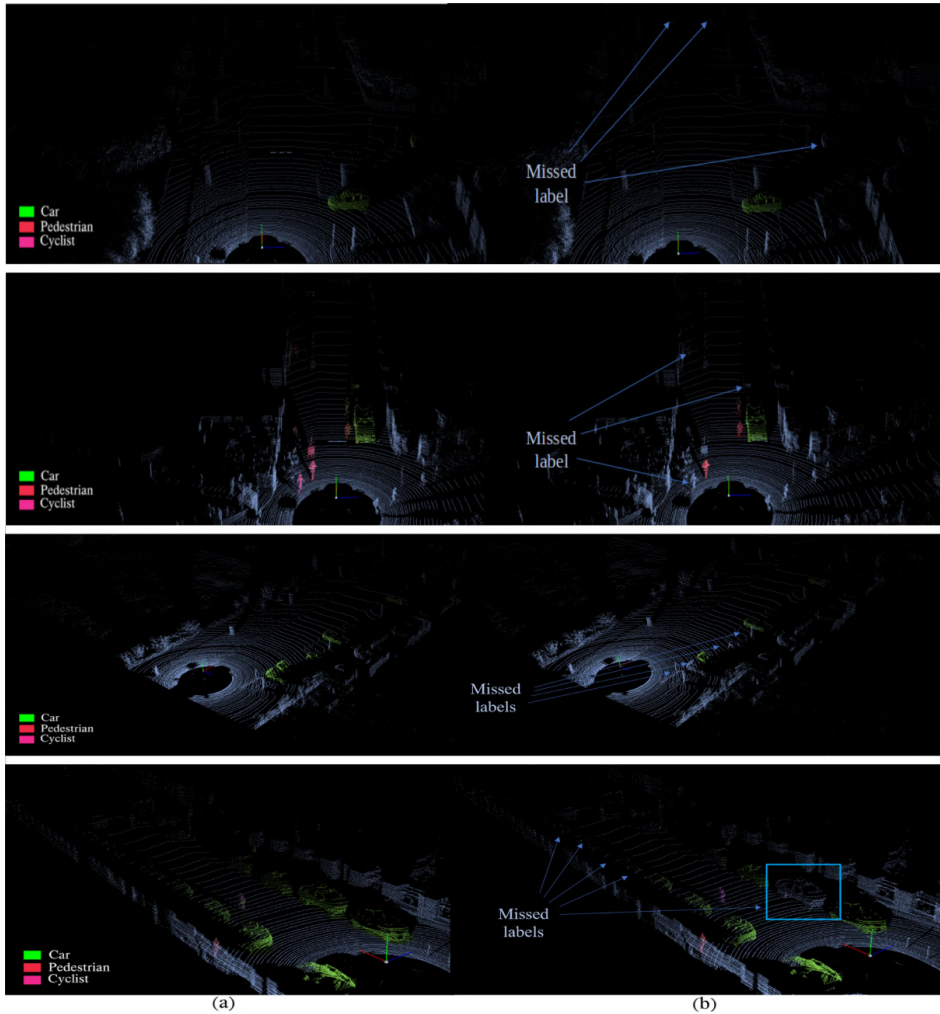


Figure 4. Point cloud visualization with (a) Prediction results, and (b) Ground truth label with missing object bounding box.

to be used as the PointCNN input. PointCNN, for every training iteration, will sample 2,048 random points to be used. It should be noted that randomly selecting 2048 points from 10,000 points per iteration will generate different point sets, hence increasing the variation of data learned by the PointCNN model.

In terms of sampling parameters, block sizes of 10, 20, and 30 coordinate values were used for the voxel-based sampling methods with the assumption that the average

point densities per block are 100, 25, and 10, respectively. The grid-based sampling methods used grid size 0.25 and 1 coordinate values. For the sake of simplicity, we refer to the voxel-based method as VB, grid-based uniform sampling as GBU, and grid-based random sampling as GBR throughout this paper. We also include versions of GBU and GBR without the rebalancing point density and call these versions GBU-Original and GBR-Original, respectively. In addition, we also compare the results for 2D and 3D grids

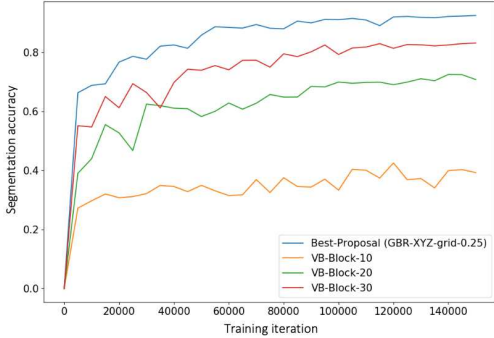


Figure 5. The *MPA* trends of the model during training using various sampling schemes.

(*XY* and *XYZ* axis) for both *GBU* and *GBR* sampling. Table 2 shows the performance of the resulting models in terms of *MPA* and *MIoU*. Fig. 5 shows the per-class accuracy using the original voxel-based method and the best-performing density-adaptive sampling method.

## 5. Discussion

In this section, we discuss our findings related to the accuracy, learning stability, and the *MIoU* performances.

### 5.1. Per-class accuracy performance

As shown in Table 2, the proposed method, using a density-adaptive sampling scheme, yields superior mean per-class accuracy performance compared to the original voxel-based sampling method. The best performance for the proposal achieves 92.25% (*GBR-XYZ-grid-0.25*) compared to 82.73% for the original sampling method (*VB-Block-30*). Table 2 shows the voxelization in the original sampling scheme does not improve the accuracy performance. It suggests that removing the partitions of the training sets while keeping the heterogeneous density, thus shifting from stratified sampling into randomized sampling based on the empirical distribution of the original data, only alleviates a part of the problem.

On the other hand, density-adaptive sampling methods, which superficially augment the representativeness of the features from all grids in the scene by means of oversampling, seems effective in increasing the generalizability of the trained model when classifying the unseen data, and works best when coupled with randomized sampling and a small grid size. The best-performing grid size for *GBR-XYZ* density-adaptive sampling class is 0.25, which is reasonable, because the smallest grid size forces the training points to be most evenly distributed covering whole scene.

Finally, Fig. 3 presents the confusion matrix showing *TP*, *TN*, *FP*, and *FN* for PointCNN trained using the original sampling method (top) and our density-adaptive method (bottom). It is clear that the proposed sampling method attains a model with better accuracy performance for all classes. The most significant increase in performance is observed for classes with fewer points (pedestrian and cyclist), thus indicating the robustness of our sampling scheme in dealing with class-imbalanced data sets.

