DGK Veröffentlichungen der DGK

Ausschuss Geodäsie der Bayerischen Akademie der Wissenschaften

Reihe C | Dissertationen | Heft Nr. 911

Michael Kölle

Forming a Hybrid Intelligence System by Combining Active Learning and Paid Crowdsourcing for Semantic 3D Point Cloud Segmentation

München 2023

Bayerische Akademie der Wissenschaften

ISSN 0065-5325 ISBN 978-3-7696-5323-6

Diese Arbeit ist gleichzeitig veröffentlicht in:
OPUS – Online Publikationen der Universität Stuttgart
http://dx.doi.org/10.18419/opus-13476, Stuttgart 2023

DGK Veröffentlichungen der DGK

Ausschuss Geodäsie der Bayerischen Akademie der Wissenschaften

Reihe C Dissertationen Heft Nr. 911

Forming a Hybrid Intelligence System by Combining Active Learning and Paid Crowdsourcing for Semantic 3D Point Cloud Segmentation

A thesis

accepted by the Faculty of Aerospace Engineering and Geodesy

of the University of Stuttgart

in partial fulfillment of the requirements for the degree of

Doctor of Engineering Sciences (Dr.-Ing.)

by

Michael Kölle, M.Sc.

born in Geislingen an der Steige

München 2023

Bayerische Akademie der Wissenschaften

ISSN 0065-5325 ISBN 978-3-7696-5323-6

Diese Arbeit ist gleichzeitig veröffentlicht in:

OPUS – Online Publikationen der Universität Stuttgart

http://dx.doi.org/10.18419/opus-13476, Stuttgart 2023

Adresse der DGK:

Ausschuss Geodäsie der Bayerischen Akademie der Wissenschaften (DGK)

Alfons-Goppel-Straße 11 ● D – 80 539 München

Telefon +49 – 331 – 288 1685 ● E-Mail post@dgk.badw.de

http://www.dgk.badw.de

Main referee: Prof. Dr.-Ing. Uwe Sörgel

Co-referee: Prof. Dr. Bernhard Höfle (Heidelberg University)

Date of defense: July 13, 2023

ISSN 0065-5325 ISBN 978-3-7696-5323-6

Contents

Abstract **5**

Kurzfassung **7**

Acronyms **9**

List of Symbols **11**

1 Introduction **13**
1.1 Motivation . . . 13
1.2 Objectives . . . 15
1.3 Outline . . . 17

2 Related Work **19**
2.1 Crowdsourcing Geospatial Information . . . 19
2.2 Ensuring Quality of Data when Generated by Lay Workers . . . 21
2.3 Motivating the Crowd to Contribute . . . 22
2.3.1 Increasing the Audience via Gamification . . . 22
2.3.2 Paid Crowdsourcing . . . 23
2.4 Crowdsourcing as Part of Hybrid Intelligence Systems . . . 26
2.5 Active Learning (AL) . . . 28
2.5.1 Querying Samples in AL . . . 28
2.5.2 Oracles in AL . . . 30
2.5.3 Terminating AL Loops . . . 31
2.5.4 AL in Remote Sensing and Semantic 3D Point Cloud Segmentation . . . 31
2.6 Transfer Learning . . . 33
2.7 Active Transfer Learning (ATL) . . . 34
2.8 Automated 3D Point Cloud Interpretation . . . 37
2.8.1 Feature-Driven Semantic Segmentation . . . 37
2.8.2 Data-Driven Semantic Segmentation . . . 39

3 Methodology **43**
3.1 An Outline of the Proposed Hybrid Intelligence System . . . 43
3.2 An Intuition *why* AL Works . . . 46
3.3 Sampling Schemes in the AL Loop . . . 49
3.3.1 Query Functions . . . 49
3.3.2 Fostering an Equally Class-Distributed Training Set . . . 52
3.3.3 Guaranteeing Diversity in Sampled Batches . . . 53
3.3.4 Addressing Imperfect Oracles in AL . . . 55
3.4 Active Transfer Learning . . . 56

3.5 Stopping the Loop . . . 59
3.6 The Crowd as AL Oracle: Working with Non-Experts for 3D Point Labeling . . . 60
3.6.1 Oracle Types in AL . . . 60
3.6.2 Designing Labeling Tools for the Crowd . . . 61
3.6.3 Automated Quality Control . . . 63
3.6.4 Which 3D Data Representation is Best Suited? . . . 64
3.6.5 Stimulate Motivation by Gamification . . . 65
3.7 Classifiers for 3D Semantic Segmentation . . . 68
3.7.1 Hand-Crafted Features & Random Forest . . . 68
3.7.2 Submanifold Sparse Convolutional Neural Network . . . 73
3.8 Automatization via the CATEGORISE Framework . . . 79

4 Data & Evaluation Metrics 83
4.1 Airborne Laser Scanning . . . 83
4.2 Data Sets . . . 84
4.2.1 ISPRS Vaihingen 3D Semantic Labeling Contest (V3D) . . . 84
4.2.2 ISPRS Hessigheim 3D Benchmark on Semantic Segmentation of High-Resolution Point Clouds and Meshes (H3D) . . . 85
4.2.3 Stuttgart 3D Point Cloud (S3D) . . . 87
4.3 Evaluation Metrics . . . 90

5 Experiments & Results 91
5.1 Evaluating and Optimizing the Machine Component . . . 91
5.1.1 Comparing Sampling Strategies . . . 91
5.1.2 Comparing Classifiers . . . 94
5.1.3 Considering Imperfect Oracles . . . 95
5.1.4 Simulation of the Hybrid Intelligence System . . . 97
5.2 Optimizing and Evaluating the Crowd Oracle . . . 100
5.2.1 A Statistical Analysis of Crowdsourced Data Annotation Tasks . . . 100
5.2.2 Evaluating the Performance of the Crowd as Labeling Engine . . . 101
5.2.3 The Impact of Different Data Modalities on the Performance of the Crowd . . . 103
5.2.4 Can the Crowd be Tricked by Gamification? . . . 105
5.2.5 The Impact of *Reducing Interpretation Uncertainty (RIU)* to Ease Labeling . . . 107
5.3 Formulating the Hybrid Intelligence System . . . 109
5.3.1 Employing the Crowd-Driven AL Loop for Efficient Segmentation of Airborne Laser Scanning (ALS) Point Clouds . . . 109
5.3.2 Evaluating the Effect of Different Data Modalities in AL Loops . . . 118
5.3.3 Assessing the Long-Term Flexibility of ATL . . . 119

6 Conclusion, Limitations and Outlook 127

Abstract

While in recent years tremendous advancements have been achieved in the development of supervised Machine Learning (ML) systems such as Convolutional Neural Networks (CNNs), still the most decisive factor for their performance is the quality of labeled training data from which the system is supposed to learn. This is why we advocate focusing more on methods to obtain such data, which we expect to be more sustainable than establishing ever new classifiers in the rapidly evolving ML field. In the geospatial domain, however, the generation process of training data for ML systems is still rather neglected in research, with typically experts ending up being occupied with such tedious labeling tasks. In our design of a system for the semantic interpretation of Airborne Laser Scanning (ALS) point clouds, we break with this convention and completely lift labeling obligations from experts. At the same time, human annotation is restricted to only those samples that actually justify manual inspection. This is accomplished by means of a hybrid intelligence system in which the machine, represented by an ML model, is actively and iteratively working together with the human component through Active Learning (AL), which acts as pointer to exactly such most decisive samples. Instead of having an expert label these samples, we propose to outsource this task to a large group of non-specialists, the crowd. But since it is rather unlikely that enough volunteers would participate in such crowdsourcing campaigns due to the tedious nature of labeling, we argue attracting workers by monetary incentives, i.e., we employ paid crowdsourcing. Relying on respective platforms, typically we have access to a vast pool of prospective workers, guaranteeing completion of jobs promptly. Thus, crowdworkers become *human processing units* that behave similarly to the *electronic processing units* of this hybrid intelligence system performing the tasks of the machine part.

With respect to the latter, we do not only evaluate *whether* an AL-based pipeline works for the semantic segmentation of ALS point clouds, but also shed light on the question of *why* it works. As crucial components of our pipeline, we test and enhance different AL sampling strategies in conjunction with both a conventional feature-driven classifier as well as a data-driven CNN classification module. In this regard, we aim to select AL points in such a manner that samples are not only informative for the machine, but also feasible to be interpreted by non-experts. These theoretical formulations are verified by various experiments in which we replace the frequently assumed but highly unrealistic error-free oracle with simulated imperfect oracles we are always confronted with when working with humans. Furthermore, we find that the need for labeled data, which is already reduced through AL to a small fraction (typically $\ll 1\,\%$ of Passive Learning training points), can be even further minimized when we reuse information from a given source domain for the semantic enrichment of a specific target domain, i.e., we utilize AL as means for Domain Adaptation.

As for the human component of our hybrid intelligence system, the special challenge we face is monetarily motivated workers with a wide variety of educational and cultural backgrounds as well as most different mindsets regarding the quality they are willing to deliver. Consequently, we are confronted with a great quality inhomogeneity in results received. Thus, when designing respective campaigns, special attention to quality control is required to be able to automatically reject submissions of low quality and to refine accepted contributions in the sense of the *Wisdom of the Crowds* principle. We further explore ways to support the crowd in labeling by experimenting with different data modalities (discretized point cloud vs. continuous textured 3D mesh surface), and also aim to shift the motivation from a purely extrinsic nature (i.e., payment)

to a more intrinsic one, which we intend to trigger through gamification. Eventually, by casting these different concepts into the so-called CATEGORISE framework, we constitute the aspired hybrid intelligence system and employ it for the semantic enrichment of ALS point clouds of different characteristics, enabled through learning from the (paid) crowd.

Kurzfassung

Während in den letzten Jahren enorme Fortschritte bei der Entwicklung überwachter Machine Learning (ML) Systeme wie Convolutional Neural Networks (CNNs) erzielt wurden, ist der entscheidendste Faktor für deren Leistung nach wie vor die Qualität der gelabelten Trainingsdaten, aus denen das System lernen soll. Deshalb plädieren wir dafür, sich mehr auf Methoden zur Gewinnung solcher Daten zu konzentrieren, was wir als nachhaltiger einschätzen als die Entwicklung immer neuer Klassifikatoren in dem sich schnell entwickelnden ML-Bereich. Für Geodaten wird der Prozess der Generierung von Trainingsdaten für ML-Systeme in der Forschung jedoch immer noch eher vernachlässigt, wobei typischerweise Experten mit solch langwierigen Annotationsaufgaben betraut werden. Bei der Entwicklung eines Systems für die semantische Interpretation von Airborne Laser Scanning (ALS) Punktwolken brechen wir mit dieser Konvention und entlasten Experten vollständig von jeglichen Labelingaufgaben. Gleichzeitig wird die menschliche Interaktion auf die Instanzen beschränkt, die eine manuelle Inspektion tatsächlich rechtfertigen. Dies wird durch ein hybrides Intelligenzsystem erreicht, bei dem die Maschine, die durch ein ML-Modell repräsentiert wird, aktiv und iterativ mit der menschlichen Komponente durch Active Learning (AL) zusammenarbeitet, wobei AL als Indikator für genau solche besonders entscheidenden Instanzen fungiert. Anstatt diese von einem Experten labeln zu lassen, schlagen wir vor, diese Aufgabe an eine große Gruppe von Laien, die Crowd, auszulagern. Da es jedoch eher unwahrscheinlich ist, dass sich genügend Freiwillige an solchen Crowdsourcing-Kampagnen beteiligen würden, weil das Labeln von Natur aus mühsam ist, plädieren wir dafür, Arbeitskräfte durch monetäre Anreize zu gewinnen, d.h., wir setzen bezahltes Crowdsourcing ein. Werden hierzu entsprechende Plattformen genutzt, haben wir in der Regel Zugang zu einem riesigen Pool potenzieller Arbeitskräfte, sodass eine rasche Erledigung der Aufgaben gewährleistet ist. So werden Crowdworker zu *menschlichen Prozessoren*, die sich ähnlich verhalten wie die *elektronischen Prozessoren* dieses hybriden Intelligenzsystems, welche die Aufgaben des maschinellen Anteils des Systems übernehmen.

In Bezug auf Letzteres evaluieren wir nicht nur, *ob* eine AL-basierte Pipeline für die semantische Segmentierung von ALS-Punktwolken geeignet ist, sondern beleuchten auch die Frage, *warum* sie so erfolgreich ist. Als entscheidende Komponenten unserer Pipeline testen und erweitern wir verschiedene AL-Selektionsstrategien in Verbindung mit einem konventionellen merkmalgestützten Klassifikator sowie einem datenbasierten CNN-Klassifikator. In diesem Zusammenhang zielen wir darauf ab, AL-Punkte so auszuwählen, dass diese nicht nur für die Maschine informativ sind, sondern auch von Laien interpretiert werden können. Diese theoretischen Formulierungen werden durch verschiedene Experimente verifiziert, in denen wir das häufig angenommene, aber höchst unrealistische fehlerfreie Orakel durch simulierte unvollkommene Orakel ersetzen, mit denen wir bei der Arbeit mit Menschen immer konfrontiert sind. Darüber hinaus stellen wir fest, dass der Bedarf an gelabelten Daten, der durch AL bereits auf einen kleinen Bruchteil (typischerweise $\ll 1\,\%$ der Trainingspunkte für Passive Learning) reduziert ist, noch weiter minimiert werden kann, wenn wir Informationen aus einer gegebenen Ursprungsdomäne für die semantische Anreicherung einer spezifischen Zieldomäne wiederverwenden, d.h., wir nutzen AL als Mittel für Domain Adaptation.

Was die menschliche Komponente unseres hybriden Intelligenzsystems betrifft, so stehen wir vor der besonderen Herausforderung, dass wir es mit monetär motivierten Arbeitskräften zu tun haben, deren

Bildungsstand und kultureller Hintergrund ebenso unterschiedlich sind wie ihre Einstellung zur Qualität, die sie zu liefern bereit sind. Folglich sind wir mit einer großen Qualitätsinhomogenität der erhaltenen Ergebnisse konfrontiert. Bei der Gestaltung entsprechender Kampagnen ist daher ein besonderes Augenmerk auf die Qualitätskontrolle zu legen, sodass qualitativ minderwertige Erfassungen automatisch abgelehnt werden können und akzeptierte Beiträge im Sinne des *Wisdom of the Crowds* Prinzips optimiert werden können. Darüber hinaus erforschen wir Möglichkeiten, die Crowd bei der Annotierung zu unterstützen, indem wir mit verschiedenen Datenmodalitäten experimentieren (diskretisierte Punktwolke vs. kontinuierliche texturierte 3D Mesh-Oberfläche) und zielen auch darauf ab, die Motivation von einer rein extrinsischen Natur (d.h. Bezahlung) zu einer eher intrinsischen zu verlagern, die wir durch Gamification auslösen wollen. Indem wir diese verschiedenen Konzepte in das so genannte CATEGORISE-Framework einbringen, bilden wir schließlich das angestrebte hybride Intelligenzsystem und setzen es für die semantische Anreicherung von ALS-Punktwolken mit unterschiedlichen Eigenschaften ein, was durch das Lernen von der (bezahlten) Crowd ermöglicht wird.

Acronyms

AI Artificial Intelligence

AL Active Learning

ALS Airborne Laser Scanning

ATL Active Transfer Learning

CART Classification and Regression Trees

CATEGORISE Crowd-Based Active Learning for Point Semantics

CIR False Color Infrared

CNN Convolutional Neural Network

CRF Conditional Random Field

CV Computer Vision

DA Domain Adaptation

DL Deep Learning

DiFS Diversity in Feature Space

DiOS Diversity in Object Space

DSM Digital Surface Model

DTM Digital Terrain Model

FWF Full-Waveform

GT Ground Truth

GSD Ground Sampling Distance

HSV Hue, Saturation, Value

H3D ISPRS Hessigheim 3D Benchmark on Semantic Segmentation of High-Resolution Point Clouds and Meshes

ISPRS International Society for Photogrammetry and Remote Sensing

Laser Light Amplification by Stimulated Emission of Radiation

LGL State Office for Spatial Information and Land Development Baden-Wuerttemberg

LiDAR Light Detection and Ranging

ML Machine Learning

MLP Multi-Layer Perceptron

MLS Mobile Laser Scanning

MRF Markov Random Field

MTurk Amazon Mechanical Turk

MW microWorkers

NMA National Mapping Agency

OA Overall Accuracy

OSM OpenStreetMap

PL Passive Learning

PTL Passive Transfer Learning

ReLU Rectified Linear Unit

RF Random Forest

RGB Red, Green, Blue

RIU Reducing Interpretation Uncertainty

SCN Submanifold Sparse Convolutional Neural Network

SVM Support Vector Machine

S3D Stuttgart 3D Point Cloud

TL Transfer Learning

TLS Terrestrial Laser Scanning

TOP True Orthophoto

UAV Unmaned Aerial Vehicle

VGI Volunteered Geographical Information

V3D ISPRS Vaihingen 3D Semantic Labeling Contest

List of Symbols

Symbol	Meaning
Active Learning Loop and its Components	
i	Active Learning iteration step
n_i	number of Active Learning iteration steps
U	pool of unlabeled points
L	pool of labeled points
n_L	number of samples included in the pool of labeled points L
L_{raw}	pool of labeled points as obtained from the first crowd campaign (see *Type A* in Section 3.6.2)
n_j	number of samples to be annotated per class in *Type A* crowd campaigns thus constituting L_{raw} (cf. Section 3.6.2)
L_{init}	pool of labeled points after filtering L_{raw} via *Type B* crowd campaigns (cf. Section 3.6.2)
L_i	pool of labeled points as obtained in the i^{th} iteration step in an Active Learning loop
M	Machine Learning model employed within the Active Learning loop
Active Learning Sampling Strategies	
$\boldsymbol{x}^+$	point entity to be labeled and added to the training data set
n^+	batch size, number of samples to be labeled and added to the training set in each iteration step
s	sampling score of a specific instance $\boldsymbol{x}$
E	*entropy*
VE	*vote entropy*
n_c	number of samples of a specific class in the current labeled training pool L
w_c	individual weight of a specific class c (dynamically computed within the AL loop)
d_{DiOS}	distance of samples in *Diversity in Object Space* sampling
d_{RIU}	distance of samples in *Reducing Interpretation Uncertainty* sampling
Active Transfer Learning	
D_S	source domain
D_T	target domain
$\boldsymbol{x}^-$	point entity to be removed from the training set
n^-	number of source domain samples to be removed from the training set
q	ratio of n^-/n^+
Stopping Criterion	
C_o	congruence over all classes
C_{ac}	averaged class-wise congruence
d_{stop}	distance of current iteration step to a previous one to compare against
n_{stop}	number of previous iteration steps to be considered for standard deviation computation
t_{stop}	threshold for the stopping criterion

Oracles	
$\mathcal{O}_O$	omniscient Ground Truth oracle
$\mathcal{O}_N(\sigma)$	noisy oracle with an error rate of $\sigma\,\%$
$\mathcal{O}_S(\sigma)$	confused oracle with an error rate of $\sigma\,\%$
$\mathcal{O}_C$	real crowd oracle
$\mathcal{O}_{CP}$	real crowd oracle being provided with point cloud data
$\mathcal{O}_{CM}$	real crowd oracle being provided with mesh data
n_{cw}	number of crowdworkers tasked with the same job, i.e., number of multiple acquisitions
n_{mult}	number of crowdworkers contributing to the pool of acquisitions necessary for determining the optimal n_{cw} value (cf. Section 3.6.3)
Classifiers (for the sake of clarity, classifier dependent parameters are not listed)	
$\mathcal{P}$	point cloud with n_p points
n_p	number of points in a point cloud $\mathcal{P}$
Ω	set of classes with n_Ω entries
n_Ω	number of classes in a specific class catalog Ω
$\boldsymbol{x}$	feature vector of a specific point entity
$p(c\|\boldsymbol{x})$	posterior probability that an instance with a specific feature vector $\boldsymbol{x}$ belongs to class c
$\mathcal{E}$	ensemble classifier with n_e members
n_e	number of committee members of an ensemble classifier $\mathcal{E}$

Chapter 1

Introduction

1.1 Motivation

In the last three decades, Machine Learning (ML) systems, both from the feature-driven and data-driven domain, with Convolutional Neural Networks (CNNs) as their currently most prominent representative, have experienced considerable attention in research. As a result, the performance of machines with minimum error rates even exceeds that of human beings (Russakovsky et al., 2015a). Thus, such systems have already become, or will become, essential in everyday life, such as for autonomous driving, speech recognition, or simply for generating watching proposals on streaming platforms or buying suggestions. However, the success of respective ML models solely stems from learning from versatile corpora of annotated training data. Hence, purely tuning the architecture of ML models as mere tools to infer interrelationships from data, will have only limited effect when the underlying training set is of suboptimal quality, e.g., being too specific for proper generalization (for instance, in case of simulated closed world scenarios) or simply being poorly labeled. For this reason, Ng (2021) only recently advised paying more attention to the careful creation of training data in such a manner that the development of ML systems is more of *data-centric* nature instead of being mainly *model-centric*. Following this mindset, companies focusing on data collection, such as *Google*, have a clear advantage in generating the most successful ML systems. Actually, in our everyday online life, we are not only entangled with ML on an operative level, but are also actively involved in *Google*'s collection of training data by means of *reCAPTCHA*s (von Ahn et al., 2008), which are typically encountered as turing test to block spammers or malicious access attempts. But since such companies tend to not allow a wider public to access their data pools, efforts have been made to build large-scale and freely available training data sets such as *ImageNet* (Deng et al., 2009; Russakovsky et al., 2015a).