### 5.2. Performance stability and learning rate

In Fig. 5, the accuracy of the model during training is computed against the validation data set in each training iteration. Notice that the model trained using the original voxel-based sampling method with grid size 10 (*VB-Block-10*) has the most volatile performance and the other original *VB* methods have less volatile performance in earlier stages of its learning procedure, while the density-adaptive methods have more stable performance during training. This observation indicates that the training set from density-adaptive methods makes it easier for the deep learning models to learn hidden features from the point cloud data, hence achieving more stable learning rate and faster convergence. It is therefore promising to devise an objective function and parameterize the sampling scheme so as to maximize the learning rate and overall performance of the deep learning model in this setting. We plan to derive such a schema in our future work.

### 5.3. The *MIoU* and inaccurate bounding boxes

Fig. 4 shows the comparison of our segmentation result (left) with the provided labels from the *Kitti* data set (right), where the trained model discovers more car objects than the ground truth label provides (the car highlighted in the right figure is not labeled). We also notice some inaccurate bounding box locations. Computing the *MIoU* metric penalizes the trained model for discovering unlabeled objects or locating the bounding boxes more accurately, due to the smaller intersection (appearing in the numerator of Equation 2) and larger union (in the denominator of Equation 2) between boxes from the provided labels and model prediction. In this situation, we argue that the *MIoU* cannot be used as an appropriate metric for segmentation performance. Therefore, we cannot use the *MIoU* from Table 2 to compare the performance of the sampling methods, though it is interesting to see that the *MIoU* score from the original voxel-based sampling scheme.

## 6. Conclusion

In this study, we have proposed a density-adaptive sampling method to be combined with a deep learning architecture for semantic segmentation using raw point cloud

data, such as the PointCNN. We have shown that the heterogeneous point density and the class imbalance of the point cloud from Velodyne-type LiDARs in AV applications render the voxel-based sampling method in the original PointCNN implementation ineffective for constructing a good training data set. By implementing density-adaptive sampling, i.e. by oversampling the less dense grids of the scene hence augmenting the representativeness of the feature in the less dense scenes using various parameterization scenarios, we show that the deep learning models yields superior performance in terms of classification accuracy and learning stability. Furthermore, we have shown empirically that the trained model is more robust against the class imbalance that is prevalent in real-world driving scenes.

## Acknowledgment

This research was funded in part by gifts from Bosch Corporate Research. The authors would like to thank Ji Eun Kim and Wan-Yi Lin for their valuable discussion and suggestions.

## References

- [1] Hasan Asyari Arief, Geir-Harald Strand, Håvard Tveite, and Ulf Indahl. Land cover segmentation of airborne lidar data using stochastic atrous network. *Remote Sensing*, 10(6):973, 2018. [2](#)
- [2] Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter*, 6(1):20–29, 2004. [2](#), [3](#)
- [3] R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. [1](#)
- [4] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017. [3](#)
- [5] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Niebner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. [2](#)
- [6] Bertrand Douillard, James Underwood, Noah Kuntz, Vsevolod Vlaskine, Alastair Quadros, Peter Morton, and Alon Frenkel. On the segmentation of 3d lidar point clouds. In *2011 IEEE International Conference on Robotics and Automation*, pages 2798–2805. IEEE, 2011. [3](#)
- [7] Andrew Estabrooks, Taeho Jo, and Nathalie Japkowicz. A multiple resampling method for learning from imbalanced data sets. *Computational intelligence*, 20(1):18–36, 2004. [3](#)
- [8] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. [4](#)
- [9] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012. [2](#)
- [10] Mark Harris. Ces 2018: Waiting for the \$100 lidar, Jan 2018. [1](#)
- [11] Julio Isidro, Jean-Luc Jannink, Deniz Akdemir, Jesse Poland, Nicolas Heslot, and Mark E Sorrells. Training set optimization under population structure in genomic selection. *Theoretical and applied genetics*, 128(1):145–158, 2015. [2](#), [3](#)
- [12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):8490, 2017. [3](#)
- [13] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointnet: Convolution on x-transformed points. In *Advances in Neural Information Processing Systems*, pages 820–830, 2018. [1](#), [2](#), [3](#)
- [14] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. *Computer Vision – ECCV 2018 Lecture Notes in Computer Science*, page 663678, 2018. [3](#)
- [15] Tomasz Malisiewicz and Alexei A Efros. Improving spatial support for objects via multiple segmentations. 2007. [3](#)
- [16] Anh Nguyen and Bac Le. 3d point cloud segmentation: A survey. In *2013 6th IEEE conference on robotics, automation and mechatronics (RAM)*, pages 225–230. IEEE, 2013. [3](#)
- [17] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. [1](#)
- [18] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017. [1](#), [3](#)
- [19] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. [3](#)
- [20] Brent Schwarz. Lidar: Mapping the world in 3d. *Nature Photonics*, 4(7):429, 2010. [3](#)
- [21] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointnet-cnn: 3d object proposal generation and detection from point cloud. *arXiv preprint arXiv:1812.04244*, 2018. [1](#), [3](#)
- [22] Yin Zhou and Oncel 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*, pages 4490–4499, 2018. [3](#)

---

**Paper D:**  
**SAnE: Smart Annotation and Evaluation  
Tools for Point Cloud Data**

---

**SAnE: Smart Annotation and Evaluation Tools for Point Cloud  
Data**

Hasan Asy'ari Arief<sup>1,2</sup>, Mansur Maturidi Arief<sup>2</sup>, Guilin Zhang<sup>2</sup>, Ulf Geir Indahl<sup>1</sup>, Håvard Tveite<sup>1</sup>, and Ding Zhao<sup>2</sup>

<sup>1</sup>*Faculty of Science and Technology, Norwegian University of Life Sciences, 1432 Ås, Norway*

<sup>2</sup>*Mechanical Engineering Department, Carnegie Mellon University, Pittsburgh, PA, USA*

Submitted on Computer Vision Conference on Nov 2019.



# SAnE: Smart annotation and evaluation tools for point cloud data

Hasan Asy'ari Arief  
Norwegian University of Life Sciences  
Ås, 1432, Norway  
hasanari@nmbu.no

Guilin Zhang  
Carnegie Mellon University  
5000 Forbes Avenue, Pittsburgh, PA, USA  
guilinzhang@link.cuhk.edu.cn

Håvard Tveite  
Norwegian University of Life Sciences  
Ås, 1432, Norway  
havard.tveite@nmbu.no

Mansur Arief  
Carnegie Mellon University  
5000 Forbes Avenue, Pittsburgh, PA, USA  
marief@andrew.cmu.edu

Ulf Geir Indahl  
Norwegian University of Life Sciences  
Ås, 1432, Norway  
ulf.indahl@nmbu.no

Ding Zhao  
Carnegie Mellon University  
5000 Forbes Avenue, Pittsburgh, PA, USA  
dingzhao@andrew.cmu.edu

## Abstract

Bridging the needs to provide high-quality, time-efficient, and easy-to-use annotation tools, we propose SAnE, a semi-automatic annotation tool for labelling point cloud data. While most current methods rely on multi-sensor approaches to provide bounding box annotations, we here focus on maximizing the potentials of point cloud data alone to provide high-quality point cloud labels. The contributions of this paper are threefold: (1) we propose a denoising pointwise segmentation strategy enabling one-click annotation, (2) we expand the motion model technique with our novel guided-tracking algorithm, easing the frame-to-frame annotation process, and (3) we provide an interactive yet robust open-source point cloud annotation tool, simplifying the creation of high-quality bounding box annotations. Using the KITTI dataset, we show that our approach speeds up the annotation process by a factor of 4.17 while achieving Intersection over Union (IoU) agreements of 92.02% and 82.22% with 2D bounding box (BBOX) and Bird Eye View (BEV), respectively. A more carefully executed annotation based on the same tool even achieves +8.84% higher BEV IoU agreement than the baseline annotation accuracies.

## 1. Introduction

The growing popularity of high-frequency point cloud data, scanning real-world driving scenes, fuels up a new re-

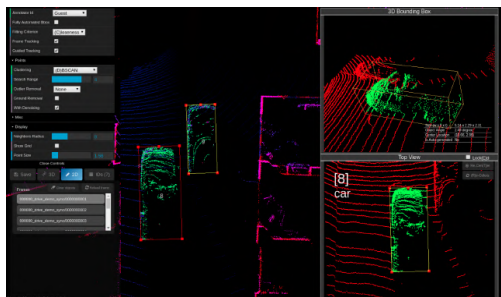


Figure 1. The interface of SAnE, a semi-automatic annotation tool based on one-click annotation scheme empowered with denoising point-wise segmentation approach and robust guided-tracking algorithm.

search stream about 3D perception systems, enriching the perception systems discussion previously centered around image analysis (from cameras) to the realm of point cloud analysis, which include point cloud classification, segmentation and object detection [6, 5]. In fact, several new large driving scene datasets containing point cloud data have recently been published by self-driving tech companies, such as ArgoVerse, Waymo, Lyft, etc [2], highlighting an increasing trend of the use and collection of the Light Detection and Ranging (LiDAR) point cloud data as the self-driving technologies are being developed and deployed in the real world.



Developing robust self-driving technologies require more than just data acquisition. Data annotations, i.e. labeling objects in the point cloud scenes, are also a necessity. However, the annotation process is usually tedious and resource-consuming while the results might be inaccurate if done manually [23]. The challenge of providing datasets with high-quality point cloud annotation include: 3D annotation complexity and human errors. Unlike annotating 2D images, drawing bounding boxes in 3D space is complicated. Annotating a single 3D instance not only requires the accurate center location, length, width, and height of the bounding box, but also requires the orientation of the object. The detailed process of 3D ground truth annotation is described in Sun RGBD [22]: it requires switching between different views to obtain accurate location and orientation. Manually providing all such details for *each* object in the scene is a complicated and tedious process. Hence, the workload of human annotators as well as the cost of providing high-quality 3D datasets grows in the complexity of the scene and dataset size. Furthermore, as the complexity of annotating 3D point clouds increases, human annotators become more prone to making mistakes. The annotation errors for 3D objects have been found to be significantly higher compared to those of 2D instances. The erroneous labels of the KITTI 3D object detection dataset (such as objects with missing labels or objects having incorrect bounding box locations) [1] is an example of the practical challenge of providing high-quality ground truth annotations.

To tackle these challenges, researchers have proposed semi-automated annotation tools in recent years, including Polygonrnn++, 3D-bat, and Latte. Polygonrnn++ [3] is an improved version of the earlier work Polygonrnn method [7] that employs semi-automatic algorithm to detect the vertices of polygons that contain the objects, which has helped reduce the annotation effort and time required to annotate image data. The 3D-bat application [28] proposes a 3D annotation toolbox which is equipped with various features, highlighting the usability and annotation efficiency but undermining the automatic functionalities. Latte [23] proposes an annotation tool for 3D point cloud annotation with one-click annotation (based on the DBSCAN algorithm [9]) and frame tracking (based on the Kalman Filter [25]), which reduces the complexity of the annotation task, attaining improved efficiency and reasonable accuracy performance.

We propose the Smart Annotation and Evaluation (SAnE) tool for efficient point cloud annotation, which adopts the Latte interactive tool and implements a 3D point cloud deep learning model and a guided tracking algorithm to boost the performance. SAnE enables human annotators to annotate both accurately and efficiently by implementing:

1. **Denoising pointwise segmentation**, a novel nearly noise free semantic segmentation strategy, enabling a robust one click annotation technique. In addition,

the denoising technique eliminates the need to have a workable ground-removal algorithm that is a requirement in Latte’s proposal [23].

2. **Guided Tracking**, based on a motion model that provides baseline tracking throughout all the frames, refined using the heuristics approaches (greedy search and backtracking algorithm). Hence, only minimal adjustment (if any) is required from the human annotator to track sequential point cloud scenes.
3. **Improved annotation flow**, enhanced with both AI-based functionalities (one-click annotation, guided-tracking, and fully automated bounding box proposals) and User Interface (UI) based improvements, such as keyboard-only annotations, multi-user environments, user-adjusted parameters, and 3D bounding box estimation.

Our experiments using the KITTI dataset [11] highlight that with  $4.17\times$  less annotation time, SAnE can achieve IoU agreements of 92.02% and 82.22% for 2D bounding boxes (BBOX) and Bird Eye View (BEV), respectively. A more carefully executed annotation even achieves 8.84% and 7.47% higher IoU agreements than the baseline annotation accuracies for objects in front of the ego vehicle and objects in the whole point cloud area, respectively.

The rest of our work is organized as follows. In Section 2, we review point cloud annotation algorithms available in the literature. In Section 3, we describe the key machineries that we have either designed or adopted from earlier work when developing the SAnE. Experiment results and discussions are provided in Section 4. Finally, conclusions are provided in Section 5.