Typically, the generation of training data and the teaching of an ML model based on that data are considered two separate and self-contained steps, with the latter being far more prominent in research than the supposedly simple but tediously perceived annotation process. But in fact, great potential lies in considering these two processes as entangled in the form of a symbiotic circle, with each aspect, i.e., both the ML model and the labeling engine, drawing advantage from working together. This aspired interaction roots from the capability of the machine to scan through massive amounts of data to identify those samples that can play an important role in forming a strong separation hypothesis between different classes. If the machine is allowed to communicate with the labeling engine, annotation effort can be focused on only those most informative samples the machine actually profits from, offering the potential to drastically minimize the amount of labels that are actually required. This link between the two actors, i.e., the ML model and the so-called oracle capable of providing answers to queries from the machine, is established by means of Active Learning (AL) (Settles, 2009). Such an iterative learning scheme assumes exactly this *active* interaction

between the two parties standing opposed to the conventional Passive Learning (PL) concept, where a static training set potentially lacking most critical examples while possibly containing redundant information is expected to be sufficient for the task at hand.

While the AL scheme is a viable alternative to boost the efficiency of ML models, it still requires the involvement of a human operator in the process, assuming the role of the oracle that is responsible for returning labels to the machine. Typically, this is accomplished by respective experts of the data in question (Waldhauser et al., 2014). Although their labeling burden can already be minimized through AL, significant effort is still needed. In a scalable approach, we would therefore split the task into many subtasks that can be quickly completed in parallel by a multitude of different annotators. Thus, each of these individual annotators acts as a *human processing unit*, behaving similarly to *electronic processing units* (Gingold et al., 2012) as employed for the ML part. When linking those *electronic* and *human processing units* by means of AL, we receive a hybrid intelligence system (Vaughan, 2018), where both parties are concerned with the task they are at their best, i.e., the unsurpassed interpretation capabilities of humans and the machine's ability for quick completion of well-defined, repetitive tasks for exploring vast data sets, with especially uncovering most informative examples and building automated ML models upon them (Waldhauser et al., 2014; Russakovsky et al., 2015b).

Although the necessary amount of labeling effort would be significantly reduced for each individual expert, from a practical point of view, running such an expert-driven system is infeasible as it is unlikely that enough experts of a designated field would actually participate. But even if so, the time complexity, mainly depending on the labeling speed of the experts who are predominantly occupied with other tasks, can be significant. Thus, and since the greater goal is to completely lift the burden of data annotation from experts, we need to turn to another, potentially larger audience, i.e., the crowd (of internet users), which is already involved in non-expert data annotation in context of the aforementioned *reCAPTCHA*s. Apart from this often unconscious participation of the crowd, tasks can be explicitly offered to workers on a voluntary basis. However, in case of such volunteered crowdsourcing, a lack of participants is also likely whenever the task is not as appealing as in citizen science projects such as the *SETI@home* project (Korpela et al., 2001), but rather concerns labeling tasks that are tedious by design. In this context, paid crowdsourcing offers a viable alternative. The idea is that tasks are outsourced to prospective contributors by means of an open call, offering them a certain amount of payment in return. This is intended to act as an extrinsic incentive instead of an intrinsic one in the volunteered crowdsourcing scenario. Building the AL oracle on such a paid crowdsourcing system thus offers the possibility to form a practical hybrid intelligence system that runs in a fully automated manner (from the view of the system operator). Although human beings, i.e., crowdworkers, are part of this system, these *human processing units* behave just like the electronic ones and deliver timely results due to the vast pool of available workers on the internet, where it is guaranteed to find enough participants to complete campaigns.

However, the special challenge of hybrid intelligence systems is the integration of non-deterministic components, i.e., *human processing units*, requiring special attention to this human aspect with respect to automated quality control (Waldhauser et al., 2014; Ye et al., 2017). This issue becomes even more difficult when we confront non-experts (i.e., the crowd) with rather special data, such as geospatial data, which typically incorporates an unfamiliar perspective on actually known environments. Even more severe, in case of (colorized) 3D Airborne Laser Scanning (ALS) point clouds, crowdworkers are additionally required to have a certain interpretation capability of 3D data, which cannot necessarily be expected as given. However, geospatial 3D data is exactly the field of application in which such a hybrid intelligence system could empower the true potential of modern ML systems by generating suitable training data on the fly, since despite recent efforts to provide open, large-scale and fine-grained labeled data sets (Niemeyer et al., 2014; Actueel Hoogtebestand Nederland, 2021; Zolanvari et al., 2019; Varney et al., 2020; Ye et al., 2020; Kölle et al., 2021a), such representatives are still scarce.

1.2 Objectives

This thesis is dedicated to establishing such a hybrid intelligence system for the special case of remote sensing data, and more precisely for the automated semantic segmentation of ALS point clouds, as depicted in Figure 1.1. Thus, a respective framework is to be developed to build up an ML model capable of enriching an arbitrary point cloud (cf. Figure 1.1, *top*) with semantic information (cf. Figure 1.1, *bottom*). This framework is intended not only to minimize the need for labeled data to a small fraction of training points (cf. Figure 1.1, *middle*), but also to completely lift the obligation of expert involvement in any labeling task. By formulating such a system, we consequently aim to put an end to the long-held understanding in remote sensing that providing training data is an expert's burden that is to be dealt with in a first self-contained step as prerequisite to employ a respective ML model (Waldhauser et al., 2014). To this end, we bring together two research fields in which we identify our contributions as follows:

Active Learning for 3D Point Cloud Semantic Segmentation

- While AL is a well-studied topic for the semantic enrichment of remote sensing imagery, only little work has been conducted with respect to applying AL for 3D point cloud semantic segmentation. Consequently, we aim to explore the capabilities of AL in this domain to significantly reduce the amount of necessary labeled training data and shed light not only on the question as to *whether* it works, but also as to *why* it works (Kölle et al., 2021a).

- Since the most crucial step of any AL system is the identification of most informative points that are sufficient to train an ML model with a similar performance as one trained on a vast amount of labeled training samples in a PL scenario, we compare the suitability of different sampling strategies found in

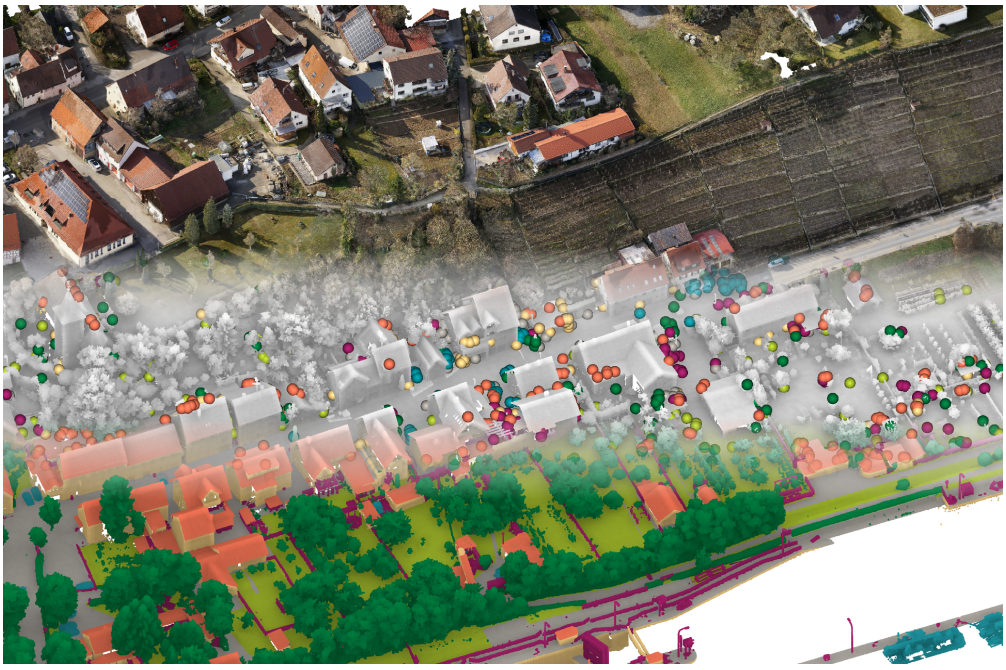


Figure 1.1: An overview of the subject of this thesis: from any given point cloud (*top*), we aim to derive a semantic segmentation (*bottom*) with a limited number of point-wise training samples (*middle*), which are generated *without* the involvement of an expert in labeling. This is achieved in context of a hybrid intelligence system constituted by linking the crowd with an ML model by means of AL.

literature and enhance those with add-ons to boost their performance. In this regard, we exploit the properties of our geospatial data to derive sampling strategies that are not only optimal for the machine part by selecting only most informative samples, but we also aim to facilitate the labeling process by selecting points that are easier to label for a human operator, i.e., the crowd. This means we tailor point queries to our specific needs of non-expert *human processing units* in the hybrid intelligence system we advocate for. Evaluation of these sampling strategies is conducted in tandem with a classifier both from the conventional feature-driven domain as well as from the data-driven domain. This allows us to give a recommendation of the more efficient classifier in our special AL scenario (Kölle et al., 2021a).

- Although AL is an efficient means to minimize the number of training labels required, resources are still wasted when an ML model has to be established from scratch for any new data set, which is typically the case since training an ML model that generalizes well across multiple different data sets is still an open research question. To ease this hindrance, we explore the capabilities of an AL-based transfer technique, Active Transfer Learning (ATL), that utilizes labeled samples from a source domain whenever they are meaningful for the target domain, but that also eliminates conflicting source domain samples while also identifying most informative samples from the new and unexplored target domain (Kölle et al., 2022).

- We lift the assumption of most research in AL, which expects an error-free agent that always returns true labels for the queries posed by the machine, which is completely unrealistic in a practical scenario where humans (whether they are experts in the field or not) are tasked with labeling (Marcus and Parameswaran, 2015). Therefore, on one hand, we simulate different error behaviors of the oracle to gain a realistic estimation of theoretically reachable accuracies. On the other hand, we turn away from any sort of simulated offline labeling engine and embrace a real human crowd as viable alternative, leading us to our second research branch (Kölle et al., 2021a,b).

Paid Crowdsourcing for the Interpretation of 3D Geospatial Data

- When (paid) crowdworkers are intended to act as oracles in an AL scenario, this also requires to design intuitive labeling tools that are easily comprehensible for non-experts, which is acknowledged to be an often overlooked challenge (Kittur et al., 2008). In this regard, having to confront crowdworkers with 3D data even amplifies this challenge, since although the dissemination of 3D data through geospatial services (such as *Google's*) might have fostered an understanding of such data, it is still likely that a considerable fraction of workers are not familiar with this kind of data. In developing tools suitable for crowdsourced 3D data annotation, we enter thus almost uncharted territory. In this context, and also motivated by *Google's* 3D data presentation, we explore whether it is beneficial to illustrate individual points to be labeled not only by the obvious choice of the 3D (colored) point cloud, but by a continuous and well-textured surface representation, i.e., a 3D textured mesh, as it is the case for *Google's* geospatial services (Kölle et al., 2021b).

- Since crowdsourcing tasks come with the inherent issue of quality inhomogeneity of collected data, this is especially to be accounted for. While it would be straightforward having an operator check the results of crowdworkers, this would drastically diminish the appeal of crowdsourcing. Thus, such quality control measures should be capable of working in an automatic manner without the need for operator involvement. To this end, we make use of measures for quality control both on task designing (which is related to the previous bullet) as well as in post-processing, specifically by implementing the idea of the *Wisdom of Crowds* (Howe, 2006) to receive high-quality labels (Kölle et al., 2021b).

- While both volunteered and paid crowdsourcing campaigns suffer from inhomogeneity of contributed data, the motivation of crowdworkers is significantly different in the two concepts (intrinsic vs. extrinsic), where we suppose that a paid, i.e., an extrinsically motivated worker, operates less carefully than one being intrinsically motivated. Therefore, we try to stimulate the motivation of our workers by means

of gamification, with the intention of both receiving results of better quality and luring workers *to play just one more time* (von Ahn and Dabbish, 2008). Although creating an enjoyable (labeling) game is a challenging endeavor, in the geospatial domain, we are fortunate as virtual 3D worlds, typically found in games, can be created just by visualizing our everyday 3D data (Walter et al., 2022a).

- Since we postulate that a hybrid intelligence system linking the machine part and the paid crowd through AL is suitable to completely relief an expert from any labeling duties and can run fully automatically, at least from the view of the operator (where human work can be considered as running *human processing units*), we also discuss the practical aspects that are required to actually run such a system. The resulting framework denoted as Crowd-Based Active Learning for Point Semantics (CATEGORISE) can also serve as one possible blueprint when such a system is to be established for a similar or related task of enabling ML in the geospatial domain (Kölle et al., 2021b).

1.3 Outline

As the purpose of this thesis is the formulation of a hybrid intelligence system by combining the individual fields of paid crowdsourcing with automated 3D point cloud semantic segmentation through AL, these are also the units structuring this thesis. We start with a comprehensive review of the current state of the art in these research fields in Chapter 2. This is followed by a detailed discussion of our methodology in Chapter 3, addressing both the human and the machine component of the hybrid system. As it is built upon an AL backbone, we begin this chapter with a comprehensive discussion of all AL related aspects to efficiently focus manual labeling effort on only those instances that actually justify human processing. To minimize such human involvement even further, we also apply a re-formulation of conventional AL runs to become suitable for conducting ATL. With regard to our *human processing units* (i.e., the crowdworkers) in this system, special emphasis is placed on measures to ease labeling as much as possible, to boost our crowdworkers' intrinsic motivation and to work towards a high-quality crowd-based data acquisition by means of various automated quality control methods.

After briefly presenting our data sets of interest in Chapter 4, consequently our experimental part in Chapter 5 is structured in the same manner as before. To estimate the theoretical performance of AL for 3D point cloud segmentation, we first evaluate results solely with respect to the machine part, without relying on real crowdworkers, but simulate imperfect oracles instead. Subsequently, we will focus exclusively on the outcome when real paid crowdworkers are engaged in 3D point cloud interpretation, before eventually evaluating the performance of a real-world hybrid intelligence system powered by paid crowdworkers. We dismiss the reader with a conclusion of the presented work in Chapter 6, as well as a discussion of limitations of our work potentially providing sparks for future research.

Chapter 2

Related Work

In context of creating the aspired hybrid intelligence system, a multitude of different components need to be considered and different concepts of literature need to be combined to succeed. For one, this affects establishing a powerful *human processing unit* (used interchangeably with the term crowd engine in the following) that is supposed to teach the machine part. Thus, the human aspect should perform as error-free as possible, which can be an obstacle when non-experts are intended to interpret unfamiliar geospatial data (Section 2.1). Hence, special measures are to be taken into account to guarantee high-quality results (Section 2.2) and to have access to a corpus of crowdworkers that is large enough to complete the task at hand (Section 2.3). The second part of the hybrid intelligence system (Section 2.4) is then to employ an ML scheme that is suitable for querying the crowd engine only when necessary (Section 2.5), and that makes use of pre-existing information whenever possible (Section 2.6 & 2.7) to finally develop a powerful ML algorithm for 3D point cloud classification (Section 2.8).

2.1 Crowdsourcing Geospatial Information

The broad dissemination of the Web 2.0 paradigm, with the internet no longer being seen only as a source of information made available by specific providers, but also allowing individual users to actively contribute information, has led to a whole new era of internet content and usage (Wilson et al., 2011). Nowadays, the internet is especially utilized as means for information sharing and collaboration. Thus, Web 2.0 promotes and enables new working concepts such as crowdsourcing, which means outsourcing tasks originally assigned to individuals or a single institution to a large group of persons, i.e., the crowd. Although the crowdsourcing term was only coined in 2006 by Jeff Howe (Howe, 2006), its basic concept has been successfully applied in the past as well. For instance, in 1714, the British Parliament posted the so-called *longitude act* - an open call to solve the problem of exactly determining a ship's longitude at sea, which had been a persisting problem for decades and was finally solved by an individual member of the crowd, John Harrison, by engineering a highly precise clock (Spencer, 2012).

Basically, the same idea is still being pursued by modern crowdsourcing campaigns on dedicated platforms such as *Wazoku*[1] (formerly *InnoCentive* at the time of examination by Vignieri (2021)) or *Upwork*[2]. But nowadays solutions are often built by a large corpus of crowdworkers acting in a collaborative manner, e.g., for creating and further developing of publicly available software (Carillo and Okoli, 2008). Often, comprehensive tasks one single person may not be capable of performing alone are split into many subtasks (referred to as microtasks) that are completed by individuals, thus forming a distributed computing engine

[1] `https://www.wazoku.com/`
[2] `https://www.upwork.com/`

(Kittur et al., 2011; Allahbakhsh et al., 2013) or *human cloud computing* engine (Hoßfeld et al., 2011). Prominent examples are related to the concept of citizen science, where interested individuals assist research, for instance, by scanning large data volumes for extraterrestrial radio sources as pursued by the *SETI@home* project (Korpela et al., 2001). Similarly, projects such as *GeoWiki*[3] and *Ushahidi*[4] (Okolloh, 2009) aim to rely on volunteers to conduct environmental monitoring (e.g., volunteers are asked to identify reasons for deforestation of the Amazon forest by analyzing satellite imagery), to map observations of locals to enable post-crisis responding or to georeference irregularities of elections.