## 2. Related Work

### 2.1. Existing point cloud dataset

Modern deep learning models are data intensive, for which reason many existing works have contributed to produce public datasets for research in autonomous driving. The KITTI 3D object detection dataset is popular for current autonomous driving projects. It contains 15,000 frames of road scenes with corresponding images, point clouds, and ground truth annotations. The Apollo Space dataset contains 12,360 frames of annotated sequential point clouds, collected from complex traffic conditions. In addition to these real-world datasets, several researchers have proposed synthetic datasets for their ease of generation and annotation. The PreSIL dataset is a synthetic dataset generated from the commercial video game GTA V, which contains over 50,000 frames with point-wise segmentation and accurate bounding box annotations for all vehicles and people. However, both the existing real-world and synthetic datasets



have drawbacks. Real-world datasets are limited in size and annotation accuracy compared to synthetic datasets, but synthetic datasets are not domain transferable at present [4]. Thus, it is still a necessity for people to customize their own datasets rather than just using the same public datasets for all specialized tasks.

## 2.2. Point cloud semantic segmentation and object detection

LiDAR based 3D object detection is essential for autonomous driving because point clouds collected from LiDAR contain rich 3D information, including location, dimension, and orientation. However, compared to 2D images, 3D point clouds appear irregular and unordered. Therefore, it is hard to leverage the traditional image analysis techniques to perform general recognition tasks on point clouds, such as semantic segmentation. In early works, people manually transformed irregular point clouds into regular 3D voxel grids [27]. Such a transformation successfully represents irregular 3D data but is constrained by the data sparsity and the shape of the objects. More recent works operate directly on 3D point clouds. **PointNet** [19] directly consumes point cloud data and provides a unified approach to general 3D recognition tasks. **PointCNN** [16] is a generalized CNN framework that includes feature learning from point clouds to achieve point cloud segmentation. We leverage and improve this method using our proposed denoising pointwise segmentation method, which boosts the accuracy and efficiency of the SAnE.

Other than the works for semantic segmentation, there are also works that achieve end-to-end object detection tasks on point clouds. Many works have tried to leverage mature 2D detectors for generating 2D proposals and perform bounding box regression in 3D space, such as the **Frustum Pointnet** [18]. Inspired by the 2D region proposal network like **F-Convnet** [24], **AVOD** [15] proposes a novel architecture that contains a feature extractor and sub-networks for 3D proposal generation and regression. To further eliminate the influences of 2D data limitations, recent work like **PointRCNN** [21] generates high quality 3D proposals directly from the point clouds by segmentation, and perform accurate refinement to generate better bounding box predictions. Comparing the segmentation frameworks, these methods provide efficient localization of vehicles in 3D spaces.

## 2.3. Annotation tools for point cloud

With the development of LiDAR-based detection methods and the rise of demands for 3D datasets in recent years, some works have contributed to make annotation tools that aim at improving the efficiency of creating useful datasets. **PolygonRNN** and **PolygonRNN++** propose a semi-automatic approach to polygon region prediction

speeding the image annotation process by a factor of 7. Apart from the success of annotation tools on 2D image data, 3D annotation tools have also improved. **3D-Bat** [28] and **Latte** [23] provide outstanding works with well-developed point cloud annotation tools integrated with semi-automatic functionalities deployed as web-based applications. **Latte** realizes one-click annotation that significantly reduces complex annotation works into a simple click operation. It also proposes frame-to-frame object tracking that further boosts the annotation efficiency of sequential data frames. However, **Latte** is still using 2D detectors (**MaskRCNN** [14]) on images, combined with points projected from 3D point clouds, for label prediction. This approach is constrained by the camera views, image qualities, and tends to mislabel closely located objects. To address this problem, we propose **denoising pointwise segmentation** to improve the prediction accuracy and the simplicity of one click annotation technique.

## 3. SAnE Annotation

Creating an open source, yet high-quality, AI-assisted point cloud annotation tool has been the goal for this project. In this section, we emphasize three key contributions of our work, namely: (1) The denoising pointwise segmentation strategy, enabling accurate one click annotation, (2) The guided-tracking algorithm, easing the frame-to-frame annotation process, and (3) An interactive yet robust point cloud annotation tool that simplifies the creation of high-quality 3D annotation datasets.

### 3.1. Denoising pointwise segmentation

Deep learning based pointwise segmentation techniques, such as **PointNet**[19], **PointNet++** [20], and **PointCNN** [16], are based on the cross-entropy loss function and the back-propagation algorithm in their kernel optimization processes. These techniques, even though they tend to provide high accuracy prediction results [20], are prone to provide a noisy segmentation near the object boundaries, see Fig. 2a and 2b. This is because the particular loss function penalizes all wrong predictions, ignoring the location of the errors, see Eq. 1.

$$L = - \sum_i^C t_i \log(f(s)_i) \quad (1)$$

A noisy pointwise segmentation complicates the annotation process, such as the **Latte** one-click-annotation technique (see [23]). This technique uses the Density-based Spatial Clustering of Applications with Noise (DBSCAN) algorithm [9] to isolate point-clusters and generate a bounding box for the selected cluster. A noisy cluster may result on a wrong bounding box shape and an inaccurate prediction of the box direction, see Fig. 2c and 2d.

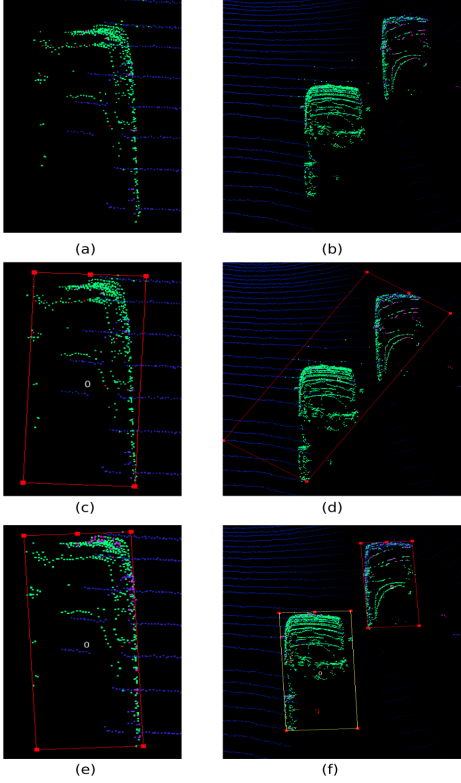


Figure 2. The impact of the denoising pointwise segmentation on estimating bounding box proposals using one-click annotation technique: (a-b) noisy boundaries pointwise segmentation, (c-d) bounding box estimation using a standard one-click annotation technique on noisy point cloud segmentation, and (e-f) bounding box estimation using the denoising pointwise segmentation technique.

The proposed denoising technique aims to provide a noise free segmentation, enabling the one-click annotation technique. In addition, the technique also does ground removal that is required for the one-click annotation process.

The main idea of the denoising technique is to force the deep learning model to avoid wrong predictions near object boundaries during the kernel optimization (training process) by increased penalization. As shown in Fig. 2e and 2f, the same one-click annotation technique provides the best bounding box proposals generated from nearly noise free point cloud segmentation data.