Also, the crowd equipped with appropriate sensors (*crowd sensing*) can become a network of monitoring devices (Resch, 2013), like the *GeigerCrowd* project created in the aftermath of the Fukushima incident. Goodchild (2007) branded this kind of user-generated crowdsourced geodata as Volunteered Geographical Information (VGI), which has established itself in the geospatial domain (See et al., 2016). Its most prominent example is *OpenStreetMap (OSM)* (Haklay and Weber, 2008; Budhathoki and Haythornthwaite, 2012), where users from all over the globe collaborate for generating comprehensive and freely available map data, which is often of higher detail richness and more up-to-date than conventional map data from National Mapping Agencys (NMAs) (Antoniou and Skopeliti, 2015). This also explains the interest of NMAs in incorporating the crowd for authoritative data to cover the increasing need for highly detailed and up-to-date maps, which were already identified to be costly (Bomford, 1980) but inevitably necessary (Bossler et al., 1990).

However, the main concern of NMAs and also a hot recent research topic is the quality (Goodchild and Li, 2012; Fan et al., 2014; Senaratne et al., 2016) and thus credibility (Flanagin and Metzger, 2008; Leibovici et al., 2017) of crowdsourced (geospatial) data. The crowd is composed of a multitude of individuals from different cultures, of different ages and each having their own educational history (Kittur et al., 2008). Although this inherent diversity is often an advantage in yielding a representative result (Howe, 2006), it also leads to heterogeneous data quality (Howe, 2006; Goodchild, 2007; Haklay and Weber, 2008; Shaw et al., 2011; Dorn et al., 2015; Fonte et al., 2017; Chandler and Paolacci, 2017; Chandler and Kapelner, 2013). This is because crowdworkers are often not acquainted with specific data concepts (such as geodata) (Hashemi and Abbaspour, 2015), and acquisition standards are often omitted (Antoniou and Skopeliti, 2015) in favor to not deter eager individuals (Fonte et al., 2017). More severely, there might be inattentive crowdworkers (Fleischer et al., 2015) or even malicious crowdworkers deliberately delivering false input (Hirth et al., 2013; Welinder et al., 2010b; Whitehill et al., 2009). Also affecting the trustworthiness of crowdsourced projects (such as *OSM*'s) is that oftentimes considerable content is created by a small number of contributors (Haklay and Weber, 2008), often referred to as *power of the elites* (Basiri et al., 2019; Whitehill et al., 2009), possibly resulting in a biased data base.

If crowdsourced data is to be combined with, or is even to replace, NMA data (Waldner et al., 2019), it needs to achieve a certain level of trust (at a level similar to NMA data) and it also needs to be accepted that in favor of detail and up-to-dateness, some errors might be present (Cho and Crompvoets, 2019). For gaining more trust, data quality of crowdsourced mapping projects such as *OSM* is extensively studied and general concepts of impact factors, quality assessment and assurance are developed (Daniel et al., 2018). Although quality measures for conventional geodata such as completeness, logical consistency, positional accuracy, temporal accuracy, and thematic accuracy (Antoniou and Skopeliti, 2015; Fonte et al., 2017) are to be considered for crowdsourced geodata as well, evaluating these measures against a gold standard is demanding since respective reference data, for example, the one provided by NMAs, is often not suitable for direct comparison due to lower detail richness and temporal inaccuracies. Hence, research shifts to identification of intrinsic quality measures (e.g., in case of *OSM*: frequency of modifications of specific objects, population density in the respective area of the object or experience of the contributor (Antoniou and Skopeliti, 2015)). Often, this is challenging, since even if crowdworkers agree on a result, it might differ from actual Ground Truth (GT) (Salk et al., 2015).

[3] `https://www.geo-wiki.org/`
[4] `https://www.ushahidi.com/`

2.2 Ensuring Quality of Data when Generated by Lay Workers

To guarantee quality of crowdsourced data, Zhang et al. (2016) distinguish between *quality control on task designing* and *quality improvement after data collection*. An overview of such methods can also be found in the work of Chandler et al. (2013).

Quality control on task designing can naturally be realized by posting the task in a manner that is easy to understand for non-experts, which concerns both the data they have to work on and the specific action that is required (Sorokin and Forsyth, 2008). Sorokin and Forsyth (2008) and Allahbakhsh et al. (2013) recommend a cheat-proof task design and suggest filtering of participating groups by employers. Typical measures for guaranteeing quality are qualification tests (Patterson et al., 2014; Estes et al., 2016; Endres et al., 2010), a training stage which needs to be completed in advance of the task and/or inclusion of tasks where true answers are already known (Sorokin and Forsyth, 2008; Estes et al., 2016; Gebru et al., 2017; See et al., 2013; Hirth et al., 2013; Zhou et al., 2014; Salk et al., 2015; Marcus and Parameswaran, 2015; Vondrick et al., 2012). The inclusion of such a *gold standard* or *sentinel operations* (Gingold et al., 2012) might also help to determine skill levels for individual participants. In the same spirit, another intrinsic quality measure can be easily realized by presenting certain data points multiple times to the same crowdworker to assess whether the results are consistent (See et al., 2013). Although time spent on conducting specific tasks might indicate careless task completion, See et al. (2013) argue that this measure is not suited for quality control since it also expresses skill of contributors, so that malicious and good faith contributions are hard to distinguish (Kittur et al., 2008).

A straightforward implementation of the second concept, *quality improvement after data collection*, are reviews of already submitted tasks by either an expert (e.g., the employer) or other crowdworkers, which might help in discarding false results (Liu et al., 2018; Russell et al., 2007). More sophisticated methods are often based on the idea of the *Wisdom of the Crowds*, first mentioned by Galton (1907) and elaborated for a multitude of applications by Surowiecki (2004). It basically means that a group of independent individuals is capable to bring forth results of similar (or even better) quality than any single (expert) member of that group. This concept is related to *Swarm Intelligence* as can be observed in nature (Krause et al., 2011; Simons, 2004), but this term is only partly valid for production of crowdsourced data, because the *Swarm Intelligence* arises from the collaboration of independent individuals (Krause et al., 2010; Ikeda et al., 2016), which is not necessarily the case for crowdsourced projects.

The consequent translation of the *Wisdom of the Crowds* paradigm to a rule of action for crowdsourced data acquisition would be to multiply a given task, propose it to a corresponding number of crowdworkers and aggregate those contributions, which can lead to high-quality results (Sorokin and Forsyth, 2008) - the more (reasonable) contributors, the better (Sheng et al., 2008). In case of assigning a finite set of classes to an entity, i.e., labeling or categorization, the simplest approach for aggregation is majority voting and plurality voting in the non-binary case respectively (Parhami, 1994). When collecting free-form geometries where an infinitely high number of forms is possible, aggregation can be achieved by conflation (Xavier et al., 2016). More sophisticated methods aim to estimate and consider individual worker quality scores to form a weighted consensus (Dawid and Skene, 1979; Zhou et al., 2012; Zhang et al., 2016; Zheng et al., 2017) and allow the detection of noisy submissions or malicious workers (Whitehill et al., 2009). However, the procedure is limited by the requirement that each crowdworker labels multiple instances (e.g., 40 in case of Zhou et al., 2012) in order to reasonably estimate his confusion behavior. Hence, majority voting is often preferred (Sheng et al., 2008; Deng et al., 2014; Zheng et al., 2017; Hecht et al., 2018; Hirth et al., 2013; Marcus and Parameswaran, 2015; Endres et al., 2010). Oftentimes, such results achieve the same level of quality (or, vice versa, the same level of noise in case of ambiguities) as contributions from experts (Snow et al., 2008; Mao et al., 2013; See et al., 2013; Walter and Soergel, 2018; Raykar et al., 2010). An inherent drawback of this approach, on the other hand, is increased costs due to assigning the same task to multiple crowdworkers. Recent research aims to evaluate from how many repetitions the effect of the *Wisdom of the Crowds* kicks in (van Dijk et al., 2020; Walter et al., 2022b). Related to this concept is *Linus' Law*, which

suggests that for long-term projects such as *OSM*, errors will eventually be identified by other users and eliminated (Haklay et al., 2010; Goodchild and Li, 2012).

2.3 Motivating the Crowd to Contribute

For the aforementioned crowdsourcing projects, the incentive of individuals for participating is of intrinsic nature. Often, crowdworkers are eager to contribute for getting a feeling of community, referred to as heavyweight collaborators in contrast to lightweight members of the crowd (Budhathoki and Haythornthwaite, 2012). Others get involved out of curiosity to take part in appealing endeavors (Korpela et al., 2001), to contribute to a humanitarian cause and out of their wish to help others (De Albuquerque et al., 2016). Also, the creation of tools helpful for everyday life such as *Wikipedia* (competing with conventional encyclopaedias (Giles, 2005)) or *OSM* (Haklay and Weber, 2008; Herfort et al., 2016; Duggan, 2019), where many users strive to fill gaps on maps (Budhathoki and Haythornthwaite, 2012), can be appealing offers.

Consequently, this poses the question of how to properly leverage the *human computing capacity* of crowdworkers to conduct tasks that might not be appealing by design such as labeling of data. Stork (1999) raises the hope that volunteers can be motivated for labeling, just like for citizen science projects when contributors can watch the system gradually developing its interpretation capabilities in analogy to parents watching their child evolve. Russell et al. (2007) propose an open, web-based and ML-supported tool for object annotation, *LabelMe*, where submissions become instantly available as part of the dynamic *LabelMe* data base, which might be appealing at least for the members of the Computer Vision (CV) community.

An elegant way to engage the wider public in an annotation campaign is to embed labeling tasks in everyday actions, that is security checks in the form of *CAPTCHAs* (von Ahn et al., 2003). These are often used on websites to prove that requests are prompted by a human and not a machine, by solving a simple task (with known result) computers are supposed to be not capable of. Von Ahn et al. (2008) suggest requesting answers to not one but two *CAPTCHA* tasks (referred to as *reCAPTCHA*), where the answer of the second task is unknown and given by the operator without being informed which task is actually required to pass the test. Usually, those new tasks are presented to multiple crowdworkers to integrate results. Hillen and Höfle (2015) demonstrated that the *reCAPTCHA* concept is also suitable for more complex annotation of free forms in aerial images, underlining the power of this concept. The challenge for such projects is to sufficiently disseminate one's own *reCAPTCHA* model among frequently visited websites competing with other providers such as *Google*.

2.3.1 Increasing the Audience via Gamification

Another idea to attract more participants in campaigns can be gamification (Marczewski, 2013). This means using gaming elements, such as points, badges, ranks, leader boards, levels, etc. (Feyisetan et al., 2015) in non-gaming environments (Deterding et al., 2011), which optimally creates a citizen science level incentive for arbitrary tasks (Chamberlain et al., 2013). The first most prominent representative of this concept is the *ESP* game developed by von Ahn and Dabbish (2004). The authors pursue the idea of exploiting the desire of humans to be entertained (i.e., to play games) by means of a game-with-a-purpose (von Ahn and Dabbish, 2008). The game is meant for labeling/tagging images on the internet in an online multiplayer setting where two players (randomly matched) collaborate to find a common keyword for an image presented to both players in order to proceed and score. To increase the number of keywords and thus yield more specific descriptions, each image is part of multiple game rounds, where previously created keywords are introduced as taboo words. If only one player is currently online, he is matched with a bot, which is a recording of a previous game session of a real player. The authors emphasize that this concept is suitable for labeling all images on the internet (in 2004) in a matter of weeks, underlining the great potential of gamification in context with crowdsourcing.

Von Ahn et al. (2006a) demonstrated that even more detailed descriptions can be achieved when changing the concept from an *output agreement* to an *inversion problem* game concept given by the *Phetch* game. Here, one player describes an image to another player, who is supposed to identify this image among others based on this very description. Further refining the idea of the *ESP* game from what is depicted in an image to where the depicted object is located in the image leads to the *Peekaboom* game (von Ahn et al., 2006b). Again, a two player setting is created, where one player (*Boom*) gradually reveals parts of an image and the other player (*Peek*) is supposed to guess a specific keyword (the one assigned to this image in the *ESP* game). Consequently, by playing *Peekaboom*, the crowd adds object locations (eventually bounding boxes or hinting arrows) to labeled images of the *ESP* game, which are obtained by aggregating results of multiple game sessions. In the same spirit, and inspired by eye-tracking, in the *Bubbles* game (Deng et al., 2013, 2016) players are confronted with a blurred image depicting a bird and are asked to assign it to one of two rather similar bird breeds by de-blurring those spots they believe to be helpful for determining the correct class. To encourage focusing on most expressive parts of the image, de-blurring causes a penalty.

However, creating an enjoyable game can be a time-consuming and complex endeavor (Prandi et al., 2015). Also, Thaler et al. (2012) identified the challenge to design a long-term motivating game that involves performing tedious tasks such as labeling. Game creators need to decide which gaming elements should be included and to what degree these measures contribute to the will of users to play it *just one more time*, as von Ahn and Dabbish (2008) formulated (Sailer et al., 2013, 2017). Among the most favored elements are earning points, time limits and leader boards (Morschheuser et al., 2017), which have proven to be important to many members of online communities (Lampel and Bhalla, 2007). The significance of a proper choice of such elements can be demonstrated by means of leader boards. An all-time leaderboard indeed boosts motivation of most frequent players, but at the same time, significantly demotivates occasional players never able to reach top positions, which can be avoided by using short-term leader boards (Chamberlain et al., 2013; Morschheuser et al., 2017). The authors also argue that rich game elements could soon cause frequent players to lose interest in the game due to the redundancy and thus boredom of such elements.

Since playing on familiar maps, such as those of one's hometown, can often lead to an even more immersive and fun experience, gamification is frequently realized in terms of location-based games (Leorke, 2019) within VGI projects. Examples can be found for creating accessibility maps by means of the *Geo-Zombie* game (Salomoni et al., 2015; Prandi et al., 2015), for collecting indoor data (Martella et al., 2015) or points of interest such as building entrances with the *GeoSnake* game (Matyas et al., 2011), or for point-based in-situ land cover and land use maps (Bayas et al., 2016). But fun alone is often not a sufficient incentive to lure a large enough number of workers to contribute more than occasionally, which is why physical or monetary awards are often granted (Salomoni et al., 2015). This especially holds true if a location-based game is not realizable and for tedious labeling tasks. Therefore, Salk et al. (2015) grant physical awards for top leaderboard ranks in their *Cropland Capture* game, similar to the strategy of Bayas et al. (2016).

2.3.2 Paid Crowdsourcing

Although gamification might attract some crowdworkers, it is unlikely to motivate enough participants for massive labeling campaigns (Kittur et al., 2013), often required for generating training data for ML. The willingness to contribute might also be influenced by the nature of the task and even the identity of the proposer of the task (Horton and Chilton, 2010). For instance, when a private commercial employer launches an inherently tedious labeling campaign, it is likely to be doomed to fail due to lack of contributions. As a remedy to such situations, the concept of paid crowdsourcing (cf. Figure 2.1) was developed, which promises easy access and motivation of crowdworkers through payment, and where tasks are typically quickly completed. The most prominent realization of this idea is *Amazon Mechanical Turk (MTurk)*, which can be considered to be the mediator between an employer and the crowd (i.e., registered users), and minimizes the technical overhead of the employer (Chandler et al., 2013).

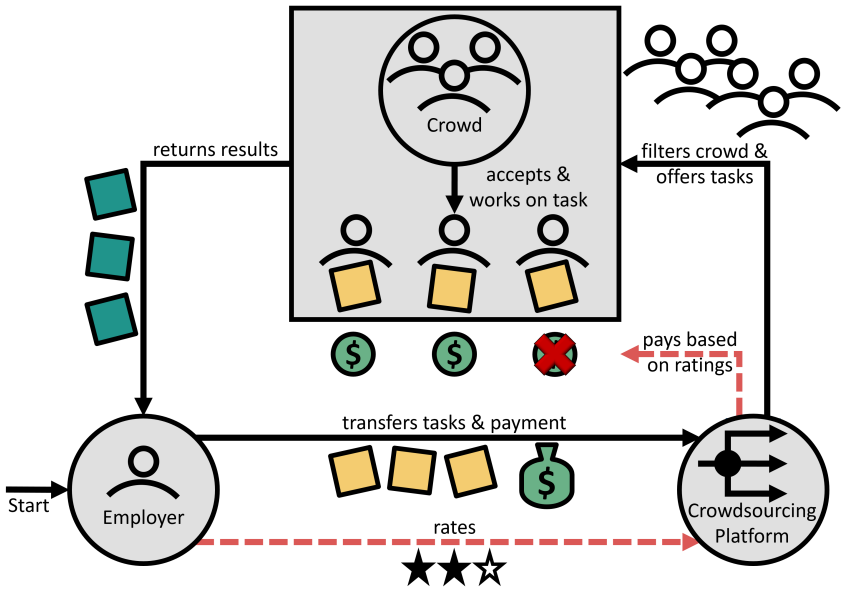


Figure 2.1: General concept of paid crowdsourcing. An employer submits tasks to be completed to a respective paid crowdsourcing platform along with the payment for the complete campaign. By relying on the platform, the employer can decide to offer the task to a specific group of workers only, who will then return their results to the employer. After data acquisition, the employer is given the opportunity to rate submitted tasks (*content*/*not content*), thus prompting or denying payment through the crowdsourcing platform (indicated by *dashed red arrows*).

Using the platform, the employer posts campaigns as an open invitation (i.e., crowdworkers decide whether they would participate) and defines necessary parameters, such as the amount of payment, and possibly limits the group of targeted persons (e.g., living in a specific country or having a certain reputation score). Although campaigns are usually quickly completed as a whole, submission of results often shows a long-tail behavior (Ipeirotis, 2010b; Wang et al., 2011), where most work is completed by only a few different annotators (Sorokin and Forsyth, 2008). After submission of processed tasks, the employer prompts payment for the workers through the platform if satisfied (cf. Figure 2.1), and may award a bonus (depending on quality) (Geiger et al., 2011). The remuneration of workers typically amounts to a few cents per task, e.g., Mao et al. (2013) recommend \$0.10 , while others suggest estimating the required time for completing a task and paying U.S. minimum wage (Salehi et al., 2015). According to Haralabopoulos et al. (2019), the median payment on *MTurk* is \$1.38 /h. Although sometimes criticism arises around the payment rate, clear advantages for crowdworkers are the flexibility that they can perform interesting tasks at any time and that they can make use of spare time (Chandler et al., 2013), e.g. while riding the train. Please note that a discussion of ethics of paid crowdwork is beyond the scope of this work and can be found in Kittur et al. (2013), for instance.

With the payment being the main incentive of crowdworkers, its influence on performance is in focus. Horton and Chilton (2010) found that a higher payment does not lead to a better level of quality as one would hope, but on the contrary, lower wages might even be preferable for non-attractive tasks (Mason and Watts, 2009). This is to encourage that only interested workers participate, for whom payment is not the main factor, and to not attract inefficient workers focusing solely on maximizing their income (Sorokin and Forsyth, 2008). However, a higher payment usually corresponds to more work being done and results in faster completion of a campaign (Mason and Watts, 2009; Walter and Soergel, 2018). Generally, Ye et al. (2017) observed that crowdworkers perform best when they believe that they are paid fairly. Furthermore, crowdworkers show different behavior depending on the applied compensation model, like payment based on the number of annotations or based on time spent, and found a time-based model to yield a better quality of

results (Mao et al., 2013). Additionally, crowdworkers react both to positive extra incentives such as granting boni (Harris, 2011), which need to be sufficiently high to be motivating (Ho et al., 2015), and to negative incentives, e.g., denying boni or payment at all and thus harming reputation score (Shaw et al., 2011). For the latter, Shaw et al. (2011) and Suri et al. (2011) found that crowdworkers pay special attention whenever they are informed about quality measures, but without disclosing the assessment. For evaluating whether boni are granted, the agreement with results with other workers (Radanovic et al., 2016; Raykar et al., 2010) or with a *gold standard* might be used. Also, other crowdworkers can be asked to grade submissions in order to decide on payment (Sorokin and Forsyth, 2008). Since not only the payment but also the social background of crowdworkers might influence results, Ipeirotis (2010a) and Hitlin et al. (2016) analyzed the demographics of *MTurk*. Interestingly, Haralabopoulos et al. (2019) have demonstrated that crowdworkers with a low income contribute results of low quality at a higher pace than workers having a higher income.