The denoising technique is implemented as a set of

penalty values to the prediction results during the loss calculation. Therefore, the technique can be implemented both for the cross-entropy loss function as well as other loss functions [5, 17].

---

#### Algorithm 1 Denoising weight penalty

---

- 1:  $W_p, nO, nW, w, zO$  ▷ Weighted penalties, noise offset, noise weight, normal weight, and distance offset to the ground.
  - 2: **for**  $obj$  **in**  $allObjects$  **do**
  - 3:  $I_{in} \leftarrow obj.pointIndicesInsideBox()$
  - 4:  $obj.dimensions \leftarrow obj.dimensions + nO$
  - 5:  $I_{out} \leftarrow obj.pointIndicesInsideBox()$
  - 6:  $W_p[I_{out}] \leftarrow nW$
  - 7:  $zMin \leftarrow \min(P[I_{in}, Z_{AXIS}]) + zO$
  - 8:  $W_p[I_{in}, \text{and}(W_p[:, Z_{AXIS}] > zMin)] \leftarrow w$
  - 9: **end for**
- 

Given a set of weighted penalties  $W_p$  in the point cloud data ( $P$ ), the denoising penalties are described in Alg. 1. For all objects in a frame, the denoising-weight-penalty calculates all point indices inside the bounding box (Line 3), and recalculate all point indices inside the enlarged ( $+nO$ ) bounding box (Line 5). Lines 6-8 assigns the noise penalty ( $nW$ ) for all boundary locations and ground object areas, forcing the loss function to give higher penalties around those areas.

### 3.2. Guided tracking algorithm

Annotating sequential frames of point cloud data can be time consuming but it can be speeded up using a frame-to-frame tracking algorithm. For example, the Kalman filtering approach [25] is adopted by Latte [23] to track the bounding box center of an object, and provides a speed-up by a factor of 4.74 compared to manually creating bounding boxes for each new frame. In fact, a tracking algorithm does not only speed-up the annotation process, but also gives better annotation agreement and accuracy of the tracked bounding boxes [23].

Formally, Latte’s implementation of the Kalman filter defines  $x_k = [p_x, p_y, v_x, v_y, a_x, a_y]^T$  as a state vector at frame  $k$ , where  $p_x, p_y$  denotes the coordinates of the center of the bounding box, while  $v_x, v_y$  and  $a_x, a_y$  denote the bounding box center velocity and acceleration along the axis  $(x, y)$ . The Kalman filter is used to estimate  $\hat{x}_{k|k}$  by weighting the observation  $z_k = [p_{x,k}, p_{y,k}]^T$  taken from the annotator adjustment of the proposed bounding box at frame  $k$ . See [23] for an in depth implementation of the Kalman filter.

The predicted state  $\hat{x}_{k+1|k}$ , on the other hand, is obtained as the motion model implementation between the estimated state  $\hat{x}_{k|k}$  and the state transition model  $F \in R^{6 \times 6}$ . It is used as the new bounding box location for the next frame

( $k + 1$ ) and is formally given by:

$$\hat{x}_{k+1|k} = F\hat{x}_{k|k} \quad (2)$$

where

$$F = \begin{bmatrix} 1 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t^2 & 0 \\ 0 & 1 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t^2 \\ 0 & 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The new bounding box location  $\hat{p}_{x,k+1}, \hat{p}_{y,k+1} \in \hat{x}_{k+1|k}$  is a representation of physical flow of an object. However,  $\hat{x}_{k+1|k}$  comes from the relative motion of an object with respect to the ego vehicle and ignores the independent assumption of those objects. Therefore, errors are expected for the prediction, and refinement of the annotations is necessary. In addition, the initial velocity  $v_{x,y}$  and acceleration  $a_{x,y}$  are set to zero, therefore the first few consecutive frames need to be refined and/or reannotated until the estimated velocity  $\hat{v}_{x,y}$  and acceleration  $\hat{a}_{x,y}$  can give more accurate predictions. Refining and reannotating bounding boxes is a time consuming effort, especially when more objects are found in the frames, which is very common in the urban areas.

The motion model technique is extended in this work by using the guided-tracking algorithm. The objective of this algorithm is to reduce the effort to refine and/or reannotate the tracked objects. The idea is that some initial bounding box location ( $\hat{p}_{x,k+1}, \hat{p}_{y,k+1}$ ) can be regressed to fit the closest point cloud cluster. The hypothesis is that each cluster belongs to a different object, therefore regressing the bounding box to fit the closest cluster will effortlessly refine the bounding box location given by the motion model.

The guided-tracking algorithm comes with three modules, namely: (1) A greedy search, regressing the bounding boxes to their closest corresponding clusters, (2) Backtracking, preventing overlapping between multiple bounding boxes, and (3) Tracking refinement, optimizing the final bounding box location based on the closest point cluster.

**Greedy Search.** The greedy search algorithm, explained in Alg. 2, works by moving the predicted bounding box around its initial location. It uses the bounding box center location and the point cloud data denote as  $b$  and  $P$ , respectively. A dictionary ( $s$ ), containing bounding box movements, is also used. For each new location of  $b$ , the number of points inside the bounding box is counted ( $numPoints$ ) and the distances between points inside the box and their closest edges, are also calculated and summarized ( $avgDist$ ). The new location with maximum  $numPoints$  and minimum  $avgDist$  is assumed to be the best possible location for the current iteration. The search

---

### Algorithm 2 Greedy search algorithm

---

```

1: procedure GREEDYSEARCH( $s, b, P$ )
2:    $b_u \leftarrow b$   $\triangleright b$  contains bounding box information.  $b.c$ 
   denotes the bounding box center locations along  $x$  and
    $y$  axis.
3:    $b.c.x \leftarrow b.c.x - s_p$ 
4:    $b.c.y \leftarrow b.c.y - s_p$ 
5:    $stride \leftarrow s_p * 2 / s_{num}$   $\triangleright s_{num}$  denotes number of
   strides.
6:   for  $x = 0; x \leq s_{num}; x++$  do
7:     for  $y = 0; y \leq s_{num}; y++$  do
8:        $x_t \leftarrow b_u.c.x \leftarrow b.c.x + x * stride$ 
9:        $y_t \leftarrow b_u.c.y \leftarrow b.c.y + y * stride$ 
10:       $Ps_{in} \leftarrow b_u.contains(P)$ 
11:       $s_{dict}[x_t, y_t][0] \leftarrow (len(Ps_{in}))$ 
12:       $s_{dict}[x_t, y_t][1] \leftarrow (b_u)$ 
13:       $s_{dict}[x_t, y_t][2] \leftarrow Dist(Ps_{in}, b_u.edges)$ 
14:       $s_{dict}[x_t, y_t][2] \leftarrow Avg(s_{dict}[x_t, y_t][2])$ 
15:    end for
16:  end for
17:   $maxIndices \leftarrow argMaxes(s_{dict}[...][0])$ 
18:  if  $len(maxIndices) > 1$  then
19:     $minIdx \leftarrow argMin(s_{dict}[maxIndices][2])$ 
20:     $numPoints, b_{now} \leftarrow s_{dict}[minIdx][0, 1]$ 
21:  else
22:     $numPoints, b_{now} \leftarrow s_{dict}[maxIndices][0, 1]$ 
23:  end if
24:  if  $numPoints > s_{numPoints}$  then  $\triangleright s_{numPoints}$ 
   is a global variable, denoting the highest  $numPoints$ 
   from previous iterations.
25:     $s_{numPoints} \leftarrow numPoints$ 
26:     $s_{dict} \leftarrow \{\}$ 
27:    Return GREEDYSEARCH( $s_{dict}, b_{now}, P$ )
28:  end if
29:  Return  $s, b, P$   $\triangleright b.c[x, y]$  is the optimal bounding
   box location.
30: end procedure

```

---

is implemented using a recursive function, and an iteration that doesn't change the value of  $numPoints$  ends the search process.

Alg. 2 relies on the value of padding size ( $s_p$ ) to move the bounding box to the best possible location with the highest  $numPoints$ . A higher  $s_p$  means a higher possibility for the bounding box to overlap with other bounding boxes, see Fig. 3a. Therefore, a backtracking algorithm is included to alleviate this problem.

**Backtracking.** The backtracking algorithm works by re-tracking (move-back) the overlapped bounding box locations ( $Bs$ ) to the best possible locations where the boxes do not overlap anymore. The first step is that for each bounding box, the distance between the initial ( $\hat{p}_{x,k+1}, \hat{p}_{y,k+1}$ )

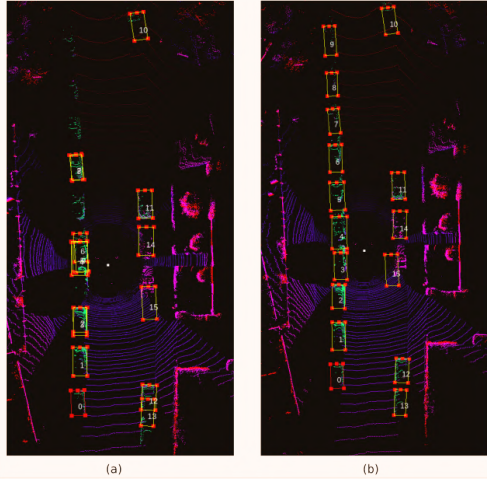


Figure 3. The backtracking algorithm for fixing the greedy search overlapping problem: (a) before and (b) after the backtracking algorithm.

and the updated bounding box location from Alg. 2 is calculated. Then, overlapping boxes are re-tracked based on those distances. The overlapping boxes with the longest distances are moved until the particular boxes are not overlapping anymore, see Alg. 3 for a more rigid explanation.

The backtracking algorithm can separate overlapping objects effectively, see Fig. 3b. However, the proposal for each updated bounding box location might not be optimal. This is because the algorithm does not optimize the  $numPoints$  and  $avgDist$  of the moved boxes. Therefore, the tracking refinement step is required to both optimize the bounding box locations while preventing overlap.

**Tracking Refinement.** The tracking refinement is a reimplementation of Alg. 2 with much smaller  $s_p$ . The intuition is that after the first greedy-search and backtracking processes, the proposed bounding box locations are already closed to the optimal solutions. Therefore, only a small change is required to find the best fitted location.

In addition to the smaller  $s_p$ , the refinement algorithm also validates collisions between bounding box proposals when striding to the new bounding box locations (Line 8-9, Algorithm 2) to assure that only new locations with non-overlapping bounding boxes are proposed.

### 3.3. AI-assisted annotation tool

Based on an effective denoising technique and the robust frame-to-frame tracking algorithm, we offer an open source semi-automatic annotation tool for 3D point cloud

---

#### Algorithm 3 Backtracking algorithm

---

```

1: procedure BACKTRACKING( $Bs$ )
2:    $isOverlapExist \leftarrow true$ 
3:   while  $isOverlapExist$  do
4:      $isOverlapExist \leftarrow false$ 
5:     for  $i = 0; i < len(Bs); i++$  do
6:       for  $j = 0; j < len(Bs); j++$  do
7:          $idx \leftarrow -1$ 
8:         if  $i \neq j$  then
9:           while  $idx \neq 0 \wedge Bs[i].Overlap(Bs[j])$  do
10:             $isOverlapExist \leftarrow true$ 
11:             $iDist \leftarrow Bs[i].centerDist()$ 
12:             $jDist \leftarrow Bs[j].centerDist()$ 
13:            if  $iDist \geq jDist$  then
14:               $idx \leftarrow Bs[i].updateIdx()$ 
15:            else
16:               $idx \leftarrow Bs[j].updateIdx()$ 
17:            end if
18:          end while
19:          end if
20:        end for
21:      end for
22:    end while
23:  end procedure

```

---

data. Several easy-to-use yet powerful features are embedded, such as fully automated bounding box generation, one click annotation, frame-to-frame tracking, and many other non-AI functionalities.

**Fully automated bounding generation.** This feature is used to automatically generate bounding boxes for a frame given a set of point cloud dataset, and is implemented in a three step process. First, PointCNN [16] with the denoising technique, followed by XCRF refinement [4] is used to provide pointwise segmentation of the input data. Then, the DBSCAN algorithm [9] is used to separate the segmented points into several point clusters. Finally, for each point cluster, the L-shape fitting algorithm [26] with its fitting criterion is used to estimate the bounding box shape and direction.

**One click annotation.** The one click annotation is adopted from Latte’s implementation [23]. The main improvement is that the denoising technique is used to replace the ground removal algorithm by inducing enhanced penalties around the ground areas. Additionally, region growing step based on a Nearest Neighbor (NN) search is used as a replacement for the Latte’s DBSCAN implementation. This is because this DBSCAN implementation is computationally slow (each click requires around 5-10 seconds to process). We hypothesize that the segmented point clusters are separated from each other by some distance, therefore, using NN search with a predefined search radius works faster

for estimating point clusters while giving similar (or better) region proposals compared to the DBSCAN implementation.

**Frame to frame tracking.** The frame to frame tracking implemented in the annotation tool follows the description from Subsection 3.2 including both the motion model technique and the guided tracking. The annotators can choose which tracking algorithm they actually want to use.