Compared to volunteered crowdsourcing projects, usage of paid crowdworking amplifies the importance of quality control, since although paid crowdworkers try to reach or maintain a reasonable reputation score (Mao et al., 2013), they are often less reliable than volunteered crowdworkers (Redi and Povoa, 2014). Even worse, thanks to the general anonymity of online crowdsourcing marketplaces (Chandler et al., 2013), just creating a new account might be a viable solution when having a low score (Suri et al., 2011). Respective crowdworkers are often referred to as *satisficers* in the paid crowdsourcing scene who only show minimum effort (Chandler et al., 2013; Marcus and Parameswaran, 2015) while trying to maximize their income (Gingold et al., 2012). For instance, Kittur et al. (2008) found that up to 30 % of submissions might be of low quality. Hence, the aim of some crowdworkers may be primarily to "game the system" (von Ahn et al., 2006b; Kittur et al., 2008). This tendency of crowdworkers to be dishonest whenever this leads to higher payment is also observed by Suri et al. (2011) and occurs especially when eligibility questions for highly paid tasks are in place (Chandler and Paolacci, 2017; Sharpe et al., 2017).

Paid Crowdsourcing & Gamification

Although seeming counterintuitive due to extrinsic motivation often overriding and diminishing intrinsic motivation (Frey and Jegen, 2001), the concept of paid crowdsourcing is sometimes paired with gamification (e.g., Deng et al., 2013), actually supposed to be motivation enough. Eickhoff et al. (2012) distinguish between i) purely *money-driven* and ii) *entertainment-driven* workers seeking diversion, for whom monetary remuneration is not the primary incentive, and argue that gamification appeals mainly to the second group of workers. The rationale of enriching paid crowdsourcing by gamification is to lure workers to conduct extra free work (thus minimizing costs) and to also (optimally) increase quality of results through a higher level of commitment. For instance, Eickhoff et al. (2012) posted the same task with and without gamification elements and observed an increase in quality and a lower number of malicious contributions by adding such elements, while at the same time reducing costs. Lichtenberg et al. (2020) identified progress bars, badges, and leader boards as proper means for increasing the output of paid crowdworkers, exemplified for intentionally boring slider tasks. Although Feyisetan et al. (2015) report improvement in both quality and quantity for an image labeling task (especially through introducing leader boards), the authors raise the concern that more demanding and/or time-intensive tasks might not profit from gamification. Chamberlain et al. (2013) caution that especially in (paid) gaming environments, malicious workers might try to cheat the game to receive their money without completing (or sloppy completing) assigned tasks.

Paid Crowdsourcing Platforms

Despite the aforementioned challenges for quality control, subsequent founding of various platforms in the image of *MTurk* emphasizes the current success of crowdsourced work. For instance, *microWorkers (MW)* (Hirth et al., 2011) is structured quite similar to *MTurk*, mainly differing in the demographics of crowdworkers. *Appen*[5] provides to take over the complete data collection and annotation process according to customers'

[5] `https://appen.com/`

needs. Other representatives focusing on data annotation are *Clickworker*[6] and *ShortTask*[7] (please note that especially the latter platforms often change name and owner; spring 2023 status).

In times of ML and particularly Deep Learning (DL), the success of these platforms is largely boosted by the great need for high-quality labeled training data (Vaughan, 2018). The higher importance of high-grade and sufficient training data over the choice of the ML model was already formulated in 1997 by Ho and Baird (1997). Stork (1999) argues that even a straightforward Nearest Neighbor ML model can achieve satisfactory results if enough training data is available. Indeed, *MTurk* is especially used to satisfy the demand for labeling (Ipeirotis, 2010b; Hitlin et al., 2016), transcription tasks (Snow et al., 2008; Gao et al., 2015) and content generation (Hitlin et al., 2016) - adopted both by industry and research (Marcus and Parameswaran, 2015), with industry revenue exceeding that of research (Hitlin et al., 2016).

Briefly after its release, *MTurk* was embraced by the CV community for cheap and quick annotation tasks (Sorokin and Forsyth, 2008; Chandler et al., 2013) to foster automatic systems by providing "big manually labeled data" (Kovashka et al., 2016). Soon, paid crowdsourcing was employed for creating various benchmarks and data bases, and respective workflows were developed for different vision tasks. The most prominent representative is the *ImageNet* data set (Deng et al., 2009; Russakovsky et al., 2015a), which provides typical images to populate the hierarchical *WordNet* structure (i.e., image-level categories/classes). *ImageNet* was constructed by first crawling the web for respective images of different categories, which were then presented to the crowd for verification. Images that obtained category-dependent convincing majorities of crowd votes were included in the data set. Other data bases generated utilizing crowdsourcing were discussed by Zhou et al. (2014), Kumar et al. (2009), Bell et al. (2013), Bell et al. (2015), Welinder et al. (2010a), Chen et al. (2009), Su et al. (2012) and Russakovsky et al. (2015a).

2.4 Crowdsourcing as Part of Hybrid Intelligence Systems

Although data labeling is indispensable for the development of modern ML methods and already sufficiently justifies the existence of paid crowdsourcing platforms (Vaughan, 2018) (and which is also the main application in industry according to Marcus and Parameswaran, 2015), they have a more general claim. As stated by Jeff Bezos such platforms are tailored for all kinds of tasks that are easy to solve for humans but highly demanding for machines: *"Normally, a human makes a request of a computer, and the computer does the computation of the task. But **artificial** artificial intelligences like Mechanical Turk invert all that. The computer has a task that is easy for a human but extraordinarily hard for the computer. So instead of calling a computer service to perform the function, it calls a human."* (Bezos, 2007).

The vision formulated herein is that machines and humans work collaboratively (Allahbakhsh et al., 2013). This leads to a concept often being pursued but differently referred to as *human-in-the-loop* system (Branson et al., 2010), *interactive machine learning* (von Ahn and Dabbish, 2008) or *hybrid intelligence* system (Vaughan, 2018). They all describe algorithms that operate under the same assumption of subroutines being not accomplished by machines (i.e., *electronic processors*) but *human processors* conducting *human computation* (Chandler et al., 2013) or *human cloud computing* (Hoßfeld et al., 2011) - ***artificial** artificial intelligence* (Gingold et al., 2012; Kittur et al., 2011; Kittur et al., 2013; Marcus and Parameswaran, 2015). Each party accomplishes those parts of the processing chain in which it is at its best (e.g., human interpretation capabilities vs. automatic performing of repetitive, easy-to-formulate tasks). This allows the creation of systems outperforming respective system designs relying on only one party (Kittur et al., 2013; Kamar, 2016; Vaughan, 2018). Exemplary implementations of this concept were realized by Russakovsky et al. (2015b), Kamar et al. (2012), Branson et al. (2010), Spiro et al. (2010), Vondrick et al. (2012), Wah et al. (2011), Wah et al. (2014) and Jain and Grauman (2013). For a better interpretation of this approach,

[6] `https://www.clickworker.com/`
[7] `http://www.shorttask.com/`

consider a semi-automated driving system where most navigation is accomplished by the autopilot, but in some complex situations, the human driver is asked to take over (Kamar, 2016).

For realizing such systems, paid crowdsourcing platforms are paramount in order to have enough crowdworkers (i.e., processors) on demand to achieve scalability. This guarantees timely results by parallel processing leveraging a geographically distributed workforce (Kittur et al., 2013), which requires proper splitting and assigning of tasks (Kittur et al., 2011). However, if humans (or "semi-qualified workers") with possibly no feeling of what errors are harmful to ML systems (Endres et al., 2010) become part of an algorithm, the outcome is no longer deterministic. It depends on a multitude of additional factors, such as incentive, motivation, understanding, ability, general treatment, and complexity of the task (Ye et al., 2017; Endres et al., 2010). These factors were previously only a subject of crowdsourcing research (cf. Section 2.2 & 2.3), but now become part of the system design and must be properly addressed by an operator (Kittur et al., 2013). Additionally, the tasks and respective acquisition tools need to be designed with special emphasis on usability for crowdworkers (Kittur et al., 2008) to become part of the system as well, which is an often overlooked topic (Sorokin and Forsyth, 2008; Kovashka et al., 2016).

In such hybrid intelligence systems, machines guide humans (Kittur et al., 2013) or even "use" them, instead of the other way around (Gingold et al., 2012). In context of ML, the idea of humans teaching machines or acting as "helpers of Artificial Intelligence (AI) systems" (Kamar, 2016) might lead to them replacing their trainers, and hence the demise of the human component (Kittur et al., 2013). Thus, human effort would not only be reduced by machines, but the human component could be replaced at all (Branson et al., 2010). Chandler et al. (2013) raise the question of paid crowdsourcing being a temporal means to build ML models. Also, von Ahn et al. (2006b) envisioned the same future for the *Peekaboom* game, where gamification is especially seen as a tool for creating training sets for AI systems (Morschheuser et al., 2017).

While paid crowdsourcing systems and ML models driven by or working in tandem with crowdsourced data are commonplace in the CV community, in the geospatial domain such techniques are still in their infancy. Nevertheless, ongoing research already focuses on establishing a powerful human component from which a respective machine component could learn. For instance, Li and Zipf (2022) propose to generate training data for respective ML models implicitly by harnessing the *OSM* data base. However, dedicated campaigns to build data for geospatial scene interpretation are only scarcely conducted, while generally, landcover classification, for instance, is well suited for crowdsourced annotation when thought of as a labeling problem (Fonte et al., 2017). Most studies in this regard deal with employing a paid (Walter and Soergel, 2018) or unpaid crowd for annotating and interpreting aerial images (Salk et al., 2015; Estes et al., 2016; Juni and Eckstein, 2017) or street view scenes (Hara et al., 2015; Hecht et al., 2018; Maddalena et al., 2020). But 3D point clouds as the data of interest for this thesis are very rarely considered. To the best of our knowledge, the first work addressing crowdsourced 3D point cloud interpretation was conducted by Herfort et al. (2018) and focused on classification of point cloud subsets into a finite set of categories and tree height estimation by means of averaging answers from multiple volunteers. Also concerned with vegetation mapping, Walter et al. (2020) launch paid crowdsourcing campaigns for object detection (trees) in 3D point clouds by means of bounding cylinders and subsequently integrate multiple annotations. Similarly, Walter et al. (2021) discuss crowdsourced 3D bounding box annotation of cars in point cloud subsets, resulting from a preceding campaign where crowdworkers marked points of interest (i.e., cars) in Digital Surface Model (DSM) shadings.

But no matter the data modality, hybrid intelligence systems, where the crowd can be thought of as a sub-routine, were previously not studied for geospatial data interpretation. Additionally, these studies aim at full annotations of respective data sets, but for reasons of cost-efficiency, Kovashka et al. (2016) urge to mind *which* of all available data points to annotate, pointing to the concept of AL.

2.5 Active Learning (AL)

Often hybrid intelligence systems as discussed in the previous section are built upon the concept of AL (Settles, 2009). Apart from linking *human* and *electronic processing units*, the key idea of AL in context of semantic segmentation tasks is to focus labeling effort on only a subset of available instances. In contrast to the typical ML scheme of PL, in AL a classifier is *actively* involved in setting up the training data set. Given an initial (suboptimal) training set, the idea is to utilize the current state of the classifier suitable for the task at hand to iteratively enhance the training data set. Thus, the performance of the classifier, learning from the step-wise increasing training set, is gradually improved. In each iteration step, the classifier derives predictions for all currently unlabeled samples, and those points the classifier is most uncertain in its prediction are labeled and added to the training data set. In this way, the epistemic uncertainty of an ML model is minimized by adding additional information (Gal et al., 2017). Concisely, the goal of AL is to enable a given classifier to achieve top accuracy with as little label effort as possible. In other words, the result of an AL loop is a model-specific (hopefully) optimal training data set, which would generally lead to suboptimal results for another classifier (Crawford et al., 2013).

Foundations of AL were developed in the CV community, with Settles (2009) and Kovashka et al. (2016) giving a comprehensive review of AL literature in this domain. AL can be considered as a system of four main components: i) the base classifier for the task at hand (Section 2.8), ii) most crucially the query function for the selection of instances to be labeled (Section 2.5.1), iii) the label engine or the so-called oracle, e.g., a human operator (Section 2.5.2) and finally iv) an abortion criterion for the iteration (Section 2.5.3). If these aspects are properly addressed, AL allows for cost-efficient solutions to many automatic annotation tasks, also in the field of geospatial data interpretation and, related to that, for the semantic enrichment of 3D point clouds (Section 2.5.4).

2.5.1 Querying Samples in AL

Since it makes a significant contribution to the performance of an AL system, most effort is put into designing an appropriate query function or selection criterion. Most prominent representatives originate from *uncertainty sampling*, *query-by-committee* strategies, or *representativeness sampling*.

Early implementations of *uncertainty sampling* are associated with Support Vector Machines (SVMs) (Cortes and Vapnik, 1995), which derive decisions solely based on support vectors, no matter the size of the provided labeled data set. This means that labeling of the remainder of instances is superfluous, thus also giving an idea as to *why* AL is so effective. Therefore, Ertekin et al. (2007) opt to sample only those points that are closest to the current decision border of the classifier. Generally speaking, *uncertainty sampling* is simply designed to select those samples the current classifier is most unsure about with respect to inter-class similarity (Settles, 2009). To quantify this uncertainty, query functions operate on the posterior probability. More precisely, this can be realized by only considering the probability predicted for the most probable class in *least certainty sampling* (Lewis and Gale, 1994), evaluating the difference between the two most probable classes in *breaking ties sampling* (i.e., "breaking the ties" between those two classes, often referred to as *margin sampling* as well) (Scheffer et al., 2001) or considering all posterior probability values for computing the corresponding *entropy* (Shannon, 1948). With respect to the categorization of such sampling functions based on aleatoric uncertainty or epistemic uncertainty (i.e., uncertainty rooting from inherent randomness vs. from lack of knowledge, cf. Hüllermeier and Waegeman, 2021; Nguyen et al., 2021), in general, uncertainty-based sampling strategies favor aleatoric over epistemic uncertainty since they aim to sample regions close to the decision border where the predictability is lowest (i.e., $p(c_1|\boldsymbol{x}) = p(c_2|\boldsymbol{x}) = 0.5$ in the binary case). However, in AL we are facing a sparse training set, especially in early iteration steps. Thus, the current decision border may be far away from the true perfect one, which is why epistemic uncertainty will also play a significant role. But with an infinite number of iteration steps, purely aleatoric uncertainty will be scored. Please note that specifying the affiliation of a sampling function (e.g., *entropy*) greatly depends on

the method of generating the input posterior probability. If $\boldsymbol{p}(\boldsymbol{c}|\boldsymbol{x})$ is the average of an ensemble classifier (e.g., *deep ensemble*) (Gal et al., 2017), a measure such as *entropy* will also score epistemic uncertainty.

Query-by-committee, on the other hand, is especially designed for reliably estimating epistemic uncertainty, but poses the requirement of having a committee or an ensemble of classifiers. Respective sampling strategies aim to select instances from (so far unrepresented) regions of feature space where the different ensemble members, all trained in accordance with the same data set, disagree. This disagreement is typically evaluated by means of *vote entropy* (Argamon-Engelson and Dagan, 1999), *Kullback-Leibler divergence* (McCallum and Nigam, 1998) or *mutual information* between model predictions and model parameters in context of *Bayesian Active Learning by Disagreement (BALD)* (Houlsby et al., 2011). The latter is realized by sampling points that have a high overall (averaged) entropy *and* for which different models (i.e., different draws of parameters) are confident in their deviating predictions (please note that this strategy can also be considered a hybrid of *uncertainty sampling* and *query-by-committee*).

Representativeness-based sampling strategies focus on querying a subset of points that is as representative as possible for the whole data set. For instance, Sener and Savarese (2018) see AL as a *core-set-selection-problem*. This means that a subset of points is selected from a given feature distribution that is representative of the complete distribution and thus well suited for training a classifier generalizable for the entire data set. This approach can be approximated as a *k-center problem*, where k centers need to be positioned such that the distance between any point and the nearest center point is minimized. Dasgupta and Hsu (2008) suggest first computing a hierarchical clustering of all data points, so that sampling of points can be achieved by gradually following the hierarchical structure, where samples from each node are drawn as long as those nodes in the hierarchy are not pure with respect to the class affiliation of encompassed points. An implementation of the DL era for *representativeness*-based sampling is *VAAL* (Sinha et al., 2019). The idea is to jointly learn a latent feature space from both labeled and unlabeled instances via a variational autoencoder as well as an adversarial discriminator that is trained to distinguish whether a sample is from the labeled or unlabeled data set. Consequently, sampling of points is driven by selecting those new instances that, according to the discriminator, are likely to come from the unlabeled set thus being sufficiently different from already existing labeled samples.

Less common strategies aim to select those points that would change the current model the most or reduce the error or the variation of the model the most (Settles, 2009). This often requires simulating all possible labels for a given instance to estimate the respective impact, which might cause significant computation effort. A prominent representative of this group is the *LAL* framework (Konyushkova et al., 2017), where a regression model uses as input both a description of the state of the current classifier (i.e., the parametrization) and data-related features which describe the current prediction of that point or its relatedness to other data points (but which are different to the specific features like color information of a data point). Based on this combined feature vector, the regressor, which is re-trained in each iteration step exploiting the already labeled data set, would predict whether this point is informative in terms of significantly reducing the error of the model.

The review of Settles (2009) summarizing the aforementioned sampling approaches, however, was written in the pre-DL era, consequently not addressing recommendations for the employment of CNNs as base classifiers. Since CNNs output (pseudo) posterior probabilities, they can theoretically be used as any conventional feature-driven classifier. However, CNNs are known for overestimating their confidence when extrapolating in regions of feature space for which no training samples were available (Gal and Ghahramani, 2016), i.e., they are unaware of epistemic uncertainty. For receiving more sensitive uncertainty measures, the authors propose to approximate *Bayesian* CNNs by *Monte Carlo dropout ensembles*, since true *Bayesian* CNNs are often intractable to be computed, as every parameter is supposed to be sampled from a distribution and consequently inference is done by integrating over all parameters (see the work of Jospin et al. (2022) for a comprehensive overview). Thus, *Monte Carlo dropout ensembles* are a viable approximation and are generated by using dropout not only as regularization in training but also while predicting. This results in an ensemble of slightly deviating classifiers, where the overall posterior probabilities are received by averaging.

Such an output can then be used for any of the aforementioned selection strategies and has proven to yield more efficient AL learning curves compared to using only one single-pass network without *Monte Carlo dropout* (Gal et al., 2017). An alternative approximation of *Bayesian* CNNs are *deep ensembles*. The idea is to train multiple networks with the exact same architecture and same training data, but with different random initialization values of weights. Beluch et al. (2018) and Feng et al. (2019) demonstrated that this approach often outperforms *Monte Carlo dropout ensembles* due to the higher capacity and greater independence of ensemble members, which, however, comes with higher computational cost for training multiple networks. A comprehensive review of both challenges and solutions of Deep AL (AL built around a CNN) were discussed by Ren et al. (2022).

Apart from scoring samples, different sampling heuristics/scenarios with respect to scope, such as *membership query synthesis* (Angluin, 1988), *stream-based* AL (Atlas et al., 1989) and *pool-based* AL (Lewis and Gale, 1994) are suggested, with the latter two being most appropriate whenever unlabeled data is easy and abundant to generate. *Stream-based* AL can help to reduce computation effort since each sample in the unlabeled data set is visited only once and is either added to the training set or irrevocably discarded depending on the current classifier. But informative samples might get lost due to the decision of a non-optimal state of the classifier. *Pool-based* AL relaxes this issue by evaluating the complete unlabeled training set in each iteration step.

However, no matter the heuristic, all sampling strategies are designed to query points based on the current state of a classifier. Since any additional training sample might change the current belief in the model, these methods work best when one instance per iteration step is selected. However, this results in an ML model being re-trained each time after only one sample has been added. This can be considered both statistically unreasonable and computationally intractable, especially when CNNs are used as base classifiers (Sener and Savarese, 2018; Kirsch et al., 2019). Therefore, often a larger batch of points is sampled in each iteration step. On the other hand, this poses the issue that multiple samples with the highest sampling score are often too similar to each other and equals sampling quasi-duplicates, which eventually results in wasting labeling resources for non-most-informative instances. Therefore, measures that yield a diverse batch of informative samples are often applied. Such diversity can be achieved, for instance, by score-weighted *k-means* clustering of feature space, where k equals the number of points to be sampled (Zhdanov, 2019; Ash et al., 2019; Prabhu et al., 2021). Also, well-studied sampling strategies such as *BALD* can be adapted for the batch-mode setting, where multiple points are jointly sampled based on the mutual information between a batch of multiple data points and model parameters (Kirsch et al., 2019). However, this might lead to high computational effort even with the originally proposed approximation by a greedy algorithm (Thoreau et al., 2022).

2.5.2 Oracles in AL

Apart from the question of which data points are most suitable for the ML process, special emphasis needs to be put on the oracle as well when designing such an AL scheme. Often, a GT oracle is assumed, i.e., an oracle which always returns perfectly correct answers to the label queries. Although this can be purposeful for assessing the theoretical performance of simulated AL runs, it is unrealistic to obtain such results when working with humans, and especially when working with crowdworkers, where results might contain both random and systematic labeling errors (Chandler et al., 2013; Lockhart et al., 2020). This issue is also subject of study in the crowdsourcing research (cf. Section 2.2), but can be addressed in the AL scenario as well with possible solutions revolving around efforts for filtering crowdworkers based on estimating individual worker scores (Donmez et al., 2009; Welinder and Perona, 2010; Long et al., 2013; Long and Hua, 2015) or assigning them tasks suitable to their skills (Zhang and Chaudhuri, 2015).

Nevertheless, it must be accepted that there may be samples that are more difficult and error-prone to label than others, and that are also more costly because they need to be assigned to multiple crowdworkers to reach a consensus (Deng et al., 2009). An alternative approach is to alleviate this issue by modifying the sampling strategy in such a manner that, at the cost of informativeness for the machine, a sample is selected

that is still as informative as possible but also feasible to be labeled by the human oracle. For instance, Mackowiak et al. (2018) fuse a common AL informativeness measure with a learned estimate of labeling effort (measured by the number of clicks) into the final score function. Similarly, Vijayanarasimhan and Grauman (2009) train a SVM classifier for predicting labeling cost based on tuples of image features and annotation times obtained from real *MTurk* crowdworkers, and use this information to decide for new images what type of annotation (full, part segmentation, or image classification) should be performed minding both labeling effort and informativeness.

When machine and human oracle are properly combined, human-in-the-loop systems like the one discussed by Vijayanarasimhan and Grauman (2011) can be powered, where an expert only needs to give some samples for a specific class in order to prompt a fully automated pipeline (with human crowd processors). Related images are then crawled online, and from those, the most informative ones are actively queried, multiply labeled by crowdworkers on *MTurk*, and used to refine the classifier within the AL loop. Please note that although humans are involved in the form of *human processing units*, from the view of an operator, such a system can indeed be considered to run in a fully automated manner. Although such impressive learning schemes have already been realized in the CV community for everyday life scenes (where crawling candidates is an easy task compared to crawling, e.g., ALS point clouds), to the best of our knowledge no comparable work implementing a fully automatized human-in-the-loop system with a paid crowd oracle, is present in the geospatial domain.

2.5.3 Terminating AL Loops

For terminating an AL iteration, a proper stopping criterion needs to be defined to guarantee that the AL loop runs long enough to gain stable performance, but does not conduct more iteration steps than necessary to minimize costs (Settles, 2009). Naturally, the easiest way is to measure the performance on a large representative test set, but this defeats the purpose of AL to require labeling of only a few samples. If only a labeled subset is used instead, this problem is partly mitigated, but also at the risk that the labeled set is not representative of the complete data set. Thus, termination of the loop should be computable *without* usage of labeled instances. A solution could be to check the similarity of newly queried samples with the ones already included in the training set (Vlachos, 2008). In the same spirit, a SVM-specific stopping criterion would be to abort the iteration as soon as no support vectors are available anymore, which can be measured by the fact that newly sampled points are further away from the decision boundary than any of the available support vectors (Ertekin et al., 2007). Other approaches rather measure the stability of the predictions of a current classifier on a large unlabeled data set in terms of confidence of the classifier (Vlachos, 2008) or agreement between current and previous predictions (Bloodgood and Vijay-Shanker, 2009). If an ensemble classifier is employed, stopping can also be based on comparing the classification disagreement of the ensemble members for the remaining unlabeled pool vs. an independent and unlabeled validation set (Olsson and Tomanek, 2009). The loop can be stopped as soon as the classifier is properly adapted to its data, i.e., as soon as the agreement on the remaining unlabeled pool exceeds the one of the validation set.

2.5.4 AL in Remote Sensing and Semantic 3D Point Cloud Segmentation

The human-in-the-loop approach of AL has also been investigated in the remote sensing community, but primarily engineered to minimize labeling effort such as visual inspection of the data or even field surveys. The main emphasis lies on semantic segmentation of aerial imagery by applying the well-known AL concepts as discussed before (see the comprehensive reviews by Tuia et al. (2011b) and Crawford et al. (2013)). Dealing with spatially meaningful data, however, poses new challenges, but also opportunities, as summarized by Crawford et al. (2013). For one, the need to modify conventional sample selection strategies arises if field surveys are to be conducted for labeling, where the selection of samples (i.e., locations) should be such that the traveling distance between queried locations is also minimized. On the other hand, spatial information can be utilized to diversify selection of most informative samples. When assuming that samples that are

close with respect to their position in object space carry similar redundant information, sampling should focus on instances that are more distant from each other. A recent comparison of sampling strategies for a state-of-the-art CNN classifier was conducted by Thoreau et al. (2022). Notably, the classic *breaking ties sampling* strategy still leads to state-of-the-art results, both in terms of fast convergence on a level of high classification accuracy and in terms of the capability to discover new classes in the data set.

Among the AL research in remote sensing, the work of Tuia and Munoz-Mari (2013) is noteworthy, as to the best of our knowledge this is the only one acknowledging an imperfect oracle and tailoring the query of points to its needs. Since AL sampling strategies tend to select instances on class borders, inhomogeneous spots, or shadowed areas, interpretation is often demanding not only for lay workers but also for experts. Therefore, the authors suggest training a classifier (apart from the one for conducting the actual task of semantic segmentation) that distinguishes between samples that are feasible for labeling and so-called bad states the oracle is not capable to label. Thus, such instances should be skipped automatically to avoid tiresome manual skipping or incorrect labeling by the human oracle if presented anyway. The training data for this second classifier is generated on the fly whenever a user rejects to label a presented instance. Eventually, only samples queried by the AL engine to be informative *and* supposed to be feasible for labeling are presented to the oracle.

Although applying AL for semantic segmentation of 2D images has been extensively investigated, utilizing AL for the semantic segmentation of 3D point clouds is only tackled scarcely, especially for ALS point clouds and is still matter of ongoing research. Thus, as formulated by Settles (2009), now research in AL focuses on applying it in practice.

To the best of our knowledge, the first work addressing this issue is the pipeline presented by Luo et al. (2018) for the semantic segmentation of high-resolution Mobile Laser Scanning (MLS) point clouds. To reduce the amount of required data, the authors suggest initializing their pipeline by transferring the point cloud representation to a supervoxel representation, and opt to use a pair-wise Conditional Random Field (CRF) model built upon a Random Forest (RF) classifier for inference. The selection of points follows the idea of especially emphasizing instances that are misclassified by the current classifier and are thus assumed to be particularly informative. The pool of samples subject to labeling is thus composed of those pair-wise supervoxels with different inferred labels (but only if the currently most underrepresented class in the current training set is involved), where the difference is seen as hint for misclassification. The selection is then conducted by a modified *breaking ties* approach where the most uncertain supervoxel is queried for labeling and afterwards adjacent supervoxels are checked for consistency with the now labeled supervoxel to also detect and prompt possibly misclassified samples for labeling.

Mainly for the classification of terrestrial point clouds (indoor and MLS point cloud data), also CNN-based approaches were proposed (Wu et al., 2021; Shi et al., 2021; Shao et al., 2022). They follow a similar pattern of first computing superpoint regions (clusters of adjacent points with similar features) as AL units instead of single points, and they naturally differ in the selection strategy. Wu et al. (2021) compute an information score based on standard *entropy* uncertainty enriched with scoring color discontinuity and structural complexity (i.e., roughness) of regions to focus on most informative inhomogeneous regions. The diversity within batches is guaranteed by measuring similarity of regions in feature space in order to enable penalizing occurrence of similar regions in the batch. Both addressing uncertainty and diversity is also the goal of Shi et al. (2021). This is achieved by combining uncertainty measured via *entropy* with a diversity measure based on the minimum feature difference between a candidate region and all previously labeled superpoints. Shao et al. (2022) recognize that although generating superpoints is beneficial for computational efficiency, it poses the risk of grouping points from different true classes and causes mislabeling of individual points when per-group labels are assigned. Thus, the authors account for this issue when computing the overall uncertainty of regions by only considering the points that are predicted to belong to the same class as the predicted majority label of that region. Top regions are then used to form a graph where nodes represent superpoints and edge weights are based on both location difference and structure difference (computed via chamfer distance). To receive a diverse batch of regions, farthest point sampling can then be applied to a so-called diversity space, which is

obtained by an edge-weighted aggregation of superpoint features with neighboring superpoint features. For the sake of completeness, another AL approach operating on MLS scans worth to be mentioned, is proposed by Feng et al. (2019), but is designed for object detection also exploiting *Monte Carlo dropout ensembles* and *deep ensembles* for *entropy*-based uncertainty estimation.

As already pointed out, for semantic segmentation of ALS point clouds, AL approaches are scarce. Hui et al. (2019) consider the generation of a Digital Terrain Model (DTM), i.e., the filtering of ground points, as a classification problem. Based on an initial training set generated from multi-scale morphological filters, the authors employ a SVM for this filtering step, where AL is used to iteratively refine the filtering and thus the resulting terrain model. In each iteration step, a sigmoid function scores the distance of points to the current DTM level. Points predicted as *Ground* and having the lowest sigmoid distance scores are added to the *Ground* samples in the training set, and vice versa, points with the greatest distances and a *Non-Ground* prediction are added to *Non-Ground* samples, thus using a kind of automatic oracle. A similar understanding of an oracle can be found in the work of Li and Pfeifer (2019), who design a semi-supervised AL pipeline for 3D point cloud classification where labels of an initially provided (suboptimal) training data set are propagated each to the point in an optimal neighborhood that incorporates the highest *breaking ties* score to gradually improve an RF classifier.

Supervised AL for large-scale point cloud classification is conducted by Lin et al. (2020b,a). The authors consider a regularly tiled point cloud as input and utilize AL for the minimization of required training data by means of only selecting most informative tiles. They opt for tiles over individual points as AL query units mainly for reasons of computational efficiency because when sampling individual points, neighboring points would have to be presented to a CNN-based classifier anyways to infer meaningful features - although not carrying any label. This means that if labeled AL points are scattered over the complete point cloud, the CNN would need to process roughly the same amount of points as for a PL approach. But in case of restricting sampling to a tile level, points to be processed can be limited to those included in the queried tiles. Also, the authors expect that selected tiles receive full point-wise labeling, following the same reasoning concerning computational effort, as complete tiles are induced into their CNN anyways. However, from an economic point of view labeling is much more costly than computation time and thus such an annotation scheme does not fully leverage the potential of AL to reduce costs by minimizing required labels. Nevertheless, labeling effort is still reduced, as many tiles can be discarded. Since tiles are scored by averaging scores of individual instances, this poses the risk of ignoring tiles where a part of the tile would provide useful information to the classifier and where remaining points have a low sampling score. Precisely, for sampling, both classic point-wise *entropy* and segment-wise *entropy* are utilized. The latter is computed as predicted pureness of points that fall within a segment generated from an unsupervised segmentation where inhomogeneity is assumed to correspond with uncertainty. To also measure epistemic uncertainty, *mutual information* based on a *Monte Carlo dropout ensemble* is evaluated. To further boost the performance of respective AL runs, the authors recommend training their *PointNet++* implementation in an incremental manner, i.e., avoid costly training from scratch in each iteration step.

2.6 Transfer Learning

Just like AL, Transfer Learning (TL) (Pan and Yang, 2010) is intended to serve the same purpose of learning under scarcity of labeled data. The transfer is more than mere generalizability of trained models among different data sets. TL is supposed to explicitly transfer information that is already available (i.e., learned) in a specific model so that it can successfully operate in other scenarios as well, where limited or no labeled data is present (Persello and Bruzzone, 2012). Such scenarios typically comprise performing a different task than the one the model was originally designed for (e.g., classification of land cover types instead of dog breeds), or the same task the model was dedicated to is to be performed in another domain. The first concept can be a viable way to minimize labeling effort, e.g., by trying to reuse the feature descriptor generated by a model that was originally designed for another task and which was trained by typical CV

image data bases. However, in remote sensing, where we are dealing with data that is significantly different from such representing everyday life scenes (e.g., aerial images vs. close-range snapshots of CV data bases), this approach might be limited (Penatti et al., 2015).

The latter concept, Domain Adaptation (DA), on the other hand, is a hot research topic in the field of remote sensing data analysis due to massive amounts of data being acquired at an ever-increasing pace, but is also prominent in the CV community (Patel et al., 2015). Since it is prohibitively expensive to manually set labels for every new data set, the greater goal is to employ labeled data from one domain, the source domain, to derive reasonable predictions in another domain, the target domain. In case of remote sensing data, typical tasks of that nature are to apply a classifier trained for one epoch to another one depicting the exact same location (Rajan et al., 2008; Demir et al., 2013), or to employ an existing classifier for another area captured from the same sensor (Jun and Ghosh, 2008; Rajan et al., 2008; Tuia et al., 2011a; Persello and Bruzzone, 2012; Paris and Bruzzone, 2018). However, the success of DA depends on the similarity between the source and target domain or, vice versa, the inherent domain gap (Quiñonero-Candela et al., 2008) between the two domains, which can have various reasons.

Formally, we distinguish between two different domain gap problems: i) the *covariate shift* problem (Shimodaira, 2000), which can arise in cases the source domain has been inadequately sampled, either due to a biased selection of training points in a PL setting by a human operator or due to insufficient feature space exploration in an AL scenario, so that the training set is not representative for the target domain. ii) its more general case, *sample selection bias* (Heckman, 1979) assumes that, additionally, the representation of the same classes in feature space is different in the two domains. The latter is the DA problem we expect to face in remote sensing data analysis (Persello and Bruzzone, 2012; Tuia et al., 2016). It can root from various effects, such as differences in the sensor configuration (sensor and/or flight planning), atmospheric conditions (e.g., humidity, cloudiness) and illumination, as well as from changes in the scene itself due to phenological phenomena (i.e., seasonal effects). Thus, bridging the domain gap by performing DA has been studied widely (Tuia et al., 2016; Crawford et al., 2013), with solutions coming from both the semi-supervised and supervised fields. These methods often focus on generating invariant or joint features or aim to adapt the classifier itself. While semi-supervised methods are designed for coping with situations where labels are only available in a source domain, in the supervised field we assume that it is possible to obtain some (few) extra labels in the target domain. Thus, supervised DA methods often outperform representatives of the semi-supervised field, at least in cases of significant domain shifts (Rajan et al., 2008), since respective methods typically expect the same classes in both domains and a sufficient correlation of the underlying statistical distributions (Demir et al., 2013). Thus, if a (limited) budget is available for retrieving true labels of the target domain, DA of the classifier can be accomplished by including this additional training data.

For this, the PL approach *TrAdaBoost* (*Transfer AdaBoost*) is presented by Dai et al. (2007), which deals with the scenario of having many labeled source domain samples and (few) labeled target domain samples. However, it is unclear which source domain samples are actually helpful for the classification of the target domain and follow the same distribution as the target domain samples. The authors propose to automatically minimize the impact of conflicting source domain samples by leveraging a boosted learning scheme to automatically decrease the weights of those source domain samples that get misclassified by the current classifier. Since labeled target domain samples are used within this method, the question arises of how to select target domain samples that are to be labeled to yield top results. This task is closely related to AL, thus often suggested in literature as viable solution, which will be referred to as ATL in the following.

2.7 Active Transfer Learning (ATL)

Rajan et al. (2008) proposed a first framework to actively sample points in the target domain applying the *KL-max* selection criterion. The idea is to select those points that would change the current posterior probabilities the most, which is achieved by testing all labels from a specific class catalog for their respective impact. Please note that assigning a wrong class label often leads to most prominent changes, which is why,

based on the current prediction, the probability that this is the correct class is also considered. Rai et al. (2010) argue that gradually adapting a classifier trained on source samples by means of AL can be further boosted if target domain samples are queried that are as different as possible to source domain samples. This is achieved by deriving a distribution separator hypothesis distinguishing between source and target domains solely based on unlabeled samples from both domains, so that the subsequent AL loop is only allowed to draw points from the target site of the separator hypothesis, thus avoiding samples from the target domain that appear too similar to the source domain, as those would lie on the source site.

However, Tuia et al. (2011a) argue that performing basic AL can be risky whenever the assumption of independent and identically distributed data is severely violated, which can very well be true for remote sensing data under domain shifts. In this case, most informative target samples are not necessarily situated near to class borders of a classifier trained for another domain where samples are commonly drawn from. Following the same reasoning, adaption of a classifier to a scenario with another (enhanced) class catalog might also be limited. Although representatives of new classes might still be sampled in the course of the loop, a potentially misguided focus of the loop could (at least) lead to a delay in the convergence of the loop. Thus, the authors suggest starting with clustering of feature space of the target domain to sample from these clusters a quantity of points proportional to the size of the clusters. This can be thought of as setting up an initialization data set in the target domain, from which a common AL loop based on the *breaking ties* selection strategy is launched. A solution from the DL domain to the same problem is given by Xie et al. (2022). The authors train energy-based models with source domain data and aim to restrict sampling to those instances from the target domain that are as different as possible from source domain samples. Those samples incorporate highest free energy values, which is measured by logarithmic exponential summation over energy values for all pairs of feature vectors and labels inserted into the estimated energy function. By this, a pool of most dissimilar samples is formed, from which those with maximum uncertainty, measured in terms of an adapted *breaking ties* strategy, are selected.