Complementing the AI functionalities, SAnE is also equipped with several useful features, including side-view refinement, height estimation, keyboard-only annotation, object recoloring, and more. The side-view refinement is used to simplify the bounding box refinement in locating, isolating, and magnifying the selected object. The height estimation is used to estimate object height based on the maximum and minimum point inside the bounding box, normalized with the RANSAC algorithm [10]. Moreover, keyboard-only annotation is used for annotation using only the keyboard by maximizing the use of predefined hotkeys while the object recoloring is used for contrasting the color of point inside and outside a selected bounding box.

## 4. Experiment results and findings

### 4.1. Experimental setup

We evaluate our approach on the KITTI tracking dataset [11], and use the training data with their labels for our experiments. The dataset contains 20 scenes and 8 object categories, including car, van, truck, pedestrian, cyclist, siter, tram, and miscellaneous. 3D Velodyne point cloud data with their colored images along with GPS/IMU data and 3D object tracklet labels are included in the dataset. Due to the limitations of the KITTI labels, i.e. inaccurate bounding boxes [23, 1] and only objects in front of ego vehicle being annotated [11], we also used an expert annotator to provide high quality Ground Truth labelling (GT). For the rest of this paper, we treat this labelling (GT) as the actual ground truth. It should be noted that the IoU agreement between GT and KITTI labels is 72.65%.

For the experiment, we selected 7 scenes and used the first 10 frames per scene to do the annotations. We then conducted the experiments by asking human annotators to annotate objects on those frames by using a manual annotation tool and the SAnE separately. Due to limited resources, we only used a handful of human annotators. Four people used the baseline annotation (each of them annotating 25% of the data), and two people, called Annot1 and Annot2, annotated the whole dataset using the full features of SAnE. It should be noted that we do not compare our results with the Latte annotation tools [23] because we can not get reasonable results with their ground removal algorithm.

In our annotation tool, we used PointCNN [16] as the pointwise segmentation architecture. It was trained by us-

	Per frame		Per object	
	Time(s)	Speed-up( $\times$ )	Time(s)	$\#Ops$
Baseline	201.22	1.00	5.25	34.09
Annot1	226.32	0.89	7.42	28.97
Annot2	<b>48.30</b>	<b>4.17</b>	<b>3.87</b>	<b>19.53</b>

Table 1. Annotation times with speed-up ( $\times$ ), number of clicks ( $\#Ops$ ), and actual time in seconds (s) are presented.

ing the density adaptive sampling method [1] and weighted using the proposed denoising technique. The DBSCAN algorithm and region growing method were used for point level clustering, and the proposed guided tracking algorithm with the motion model was implemented for simplifying the frame to frame tracking processes.

### 4.2. Metrics

Intersection over Union (IoU) was used for quantifying the bounding box accuracies. The IoU agreement can be seen as the proportional overlap between the intersection areas of the bounding box proposal and its corresponding ground truth data, and the combination (union) of those areas. We used the KITTI evaluation script to calculate the accuracy values [12].

The annotation times were also calculated to demonstrate the speed-up of the annotation process provided by our annotation tool. The annotation times are represented as the number of clicks called  $\#Ops$  (number of operations) and speed-up values represented as the number of times ( $\times$ ).

### 4.3. Results and findings

Table 1 shows that the fastest per frame annotation time occurred in about 48.30s i.e. 4.17 times faster than the baseline annotation (201.22s). Moreover, 3.87s with 19.53  $\#Ops$  and 7.42s with 28.97  $\#Ops$  are required to annotate an object using SAnE by Annot2 and Annot1, respectively, while using manual annotation, the annotator required 5.25s with 34.09  $\#Ops$ .

In term of accuracies, Table 2 in the  $\approx 90$ -degree view shows that the Annot2 achieves the highest BBOX IoU agreement 92.02%. It is 2.53% and 19.37% higher than the Baseline and KITTI IoU agreement with the GT, respectively. For BEV accuracies in the 360-degree view, on the other hand, Annot1 achieves the highest IoU agreement, 84.57%, 7.47% higher IoU agreement than the Baseline accuracy.

As can be seen in Table 2, relying solely on point cloud data to generate bounding box annotations leads to some problems, especially when using IoU as the accuracy metric. The IoU accuracies for the baseline and our annotation tool (Annot1) are only 76.35% and 85.19%, respectively. The IoU improvement of 8.84% is good but the IoU accuracies can be considered low for a semi-automatic technique.



	IoU value		vs Baseline		vs KITTI
	BBOX	BEV	BBOX	BEV	BBOX
Objects in $\approx 90$ -degree view					
GT	100.00	100.00	-	-	27.35
Baseline	89.49	76.35	0.00	0.00	16.84
Annot1	90.87	<b>85.19</b>	1.38	<b>8.84</b>	18.22
Annot2	<b>92.02</b>	82.22	<b>2.53</b>	5.87	<b>19.37</b>
Objects in 360-degree view					
Baseline	-	77.10	-	0.00	-
Annot1	-	<b>84.57</b>	-	<b>7.47</b>	-
Annot2	-	79.57	-	2.47	-

Table 2. Bounding box accuracies for objects in front of the ego vehicle and objects in the whole area of the point cloud using IoU agreement between annotated bounding boxes and GT. BBOX denotes the accuracies for bounding boxes projected in the image while BEV (Bird Eye View) denotes the accuracies for bounding box from the top view of point cloud scene.

\*The IoU agreement between KITTI labels and GT labels is 72.65%.

This is because the IoU is a very sensitive metric and perfect overlap (IoU=1.0) between two bounding boxes on the point cloud scene is almost impossible, see Fig. 4a. Moreover, objects in the point cloud scene can be represented by only a few points, see Fig. 4b. Therefore, a tightly fitted bounding box is constrained by the annotators subjective preferences.

#### 4.4. Limitations

Based on a straight forward weighted penalty, the denoising technique is easy to use and offers powerful guidance for bounding box generation, especially using the one click annotation algorithm [23]. However, as the penalties emphasize in the object boundaries, other areas far from the boundaries suffer with bad prediction results. This is understandable and even desirable for annotation tools, but for fully automated pointwise segmentation, this approach yields a lower prediction accuracy.

In addition to the segmentation accuracy problem, the one click annotation is also sensitive to point-density distribution and object shape representations. The L-shape fitting algorithms [26] are good for generating high quality bounding boxes when the annotated objects are in perfect L-shape forms. However, the L-shape is not the only shape of objects appearing in point cloud scenes. I-shape, U-Shape and even dot-shape are other typical shapes, and the fitting algorithm does not really work on all of these shapes.