To make AL more effective for the DA setting, Su et al. (2019) propose to employ the *Domain Adversarial Neural Network DANN* (Ganin et al., 2016), which consists of a feature extractor, a classifier and a domain discriminator. The latter two make use of the feature extractor and are jointly trained in an adversarial manner following the objective of maximum classification accuracy with respect to labeled source domain samples and maximum confusion of the domain discriminator, i.e., a generalized set of features is to be obtained. Active selection within the AL loop can then be achieved based on combining predictive uncertainty (using the classification model) and dissimilarity to known samples (using the discriminator). Similarly, Zhou et al. (2021) train a domain-invariant feature extractor to align the marginal distributions of source and target domains by adversarial training of a classifier based on labeled instances and a domain discriminator based on estimating *Wasserstein* distance. This model is embedded in an AL loop, again using a combination of uncertainty of the classifier and dissimilarity to the source domain obtained from the domain discriminator for querying in conjunction with a dedicated class weighting scheme.

Instead of trying to derive invariant features, Kellenberger et al. (2019) use a source domain model to predict on the target domain, but consider it as pure feature extractor, so that an explicit mapping between the source domain and target domain distribution can take place by means of *Optimal Transport* to minimize the *Wasserstein* distance. This means that due to the domain gap, the authors mistrust target domain prediction scores and therefore aim to propagate trustworthy source domain prediction scores to feature matched target domain samples that are used for sampling in the target domain. Thus, the original source domain model can be adapted for gaining a more true feature description in the target domain used for the AL-based refinement.

Also, Fu et al. (2021) argue that purely continuing an AL loop on a target domain is suboptimal, exemplified for a committee classifier. Different decision boundaries of the committee members might pass through high-density regions of the target domain since the model is unaware of the presence of respective data points in those regions. Thus, different valid committee members disagree on possibly significantly different target samples, which also leads to a high level of randomness in sample selection. As a remedy, the

authors suggest enforcing predictions to be more consistent with respect to the target domain by including adversarial samples as perturbations of target domain samples whose predictions are to stay the same, which is considered in the employed network's loss computation of the training procedure. Based on the resulting classifier, basic selection strategies relying on *uncertainty* or *query-by-committee* can be defined. To focus more on samples dissimilar to the source domain, but to avoid selection of outliers, the authors train a domain discriminator which operates on the set of features derived by the classifier. This allows the definition of a sampling score combining uncertainty and dissimilarity to the source domain along with incorporating diversity in a batch-mode setting by randomly sampling from instances with top scores. For further boosting batch-mode AL, Prabhu et al. (2021) design an ATL scheme built around a (domain adaptive) CNN classifier and recommend employing CNN features (from the penultimate layer) as basis for uncertainty-weighted *k-means* clustering of feature space, so that one instance per cluster can be queried for guaranteeing diversity. Moreover, the authors claim that a domain discriminator is implicitly encoded in the uncertainty measure and thus considered when sampling points.

With respect to minimizing labeling effort of an oracle, especially tailored for interpretation of multi-temporal imagery, Demir et al. (2013) suggest propagating labels from previous epochs for all pixels that appear to be unchanged (based on their feature vector similarity). They consider pixels to be informative (i.e., supposed to belong to a new class or to a known class with new characteristics) if they, or more precisely, if the cluster they are assigned to during unsupervised clustering is dissimilar to all known classes, causing them to get sampled before launching a common AL run. A related concept of propagating labels from a source domain to actively sampled points of the target domain by means of an explicitly designed transfer classifier is discussed by Shi et al. (2008). It serves the purpose of avoiding turning to a human operator for labeling target domain samples whenever the transfer classifier (trained by both source and target domain samples) confidently predicts the label and when it agrees with an in-domain target domain classifier. Similar to this, Saha et al. (2011) first perform unsupervised domain adaptation to obtain a classifier based on labeled source domain and unlabeled source and target domain data to be employed in an AL setting. The pipeline relies on a hybrid oracle, where a human operator is asked only if uncertainty-queried target domain samples appear different from source domain samples, i.e., if they are positioned on the target side of a determined domain separator hypothesis, which can be learned from unlabeled data of both domains only. Paris and Bruzzone (2018) also aim to label target samples automatically by leveraging information already available. In this approach invariant feature sets (image bands) are identified between domains allowing to automatically propagate labels from a source to a target domain. Only remaining unlabeled target samples compose the pool the AL query function is allowed to draw from.

However, if DA is to be achieved by only enhancing the training pool by adding most informative target samples, there is a risk of keeping source domain samples in the training set that might no longer be representative of the target domain. Thus, such a procedure is merely capable of solving the *covariate shift* problem. But whenever target domain labels can be obtained, the effect of misleading information can be reduced (Jiang and Zhai, 2007). In this spirit Dai et al. (2007) motivate down-weighting of misleading source domain samples in the *TrAdaBoost* framework to be more flexible and to also overcome the *sample selection bias* issue. Matasci et al. (2012) transfer this idea to the AL setting by building a respective AL loop around the *AdaBoost* learning scheme. Precisely, a SVM is allowed to sample points via *breaking ties* in an outer AL loop, and an inner *TrAdaBoost* loop performs just as described by Dai et al. (2007), so that final weights can be reused to commence the next AL iteration step. Jun and Ghosh (2008) also minimize the effect of misleading source domain samples in their *KL-max*-based AL loop by designing an empirical weighting where generally a higher impact and thus weight is granted to misclassified samples of the target domain. The weights of misclassified source domain samples can be gradually reduced as the iteration and thus sampling of new target points proceeds.

To avoid the empirical formulation of weighting rules, Persello and Bruzzone (2012) suggest measuring the disagreement between two classifiers, one being solely trained by source domain samples and the other one being trained by the current training set consisting of both actively *breaking ties*-sampled labels from the

target domain and samples of the source domain. Then, a defined number of most disagreeing predictions with regard to posterior probabilities are eliminated. In the work of Persello (2013), handling of source domain samples is performed slightly differently by only eliminating those samples that are misclassified by the current classifier (trained by source and target domain samples). Remaining source domain samples are individually weighted based on the similarity between source and target domain samples of the same class measured in feature space (please note that similar strategies are also employed in the unsupervised DA domain, e.g., by Paul et al. (2016)). Similarly, Jiang and Zhai (2007) eliminate source domain samples that were incorrectly predicted by a classifier trained by the current combined training set of source and target domain samples, and like Chan and Ng (2007), also opt to reserve a higher weight for target domain samples. Chattopadhyay et al. (2013) propose to perform AL and DA simultaneously in such a manner that the marginal distribution of re-weighted source domain samples, labeled target domain samples, and the currently queried target domain samples is closest to the distribution of unlabeled target domain samples, formulated as an optimization problem.

2.8 Automated 3D Point Cloud Interpretation

Although the basic assumption underlying the AL concept is that a theoretically arbitrary (and even very simple) classifier can be tuned for the task at hand solely by improving (i.e., enhancing) the training pool (Ho and Baird, 1997; Stork, 1999), it still needs to be suited for the intended classification task, i.e., 3D point cloud semantic segmentation. Current concepts for data interpretation can be divided into classical feature-driven and more recent data-driven methods. The latter have seen considerable advances in the last years, but unlike for 2D data (i.e., imagery), where a rather mature level has been established (Zhu et al., 2017; Ma et al., 2019), still a lot of research is put into development and enhancement of different 3D point cloud classification approaches. Hence, we aim to reflect the current state of the art for the two different heuristics.

2.8.1 Feature-Driven Semantic Segmentation

A conventional supervised classification scheme involves two main steps: i) the definition and computation of reasonable features and ii) the training of a proper classifier, where most effort is put into the first step. In context of 3D point cloud classification, a combination of both geometric and radiometric features is often pursued. As for describing the local geometry of each point, usually a point neighborhood (used interchangeably with the term support volume) has to be defined and extracted, often boosted by hierarchical search structures such as *k-d trees* (Bentley, 1975). The vicinity of individual 3D points is typically encoded by a (vertically oriented, unbounded) cylinder (Filin and Pfeifer, 2005) or a sphere (Lee and Schenk, 2002), which can also be expressed in terms of a *k-nn* neighborhood (Linsen and Prautzsch, 2001). All neighborhood types are parametrized by either the radius of the cylinder/sphere or the number of nearest neighbors k respectively. A reasonable choice of this parameter may vary depending on the data set or target classes at hand (Blomley et al., 2014), and could also be chosen dynamically within a given data set, especially in case of varying point density (as it is the case for terrestrial scan data).

Therefore, although being computationally expensive, Lalonde et al. (2005), Demantké et al. (2012) and Weinmann et al. (2014, 2015a,c) suggest defining the neighborhood size for each point independently to enable an optimal most characteristic description of each local surface. The key is to find a compromise between a reasonably large enough support volume to suppress noise while avoiding ambiguous feature values at class borders in case of neighborhoods chosen too large. In order to encode a wider bandwidth of spatial context into the feature vector, often features are evaluated for the same neighborhood type at different radii (Brodu and Lague, 2012; Blomley et al., 2016; Becker et al., 2017) and also for different neighborhood types (Niemeyer et al., 2014; Blomley et al., 2016).

Based on a respective point neighborhood, a typical set of geometric features is computed utilizing the covariance matrix of neighboring points (Pauly et al., 2003; West et al., 2004; Belton and Lichti, 2006; Gross and Thoennessen, 2006; Jutzi and Gross, 2009; Waldhauser et al., 2014). This matrix is often denoted as the structural tensor, which comprises second-order 3D moments (Maas and Vosselman, 1999). By extracting eigenvalues from this tensor (hence dubbed eigenvalue-based features), a set of distinctive features describing local geometry can be employed (Chehata et al., 2009; Kim and Sohn, 2010; Mallet et al., 2011; Weinmann et al., 2013; Blomley et al., 2014; Niemeyer et al., 2014; Weinmann et al., 2014, 2015a; Blomley et al., 2016). In addition to these scale, rotation and shift invariant features (Maas and Vosselman, 1999; Jutzi and Gross, 2009), the local orientation of surfaces with respect to a superordinate reference system, i.e., the *verticality* (Weinmann et al., 2013), or the local *point density* (Weinmann et al., 2013) can be helpful descriptors.

Furthermore, features for capturing characteristic height-related properties, such as *height above ground* or *height variation* within a cylindrical neighborhood, are commonplace (Shapovalov et al., 2010; Blomley et al., 2016). Another family of low-level descriptors for classification are so-called histogram-based features (Osada et al., 2002; Rusu et al., 2008; Tombari et al., 2010; Blomley et al., 2014, 2016; Hackel et al., 2016). These are designed to repeatedly evaluate rudimentary geometric properties based on the relation of points in the local neighborhood definition (e.g., distance between points or angle between point normals), storing them in respective histograms to obtain characteristic signatures. Sometimes 3D points are projected onto either a horizontally or normal oriented plane, allowing additional descriptors such as 2D eigenvalue-based features or accumulation maps indicating high concentration of points along the projection direction (Weinmann et al., 2013, 2015a,c; Guo et al., 2015).

Considering the computational effort for (multi-scale) feature calculation of modern dense point clouds, Brodu and Lague (2012) recommend to derive features only at regularly sampled core points (i.e., for a subsampled point cloud) with subsequent propagation of labels in inference. This formulation is closely related to applying a voxelization of 3D space, where encapsulated points in each voxel cell are used to derive per-voxel features, thus also performing inference on voxel level. This means that predicted voxel labels are propagated to all points within a voxel cell (Kim and Sohn, 2010). A similar approach to reduce the number of instances to classify and to achieve a "smoother" result is to perform an unsupervised segmentation in the first place and predict class labels for complete point segments with additional features describing those segments (Khoshelham et al., 2013; Xu et al., 2014; Vosselman et al., 2017). However, final classification accuracy greatly depends on the quality of the unsupervised segmentation, which is why a point-based/voxel-based approach might be preferable. To especially boost the computation of features at multiple scales, Hackel et al. (2016) argue that the greater the considered support region of feature computation, the more sparsely sampled the underlying point cloud to recover neighbors from may be. Further, they argue that relying on multiple neighborhood scales avoids time-consuming computation of optimal neighborhoods and might be capable of better capturing contextuality even making context-aware classification schemes (see section end) obsolete.

To allow the classifier to distinguish between structures of the same geometric properties but still of different class affiliations, the feature vector can be complemented by radiometric features. These are typically based on waveform processing (usually obtained via gaussian decomposition of received signals or by stochastic modeling (Mallet et al., 2011)) of echo readings of modern full-waveform laser scanning systems, yielding measures such as *amplitude*, *echo width* and *echo shape* (Mallet et al., 2008; Schmidt et al., 2014), or simply the *echo ratio* for each point (Chehata et al., 2009; Mallet et al., 2011). Although Red, Green, Blue (RGB) color information based on an orthogonal projection of orthophoto colors to Light Detection and Ranging (LiDAR) points was also applied in the past for extending the feature vector (Charaniya et al., 2004; Lodha et al., 2006), this approach is naturally limited by the actuality and also the 3D nature of objects. But thanks to the wider dissemination of hybrid sensor systems carrying a laser scanner and camera system(s), RGB color information can be used much more accurately now as basis for radiometric feature computation (Haala et al., 2020).

After completion of feature calculation and optional filtering of features identified as carrying negligible additional information to reduce overfitting chances and thus to improve generalizability and interpretability (Chehata et al., 2009; Weinmann et al., 2013, 2014, 2015a), it is presented to a supervised classification scheme. This may be a Nearest Neighbor classifier (Jutzi and Gross, 2009; Weinmann et al., 2013) or more sophisticated approaches like a SVM (Lodha et al., 2006; Mallet et al., 2011; Weinmann et al., 2013; Blomley et al., 2014) or Boosting (Lodha et al., 2007; Guo et al., 2015). However, the quasi-standard for feature-driven point cloud semantic segmentation is the RF classifier, which is mainly due to its parallelization capabilities (Chehata et al., 2009; Shapovalov et al., 2010; Kim and Sohn, 2010; Niemeyer et al., 2014; Weinmann et al., 2014, 2015c). Due to the point-wise inference scheme, obtained classification results might be noisy with respect to the class affiliation of points compared to their neighboring points. To counter this, even simple regularization approaches can lead to a significant improvement of the overall classification result (Schindler, 2012), as proven for 3D point clouds by Charaniya et al. (2004). More sophisticated methods achieve regularization by embedding previously mentioned classification approaches as a base classifier in a Markov Random Field (MRF) (Anguelov et al., 2005; Lu and Rasmussen, 2012) or a CRF (Anguelov et al., 2005; Shapovalov et al., 2010; Lu and Rasmussen, 2012; Niemeyer et al., 2012, 2014; Najafi et al., 2014; Schmidt et al., 2014; Weinmann et al., 2015b; Vosselman et al., 2017; Landrieu et al., 2017).

2.8.2 Data-Driven Semantic Segmentation

From early developments of modern neural networks, formulated as CNNs by Lecun (1989), to their breakthrough in 2012 with *AlexNet* (Krizhevsky et al., 2012), the idea of ML has fundamentally changed. Feature computation is no longer seen as a pre-processing step for feeding a respective classifier, but is understood as part of the ML process. This means that only the raw data is presented to the classifier and the required set of features is to be inferred by the machine. Thus, engineering by an expert is no longer required for feature calculation and possible limitations of data description to which humans might (unknowingly) be subjected are lifted (e.g., try to consider a set of features describing the presence of a hand in an image). In the era of DL (or AI to name it in a more popular scientific manner), it is the expert's responsibility to design a network that is capable of inferring meaningful features from the data. Since such CNNs achieved astonishing results in benchmark challenges such as the *ImageNet Large Scale Visual Recognition Challenge* (Russakovsky et al., 2015a) for 2D imagery, where they have reached a rather mature state with marginal error rates, it is only natural trying to apply them to 3D data (i.e., point clouds in our case) as well. However, this poses new challenges since 3D data is not organized in a regular grid structure like 2D imagery. Therefore, the core operation of CNNs, the convolution of input data with a specific kernel comprising a set of learnable weights (Goodfellow et al., 2016, p. 322), must be defined to be applicable to 3D data. To this end, various approaches have been proposed recently, as discussed by Griffiths and Boehm (2019), Guo et al. (2021) and He et al. (2021).

While these methods differ in the manner convolution is being performed, many follow the well-known *U-Net* architecture (Ronneberger et al., 2015) when being employed for semantic segmentation. This architecture is capable of deriving instance-wise (i.e., pixel-wise or point-wise) predictions, where in the encoding branch a given input is subsequently subsampled. This allows to enlarge the receptive field and to apply an increased number of kernels for gaining a deep understanding of the input. Subsequently, this representation is upsampled to the original input shape in a symmetric decoder branch utilizing explicit links from the encoder. In the following, some of the most prominent DL representatives for 3D semantic segmentation are reviewed with special emphasis on how they solve 3D convolution as core function in a *U-Net* architecture.

A straightforward implementation of convolution for 3D point clouds is to resort to a voxelization to the 3D data to obtain a regular (but sparse) grid representation, where theoretically convolution can be applied simply by adding another dimension to typical 2D filter kernels. However, this is computationally prohibitive and would also be inefficient in case of geospatial data since the resulting voxelization of a typical ALS point cloud will mainly be built from empty cells, as we are basically dealing with a surface (i.e., the earth's surface) instead of extensive 3D volumes. A solution to this problem is provided by Graham (2015)

and Graham et al. (2018), who advocate simplifying the convolutional operation by computing it only for active input cells, i.e., cells that enclose points. Strictly speaking, we are no longer dealing with a conventional convolution since active input cells are processed independently. However, the authors argue that aggregation operations (e.g., pooling or strided convolution) allow propagation of information in the network, so that independently processed cells communicate implicitly. This simplified convolution operation is embedded in a typical *U-Net*-like encoder-decoder architecture, which works analogously to the 2D counterpart. More details of the resulting Submanifold Sparse Convolutional Neural Network (SCN) for semantic segmentation can be found in Section 3.7.2.

One of the first solutions for applying convolutions directly on 3D points was provided in *PointNet* (Qi et al., 2017a). The core idea is that a set of points (which might also carry features) can be mapped to a high-level understanding (i.e., vector) by pursuing a series of convolutions in terms of shared Multi-Layer Perceptrons (MLPs) and preceding transformation operations for normalization. This is followed by a symmetric function (such as max pooling) to summarize the input and to gain permutation invariance. Finally, such a global feature vector can be mapped to a finite set of classes in case of the classification network. If formulated as segmentation network, the global feature vector is concatenated with high-level per-point feature vectors from earlier layers to be mapped to per-point class scores by again employing intermediate MLP layers. Since by design this architecture is not capable of assessing fine-grained local structures, the authors developed a hierarchical network denoted as *PointNet++* (Qi et al., 2017b), basically representing an implementation of the *U-Net* concept. In the encoding branch, a set of points is thinned out before each convolutional operation, where the output point set is obtained by farthest point sampling (sampling layer). For each point kept, the respective point neighborhood (where potential neighbors come from the point set of the previous layer) is identified relying on a spherical or *k-nn* neighborhood (grouping layer) and encoded in a (deep) feature vector carried by this point by means of a mini-*PointNet* layer. To avoid unstable feature learning due to varying point density in a point cloud, special attention is paid to generating multi-scale features by concatenating *PointNet* responses for multiple neighborhoods. For obtaining per-point class scores for semantic segmentation, required upsampling in the decoding branch is achieved by interpolating features from a previous layer to the point set in the current layer, which are then concatenated with skip-linked point features from the opposing layer in the encoder and aggregated by a unit-convolution/*PointNet*.

Motivated by convolution on rastered data with well-defined filter kernels, Thomas et al. (2019) introduced *KPConv*. The input is the same as for *PointNet*/*PointNet++*, i.e., feature vectors at specific point locations, but convolution is performed by aggregating neighborhood points with so-called kernel points positioned in the vicinity of each input point (which can be assumed to be fixed within the neighborhood or to be deformable controlled by learnable shifts). This means that a spherical neighborhood is centered in each input point, in which a well-distributed set of kernel points is located, each carrying a weight matrix. For each neighbor point, its feature vector is first multiplied by the weight matrix of each kernel point before computing the weighted sum over all kernel points, where weights and thus the impact of a specific kernel point on the neighbor point (the so-called *correlation*) is based on the distance between the two points. The result of the convolution for the specific point of focus is then obtained by summing the filter responses of all neighbor points. When being utilized for semantic segmentation, the convolutional operation is again employed in a *U-Net* like architecture. Although *KPConv* performs convolutions directly on 3D points, the input is typically subsampled by usage of a regular grid, where for each grid cell the barycenter of all included points is computed and kept as point location to carry a convolution's result. Thus, it greatly resembles voxelization of the input as used for SCN. In the encoding layer, resolution is gradually reduced by doubling the size of the raster cells, thus increasing the receptive field. Similar to *PointNet++*, the decoding branch is built by nearest upsampling in combination with concatenating of those features with skip-linked features and aggregated by an unary convolution to finally obtain point-wise class scores.

With especially having large-scale outdoor point clouds in mind, Hu et al. (2020) designed *RandLA-Net*. Its key feature for efficient processing of such point clouds is progressive downsampling by simple

random sampling of points from one layer to another. However, this poses the risk of losing most prominent information. Thus, a local feature aggregation module has been developed alleviating this issue and defining the convolutional operation on 3D points. Again, for each 3D input point with possibly available additional attributes, the distinctive characteristic of the local surrounding is to be encoded in terms of a feature vector. Hence, the first step is to define a set of neighboring points via *k-nn* search. For each neighboring point, both (relative) 3D coordinates and features are separately mapped to a deeper understanding via MLPs and afterwards concatenated (*local spatial encoding*) similar to *PointNet*. Unlike the max pooling operation in *PointNet*, the global feature vector in *RandLA-Net* is derived in a more information-preserving way by *attentive pooling* where input features are aggregated by a weighted summation based on learned feature importances. To be less prone to losing information due to random sampling, the authors opt for a dilated residual block where two sets, each consisting of a *local spatial encoding* module and *attentive pooling* module, are stacked together and combined with a skip connection to be able to increase the receptive field. This represents the core functionality of *RandLA-Net* and is applied in conjunction with random sampling to build the encoder branch of a typical encoder-decoder network. Like for *PointNet++*, in the decoding branch learned features are propagated through nearest neighbor interpolation to points of a higher resolution level, concatenated with the opposing features of the encoder branch and fed into an MLP layer. Finally, obtained point-wise features are mapped to class scores by means of shared fully-connected layers.

Chapter 3

Methodology

To form the aspired hybrid intelligence system, there are two different main components we need to combine, namely i) human intelligence represented by the crowd and ii) an ML system capable to perform the task of semantic segmentation of 3D point clouds by learning from crowd-generated labels. To fuse both actors, the overall framework we rely on is AL, a specific learning scheme for ML models to drastically reduce human labeling effort to only valuable samples.

Before explaining the individual elements of our proposed methodology, we will start this chapter with an outline of our overall system design (Section 3.1) and an intuition of *why* this is a cost-efficient solution to our problem statement of semantic point cloud enrichment (Section 3.2). Section 3.3 is dedicated to comprehensively address the key element of the AL system, which is detecting most informative samples. To minimize costs even further, we i) rely on a reformulation of AL to allow for reusing pre-existing information in the sense of TL (Section 3.4), and ii) aim to stop our AL respectively ATL loops as soon as convergence is reached (Section 3.5). With AL essentially being the mediator between *human* and *electronic processing units*, we further elaborate on those two components, represented by our paid crowd (Section 3.6) and a respective ML algorithm (Section 3.7). Finally, we comment on relevant requirements on the technical infrastructure and realization of the envisioned system (Section 3.8).

3.1 An Outline of the Proposed Hybrid Intelligence System

Our approach is designed to semantically enrich laser scanning point cloud data, which is nowadays available abundantly but only rarely carries semantic information. We do not formulate any requirements for the point clouds, i.e., neither precise georeferencing nor proper co-registration of individual scans is necessarily required, although the latter is for sure beneficial both in terms of human and machine interpretability. Thus, theoretically, it is possible to process point cloud data from different sources, such as Terrestrial Laser Scanning (TLS), MLS or ALS and even multi-view stereo. But we especially focus on ALS point clouds - captured either from manned or unmanned systems and thus having a different scale and resolution.

To initialize the process as schematically depicted in Figure 3.1 and described in Algorithm 1, the unlabeled point cloud U (or subsets in case of extended clouds) is presented to an annotation engine. The annotator or the so-called oracle $\mathcal{O}$ can be a simulated GT oracle, but in our case, we do not only aim at demonstrating that AL is suited for minimization of data labeling, but our greater goal is to employ a realistic crowd oracle for annotation (cf. Section 3.6). In an ideal world, we would be able to find and motivate a large corpus of crowdworkers who voluntarily participate in our labeling campaigns. However, advertising such a campaign and recruiting crowdworkers is a time-consuming and, especially in case of tedious data annotation tasks, often futile endeavor. To address these recruitment limitations, we rely on paid crowdsourcing. By us-

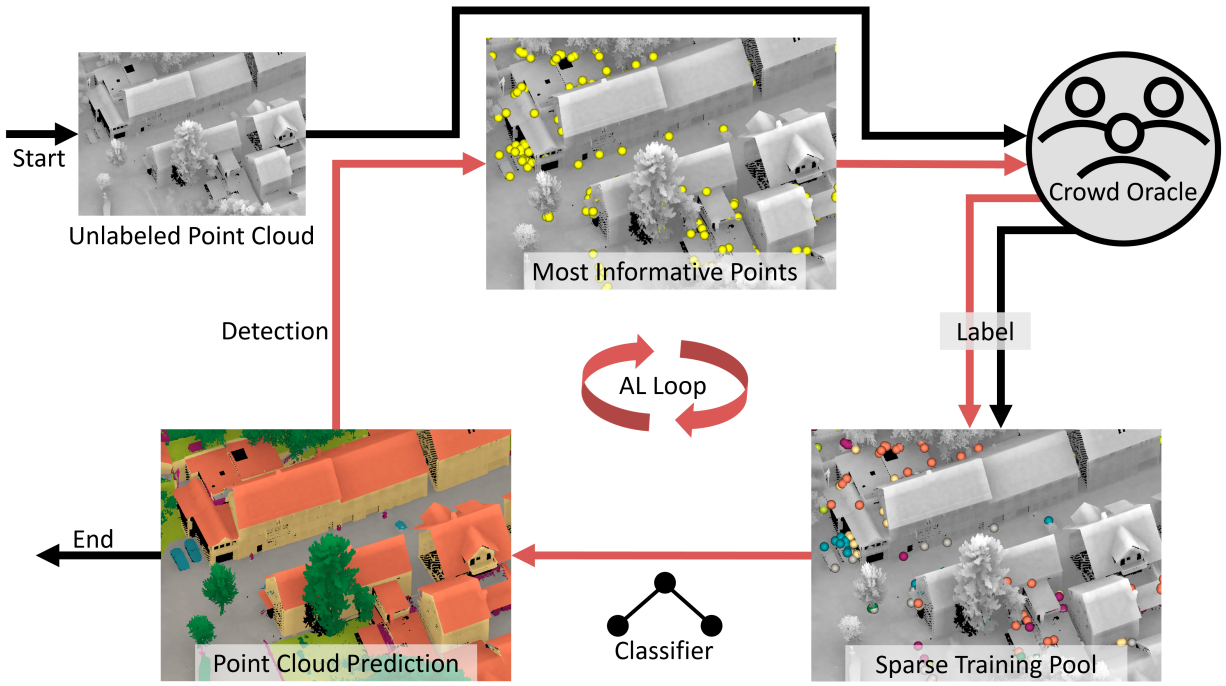


Figure 3.1: An overview of our crowd-based overall pipeline for semantic segmentation of 3D point clouds. *Red arrows* indicate steps that are conducted multiply as part of the AL iteration loop.

age of respective platforms (such as *MW* or *MTurk*), we can post campaigns as open calls to all crowdworkers registered on the specific platform, significantly easing the advertisement component. The lack of motivation to actually complete tasks and especially in a timely manner, can be easily solved by extrinsically motivating crowdworkers through payment. However, we also develop a gamification approach to the labeling problem, which is intended to increase the intrinsic motivation of crowdworkers. This approach ideally leads to more labels being set and to a higher level of accuracy (cf. Section 3.6.5).

The first task of our crowd is to set up an initialization data set L_{raw} with n_j samples for each of the n_Ω classes defined by an operator (cf. Algorithm 1). For this purpose, an appropriate tool needs to be provided to enable the crowd to freely indicate samples for each class (cf. Section 3.6.2). The challenge inherent in this free point selection, however, is that quality control is hard to implement. In order to minimize errors, we conduct a subsequent campaign in which crowdworkers check the results of crowdworkers from the first campaign in order to filter the initially produced training set L_{raw} to obtain the actual initialization data set L_{init} (cf. Algorithm 1). But if only one crowdworker checks a specific point, we end up with one word against another. Thus, the idea is to check the validity multiple times, leading to the question of *how many* multiple and favorably good-faith acquisitions are actually required. Integrating gross error filtered multiple acquisitions can then be considered as an automated quality control mechanism for receiving high-quality labels (cf. Section 3.6.3).

Eventually, the crowd provides a first set of labeled point instances L that in our case can be used for training an ML model M for semantic segmentation of 3D point clouds (cf. Figure 3.1 & Algorithm 1). The AL loop as backbone of our pipeline can be built around most state-of-the-art classifiers, as long as they are capable to express besides their predictions also their uncertainty (cf. Section 3.7). The only difference to training a classifier in AL, compared to PL, where generally completely labeled training sets are expected, is

Algorithm 1 Crowd-driven AL loop for the semantic segmentation of 3D point clouds

Input
- unlabeled point cloud $U = \{\boldsymbol{x}_u\}_{u=1}^{n_p}$
- number of samples n^+ to be labeled in each iteration step i
- definition of desired class catalog containing n_Ω classes
- access to oracle $\mathcal{O}$
- (labeling budget)

initialize labeled training set $L = \{\}$
query $\mathcal{O}$ to generate raw initialization data set $L_{raw} = \{(\boldsymbol{x}_r, c_r)\}_{r=1}^{n_j \cdot n_\Omega}$ containing n_j samples per class
use $\mathcal{O}$ to filter L_{raw} by means of quality checks and discard all false representatives of L_{raw} to receive $L_{init} = \{(\boldsymbol{x}_j, c_j)\}_{j=1}^{len(L_{raw})-?}$
set $L = L \cup L_{init}$ and $U = U \setminus L_{init}$
initialize queried label set $L_i = \{\}$
while stopping criterion not met or labeling budget not exhausted **do**
    train the ML model M using L
    predict on U to get class probabilities $\boldsymbol{p}(\boldsymbol{c}|\boldsymbol{x}_u)\ \forall\ \boldsymbol{x}_u \in U$
    derive a sampling score $s\ \forall\ \boldsymbol{x}_u \in U$ (cf. Section 3.3)
    select the n^+ samples with highest score and ask $\mathcal{O}$ for labels thus generating $L_i = \left\{\left(\boldsymbol{x}_i^+, c_i^+\right)\right\}_{i=1}^{n^+}$
    set $L = L \cup L_i$ and $U = U \setminus L_i$
    set $L_i = \{\}$
end while

Output
- final training set L
- trained model M
- full annotation of originally provided point cloud U with points being either manually annotated by $\mathcal{O}$ or automatically labeled by M

that the training point cloud in our case is sparsely labeled. This means that only a few points are annotated, but remaining "background" points are still indispensable for deriving meaningful features. This property needs to be considered in setting up respective classifiers since it deviates from conventional training schemes. Nevertheless, provided presence of a suitable classifier, based on the initial training data set, a first training cycle and thus prediction on all remaining unlabeled data points (i.e., the complete point cloud except for the initially labeled points) can be performed (cf. Figure 3.1). This already results in the trained model M and a full annotation of the originally provided point cloud U, which is now either manually annotated by $\mathcal{O}$ or automatically labeled by M. Thus, theoretically, the pipeline could already be stopped at this point. However, since it is unlikely that a model solely based on such a sparse initialization data set yields satisfactory results, a refinement of the model and the derived labeling is now being pursued.

The idea underlying the continuation of the workflow is that the classifier is capable of identifying those samples in the remaining pool of unlabeled data U with the most predictive uncertainty, and that inclusion of those points into the training data set would lead to a better performing model (i.e., we assume a pool-based AL setting). In other words, we focus manual labeling effort on only those points which are worth it. For discovering the few most informative points (cf. Figure 3.1), a respective query function is to be employed to compute individual sampling scores s (cf. Algorithm 1). Although sampling most uncertain points is a viable way for the ML model under development, a realistic crowd oracle diminishes the potential improvement through those points. This is because crowdworkers will struggle to label them since they are often situated directly on class borders. Thus, sampling strategies with special emphasis on points that are both informative *and* feasible to label need to be developed. Since 3D data is nowadays utilized and presented not only as 3D point clouds but also as 3D textured meshes, we can also alleviate possibly difficult interpretation of

3D data based only on discrete points by presenting the crowd a continuous and textured mesh instead (cf. Section 3.6.4).

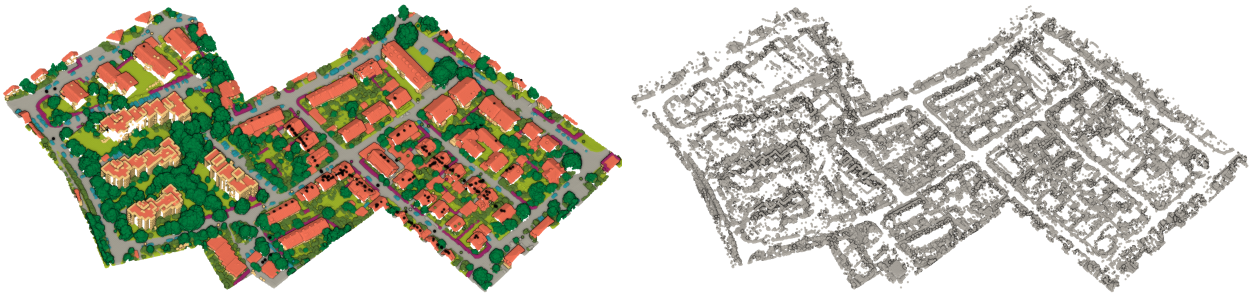
No matter the query function or the chosen 3D representation, the selected n^+ points are then presented to the oracle $\mathcal{O}$. Again a proper tool is required to ask the crowd for the labels of specifically indicated points (cf. Section 3.6.2) in the current round i to compose L_i (cf. Algorithm 1). Based on the training pool enhanced with the current batch of training samples $L = L \cup L_i$ (cf. Algorithm 1), the next training cycle of this iterative procedure can be started and the loop can be repeated until exhaustion of labeling budget or, preferably, until convergence, where more labeling effort would only lead to marginal improvements in performance. Detection of convergence and consequent termination of the loop can be achieved by defining a robust stopping criterion (cf. Section 3.5). Although such an AL-based learning setup is a viable solution to establish a powerful ML model for reasonable predictions, the question of sufficient generalizability for a new (but related) point cloud arises. It is to be assumed that a model built for a specific data set of the so-called source domain will yield suboptimal predictions on another point cloud of the target domain. However, AL, or more precisely ATL, provides a solution to this issue by making use of the information already available or learned that is also beneficial for the target domain, but also by adding a limited number of the most relevant samples from this new domain (cf. Section 3.4).

We argue that the complete AL and ATL loop, respectively, incorporating the crowd can be considered as a fully automated workflow i) if we are able to employ an automated quality control of crowd-generated labels (cf. Section 3.6.3) and ii) if we find a way to completely automate the communication between the machine conducting the AL loop and the crowd. Finally, this leads to the so-called CATEGORISE framework (cf. Section 3.8). Although humans are part of such a hybrid intelligence system, they can be thought of as *human processing units* working in collaboration with *electronic processing units*. While the latter are naturally responsible for all operations on the machine side (e.g., automatic classification, sampling of points, etc.), labeling tasks assigned to *human processing units* are completed by means of ***artificial*** *artificial intelligence* (Bezos, 2007) and from the view of an operator, the *human processing units* behave just like the *electronic* ones.

3.2 An Intuition *why* AL Works

Before going into the details of the AL working principle and how to employ crowdworkers in this concept, we would like to give an intuition of *why* learning with only a few samples is a well-justified way to proceed, exemplified with respect to 3D geospatial data. The idea of building a classifier based on only few samples is not only an achievement of AL, but is also implemented in the well-known SVM classifier. For training such a classifier, a completely labeled training set is often employed, but eventually training a SVM means identifying the so-called support vectors. In the end, the separation of feature space by means of hyperplanes is solely based on those support vectors. This concept is also visualized in Figure 3.2, where we have trained a SVM classifier based on both geometric and radiometric features (discussed later in Section 3.7.1) for the ISPRS Vaihingen 3D Semantic Labeling Contest (V3D) benchmark data set (further presented in Section 4.2.1). We can observe that the training process equals a filtering step, keeping only points that describe object borders. The point cloud of support vectors (more precisely, their coordinates in object space) in Figure 3.2 *(b)* resembles a map showing the delineation or contour lines of individual buildings and estates. Therefore, we assume that such points are the most informative for the classification process.

If one of the top ranking classifiers of the pre-DL era, i.e., a SVM, can operate by means of only such few samples, other classifiers, including modern CNNs, should be capable of doing so as well. The significance of this observation is the vast potential to save labeling effort. For instance, our SVM classifier only keeps 21.68 % of provided training points and derives its predictions only from them, which is why labeling the remaining points is dispensable. This observation is in accordance with the strategy often pursued by dedicated experts who also try to label only a subset of available points sufficiently representing the diversity of data based on their data knowledge and experience (Waldhauser et al., 2014).



(a) *(b)*

Figure 3.2: Comparison of V3D's labeled training data set *(a)* and the support vectors derived from it *(b)*. Support vectors are mainly situated on class borders, so that contours are clearly visible. Class color coding is depicted in Figure 3.3.