The guided tracking algorithm is relatively slow compared to the motion model implementation, especially when the tracking refinement step is enforced. This is a weakness when the algorithm is implemented in fully automated scenarios. In addition, the algorithm depends on the point den-

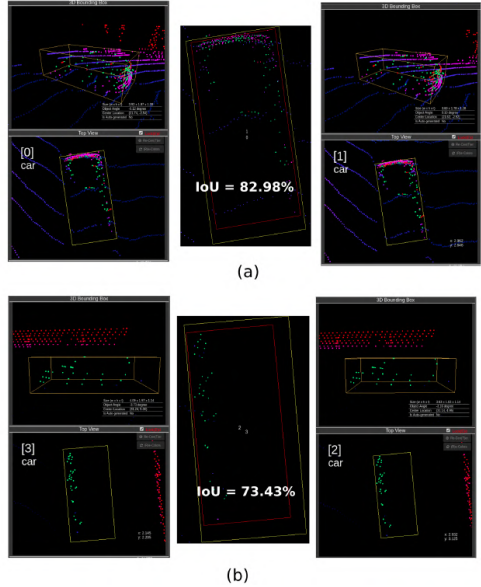


Figure 4. Problems on drawing high accuracy bounding boxes (IoU=1.0) on point cloud scene. Annotator subjective preference plays an important role in judging the tightness and correctly fitted bounding box proposals.

sity of an object. It should be noted that objects represented by high-density point-cloud-clusters are easy to track while objects represented by only few points are harder. More work, such as scene completion [8] and point cloud generation [13], is needed to overcome these problems.

#### 5. Conclusions

In this work we have introduced SANe, a robust semi-automatic annotation tool to simplify the creation of high accuracy bounding box annotation for point cloud data. The main contributions of our research are threefold. Firstly, we have proposed a denoising point-wise segmentation strategy that can provide a nearly noise free point level classifications enabling one click annotation. Secondly, we have developed a novel guided tracking algorithm, enhancing the motion model tracking using the combination of greedy search and backtracking algorithm, easing the frame-to-frame annotation processes. Finally, we provide an open-source and easy-to-use annotation tool combining AI-based functionalities, such as fully automated bounding box proposals (and one-click annotation), frame-to-frame tracking, and UI-based enhancements, i.e side-view refinement, height estimation, keyboard-only annotation, object recol-

oring, and more.

Experiments were carried out on the KITTI dataset and the results show that SAnE can speed up the annotation process by a factor of 4.17 while achieving higher accuracy than the manual annotation process. Our proposal achieves IoU agreements of 92.02% for BBOX and 82.22% for BEV. Moreover, with a more careful annotation process, the BEV IoU agreement of SAnE can reach 85.19% that is 8.84% better than the baseline accuracy. Further improvement may be achieved by combining the scene completion and point cloud generation algorithms, alleviating the limitations of point cloud data on representing complete object structure, reducing the subjectivity of human annotation preference.

## Acknowledgment

This research was funded in part by gifts from Bosch Corporate Research. The authors would like to thank Ji Eun Kim and Wan-Yi Lin for their valuable discussion and suggestions.

## References

- [1] Hasan A Arief, Mansur Arief, Manoj Bhat, Ulf Indahl, Håvard Tveite, and Ding Zhao. Density-adaptive sampling for heterogeneous point cloud object segmentation in autonomous vehicle applications. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 26–33, 2019. 2, 7
- [2] Sam Abuelsamid. Argo AI And Waymo Release Automated Driving Data Sets, Jun 2019. Retrieved from <https://www.forbes.com/sites/samabuelsamid/2019/06/19/argo-ai-and-waymo-release-automated-driving-data-sets/>. 1
- [3] David Acuna, Huan Ling, Amlan Kar, and Sanja Fidler. Efficient interactive annotation of segmentation datasets with Polygon-RNN++. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 859–868, 2018. 2
- [4] Hasan Asyari Arief, Ulf Geir Indahl, Geir-Harald Strand, and Håvard Tveite. Addressing overfitting on point cloud classification using Atrous XCRF. *ISPRS Journal of Photogrammetry and Remote Sensing*, 155:90–101, 2019. 3, 6
- [5] Hasan Asyari Arief, Geir-Harald Strand, Håvard Tveite, and Ulf Indahl. Land cover segmentation of airborne lidar data using stochastic atrous network. *Remote Sensing*, 10(6):973, 2018. 1, 4
- [6] Eduardo Arnold, Omar Y Al-Jarrah, Mehrdad Dianati, Saber Fallah, David Oxtoby, and Alex Mouzakitis. A survey on 3d object detection methods for autonomous driving applications. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–14, 2019. 1
- [7] Lluís Castrejon, Kaustav Kundu, Raquel Urtasun, and Sanja Fidler. Annotating object instances with a Polygon-RNN. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5230–5238, 2017. 2
- [8] Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2018. 8
- [9] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996. 2, 3, 6
- [10] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 7
- [11] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 2, 7
- [12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. pages 3354–3361, 2012. 7
- [13] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. pages 216–224, 2018. 8
- [14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 3
- [15] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018. 3
- [16] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. PointCNN: Convolution on X-transformed points. In *Advances in Neural Information Processing Systems*, pages 820–830. 3, 6, 7
- [17] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 4
- [18] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum PointNets for 3D Object Detection from RGB-D Data. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. 3
- [19] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660. 3
- [20] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108. 3
- [21] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019. 3
- [22] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. SUN RGB-D: A RGB-D Scene Understanding Benchmark

- Suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015. 2
- [23] Bernie Wang, Virginia Wu, Bichen Wu, and Kurt Keutzer. Latte: Accelerating lidar point cloud annotation via sensor fusion, one-click annotation, and tracking. *arXiv preprint arXiv:1904.09085*, 2019. 2, 3, 4, 6, 7, 8
- [24] Zhixin Wang and Kui Jia. Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. *arXiv preprint arXiv:1903.01864*, 2019. 3
- [25] Greg Welch, Gary Bishop, et al. An introduction to the kalman filter. 1995. 2, 4
- [26] Xiao Zhang, Wenda Xu, Chiyu Dong, and John M Dolan. Efficient l-shape fitting for vehicle detection using laser scanners. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 54–59. IEEE, 2017. 6, 8
- [27] Yin Zhou and Oncel 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*, pages 4490–4499, 2018. 3
- [28] Walter Zimmer, Akshay Rangesh, and Mohan Trivedi. 3d bat: A semi-automatic, web-based 3d annotation toolbox for full-surround, multi-modal data streams. *arXiv preprint arXiv:1905.00525*, 2019. 2, 3





ISBN: 978-82-575-1688-8

ISSN: 1894-6402



Norwegian University  
of Life Sciences

Postboks 5003  
NO-1432 Ås, Norway  
+47 67 23 00 00  
[www.nmbu.no](http://www.nmbu.no)