This means that if we are able to find a way to directly and automatically identify the points that will impact the final model the most, *without* knowing their labels, we could really save the costs of annotating the rest of the points - this is where AL comes in. Based on an initially provided training data set, we evaluate the uncertainty of the model when predicting on unlabeled data points and query those the model is the most uncertain about. If this selection is accomplished by an efficient query strategy, often even a lower number of points needs to be labeled than a SVM would find support vectors (Mackowiak et al., 2018; Kellenberger et al., 2019). This is due to the fact that the SVM exploits the complete proximity of the decision border. Thus, lots of informative, but often similar points, quasi-duplicates, are used. AL attempts to avoid that by especially tailored query functions (cf. Section 3.3). Vice versa, in our later experiments, up to 81.21 % of sampled points are actually support vectors. This implies that there are also non-support vectors in the training set, which are mainly sampled in early iteration steps, where most demanding regions cannot yet be identified due to a suboptimal estimation of the separation hypothesis at this stage (Tuia et al., 2011b). In the course of the iteration, however, with the gradual enhancement of the training set, separating hyperplanes approximate an optimal position, thus also allowing for more precise queries (of support vectors).

To gain further insights into the working principle and success of AL, we aim to visualize the regions AL focuses on in both feature and object space. However, since classifiers operate in high-dimensional feature space, humans tend to prefer analyzing the distribution of selected points in object space. But this selection of points is only the consequence of effects in feature space, which is why we opt to focus on it as well. For the sake of interpretability, however, we conduct a remapping of high-dimensional feature space to 2D. Nevertheless, we are aware that this will inevitably lead to loss of information, thus not all inter-class relationships can be necessarily observed. For this remapping we opt to use *t-SNE* (van der Maaten and Hinton, 2008) as it allows a non-linear mapping (or dimensionality reduction) while keeping the relative distance between samples based on their similarity. This is done by iteratively minimizing the "distance" (measured by *Kullback-Leibler divergence*) between a distribution reflecting the situation in the high-dimensional space and one in the newly established low-dimensional space. While the source distribution is modeled by a symmetrized Gaussian distribution based on the distance between samples and with the variance being implicitly set by the user as the number of expected neighbors, called *perplexity*, for the target distribution a Student t-distribution is used (instead of the straightforward choice of another Gaussian distribution) to reduce the crowding problem (van der Maaten and Hinton, 2008). The optimization is accomplished by stochastic gradient descent. The result depends greatly on the value for *perplexity*, requiring to be set as a trade-off between focusing on local vs. global structures, which can lead to significantly different results (Wattenberg et al., 2016). For alleviating this issue, different values for the *perplexity* were explored and finally set to

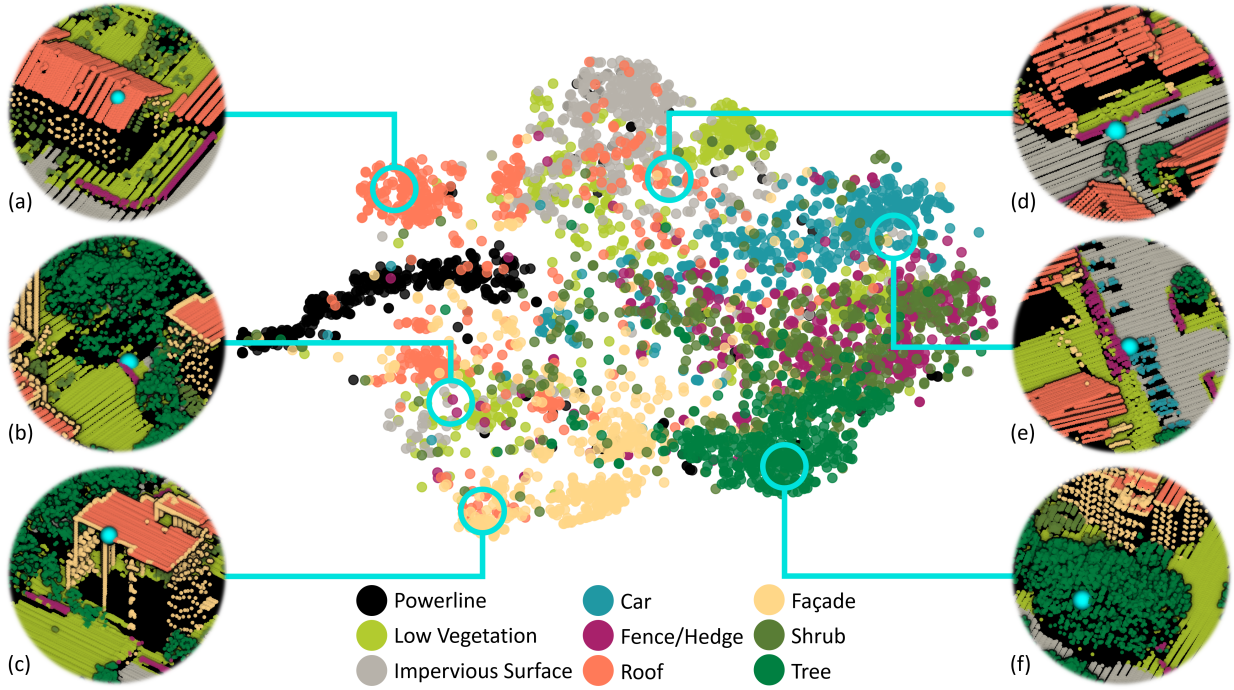


Figure 3.3: Embedding of feature vectors of V3D training data into 2D space by *t-SNE* along with an exemplary depiction of regions where AL points originate from (*blue circles*). For each indicated region, a representative is traced back to object space and indicated in *blue*. While points *(a)* and *(f)* were selected by crowdworkers in the initialization step, the remaining examples were actively queried in the course of the AL loop.

50 leading to the result as depicted in Figure 3.3 for the training set of the V3D data, relying on the same features as for the SVM classification (cf. Section 3.7.1).

The pure *t-SNE* output in Figure 3.3 gives an idea of the separability of individual classes. Overall, we can observe that some classes such as *Powerline*, *Façade* and partly class *Roof* clearly stand out even in a 2D representation. On the other hand, classes like *Fence/Hedge*, *Shrub*, and to some degree *Low Vegetation* are greatly entangled. However, overlapping features of those classes are exactly what one would expect. But please note that this does not mean that a classifier is doomed to fail on these classes. First of all, we can observe classes that *t-SNE* struggles to separate, which is likely due to overlapping features. But there is a remaining risk of suboptimal *t-SNE* parametrization, so that in the high-dimensional space a clear distinction could be possible. Thus, these results should not be overinterpreted.

In Figure 3.3, we also indicate exemplary regions of the training set AL draws its points from in (reduced) feature space, and also trace those selections into object space (the same figure with a dedicated visualization of all AL points can be found in the Appendix in Figure A1). We differentiate between regions where samples are drawn from in the initialization step (examples *(a)&(f)*) and regions visited in the course of the iteration (examples *(b),(c),(d)&(e)*). In the initialization step, crowdworkers are free to pick representatives of all classes and operate in object space. Naturally, they tend to select points that are as easy to label as possible, i.e., they would just pick a point in the middle of a respective object well away from class borders (see Figure 3.3 object space excerpts). This matches well with the depiction in the *t-SNE* plot, where we can see rather homogenous regions. However, active sampling within the AL loop takes place in feature space and mainly favors heterogeneous regions where a mixture of different class representatives is present. If exemplary points of such regions are traced back to object space, we can observe that such points are not only situated

close to the decision borders in feature space but also close to the class borders in object space. Thus, only the question of how to find these most informative instances in the high-dimensional feature space remains, being the focus of the next section.

3.3 Sampling Schemes in the AL Loop

From the perspective of the machine, the most crucial step is the definition of a proper query function that is capable to identify most informative points. Thus, most research in AL is concerned with developing such sampling strategies, and there is a great variety of possible approaches that can be found in literature (Settles, 2009). We will restrict ourselves to the discussion of those strategies we have employed in our endeavor of identifying most informative ALS points (Section 3.3.1). We also address possible refinements to meet the challenges of class-imbalanced data (Section 3.3.2), sampling under a pool-based AL scheme in batch-mode (Section 3.3.3) and working with an imperfect oracle (Section 3.3.4).

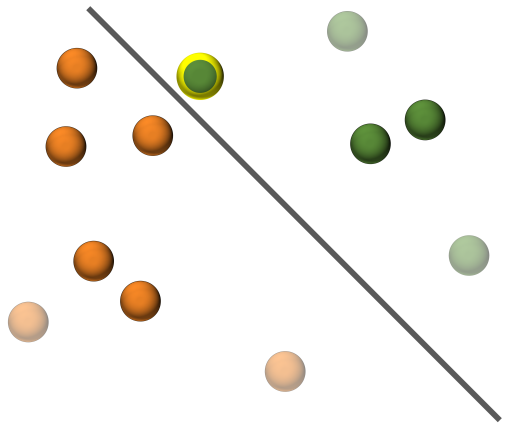
3.3.1 Query Functions

The easiest approach of sample selection in AL is *random sampling*. This means that in each iteration step we would just pick a fixed number of points n^+ in a random manner, without considering the current state of the classifier and its uncertainty at all. Nevertheless, this method quite frequently works surprisingly well, because often by chance we obtain some of the most informative points but of course also points that are not helpful to refine the classification rule, thus unjustly increasing labeling cost. Even worse, some of the most informative points will never get sampled, and hence the maximum achievable accuracy can be considered limited. With respect to the representativeness of the different classes in the final training pool, we assume that the relative distribution is quite similar to that in the complete underlying data set. In case of geospatial data, however, we are generally dealing with rather class-imbalanced data sets - for instance, consider the number of points representing chimneys compared to points that depict building roofs. If *random sampling* is applied, underrepresented classes in the complete training set will also be underrepresented in the sampled training pool, thereby only insufficiently reflecting respective intra-class variety. In the worst case, no points of certain (underrepresented) classes will be actively queried at all. Thus, more sophisticated sampling strategies are required.

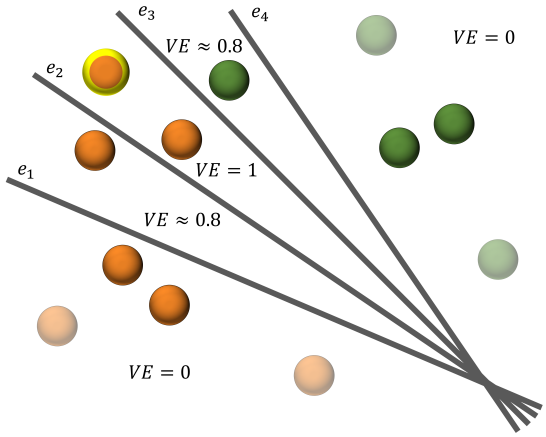
A more targeted sampling strategy would consider the predictive uncertainty of the classifier at its current state. Uncertainty-based sampling strategies are directly built upon the posterior probability $p(c|\boldsymbol{x})$ that point $\boldsymbol{x}$ belongs to class c. Since we are dealing with multi-class settings where we would like to take into account the predicted score for all n_Ω classes, we employ *entropy* E (Shannon, 1948) to evaluate this uncertainty. This metric theoretically assesses aleatoric uncertainty, i.e., samples are drawn from the vicinity of the current decision border when the posterior probability is given by a single classification model (although in early iteration steps epistemic uncertainty might dominate point scores). But it can also measure epistemic uncertainty, for example, if the averaged posterior probability of multiple hypotheses, such as from a *deep ensemble*, is used. Samples are then drawn according to:

$$\boldsymbol{x}_E^+ = \underset{\boldsymbol{x}\in U}{\operatorname{argmax}} \left(-\sum_{i=1}^{n_\Omega} p(c_i|\boldsymbol{x}) \cdot \log_2 p(c_i|\boldsymbol{x}) \right) \tag{3.1}$$

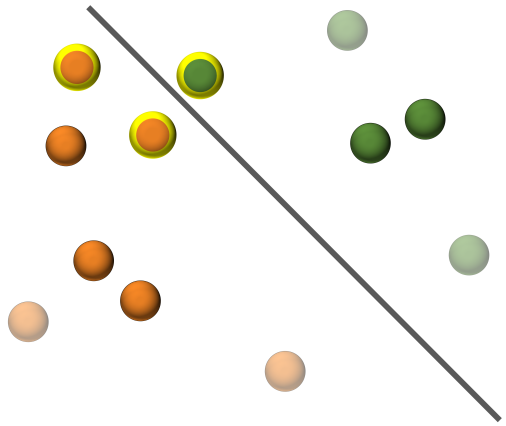
When evaluating *entropy*, we receive the minimum value, i.e. 0, when we have the score 1 for one class and consequently the score of 0 for each of the remaining classes. The maximum value, on the other hand, will be obtained whenever we are confronted with an equal distribution, i.e., every class gets the same score $1/n_\Omega$. Such points, the classifier is completely confused about are considered informative and will therefore be sampled preferably.



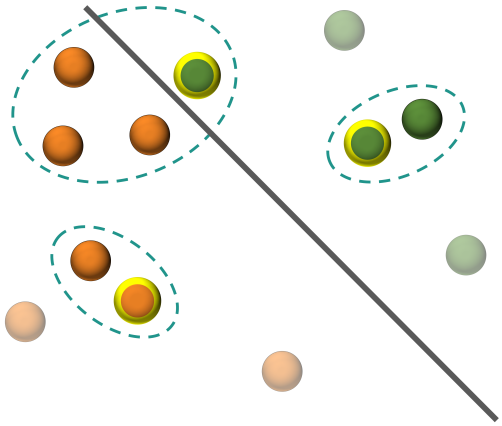
(a) Entropy sampling - emphasizing aleatoric uncertainty, but also epistemic uncertainty (especially in early iteration steps)



(b) Vote entropy sampling with $n_e = 4$ - scoring mainly epistemic uncertainty. VE values are valid for respective regions framed by decision boundaries



(c) Batch-mode AL with batch size 3 in combination with *entropy* sampling. Except for the point closest to the decision border, 2 quasi-duplicates get selected



(d) Batch-mode AL with batch size 3 in combination with *entropy* sampling and applied diversity criterion. *Dotted lines* represent formed *k-means* clusters

Figure 3.4: Comparison of different sampling strategies to gain most informative unlabeled points in an exemplary 2D feature space. *Transparent* points of both classes represent the current training data and are not subject to sampling. The current decision boundary is (or boundaries in case of *vote entropy* sampling are) depicted in *gray*. A *yellow border line* marks the point with highest score (*top row*) or multiple points with highest scores (*bottom row*). Please note that the class colors are only meant for a better visual interpretation since except for the *transparent* training points, no class labels are known to the ML model.

Figure 3.4*(a)* gives a visual interpretation of *entropy* sampling. When dealing with a two-class problem, the *entropy*-based sampling strategy is equivalent to sampling the point being closest to the model's current decision border learned from training points that are already available. Thus, in the binary case, it equals the *least confident* and *breaking ties* sampling strategies (Settles, 2009). Figure 3.5*(a)&(b)* gives an exemplary evaluation of *entropy* scores for the V3D data set for different states of the iteration. Although the classifier gets more confident in the course of the iteration, we can observe that throughout the loop, points situated on or near the class borders are most uncertain for the classifier to categorize, because they may reveal ambiguous

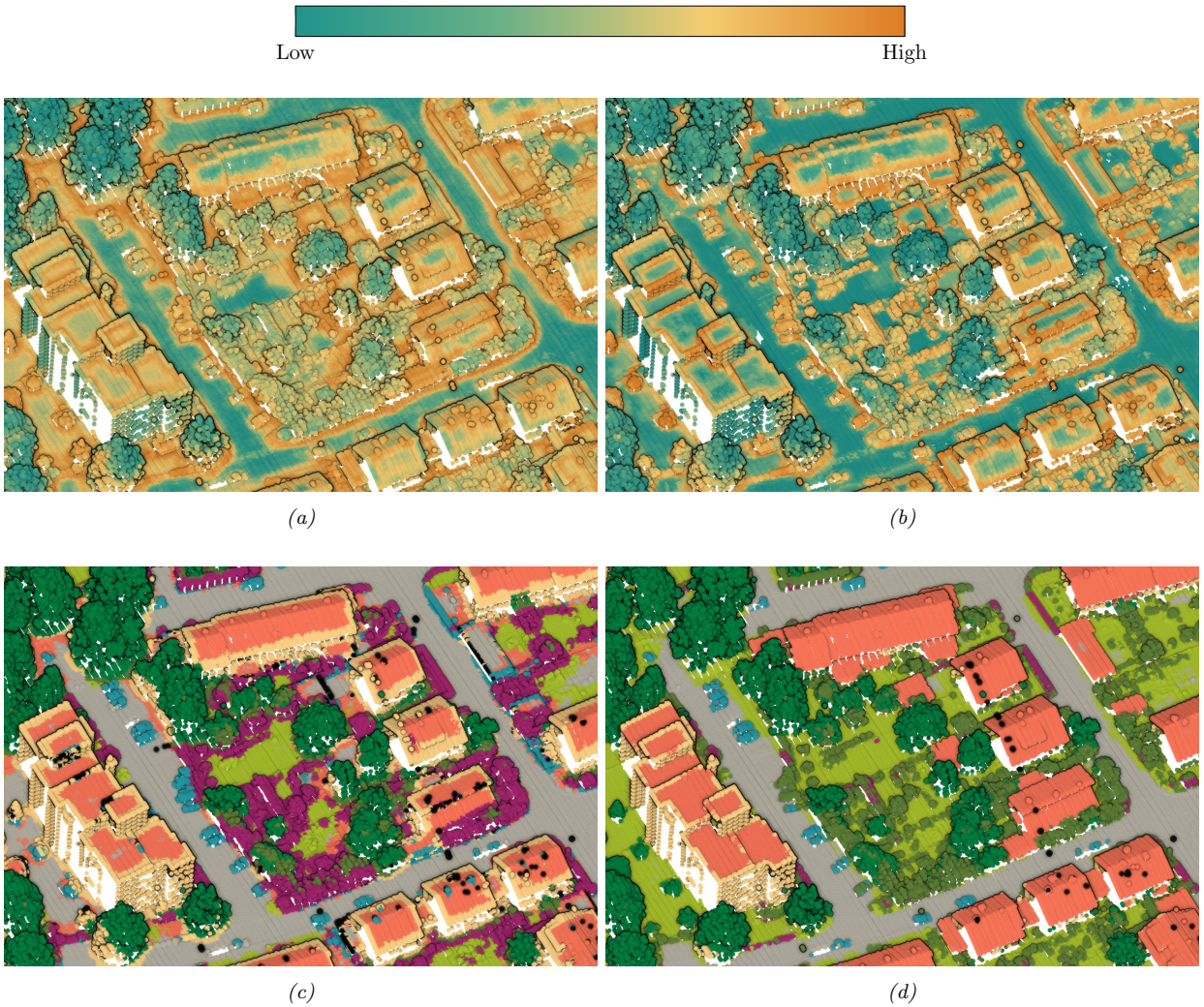


Figure 3.5: Entropy evaluated in the course of the iteration exemplarily for the first *(a)* and the last *(b)* iteration step for the V3D data set. Overall, the classifier gets more confident in its predictions, well corresponding to an improvement of predicted class labels (*(c)* vs. *(d)*), but class borders remain most challenging. Class color coding is depicted in Figure 3.3.

feature vectors, but will also provide most helpful clues for refining the decision border. Eventually, we prioritize sampling of support vectors (cf. Section 3.2) to gradually improve the current separation hypothesis, thus being an appropriate measure for our AL loop.

Another popular sampling strategy comes from the *query-by-committee* domain and is *vote entropy* sampling. There, we assume to have a committee or an ensemble $\mathcal{E}$ of n_e different (but similar) classifiers e. This can either be an RF ensemble (cf. Section 3.7.1) or a *deep ensemble* (cf. Section 3.7.2). The main difference to *entropy* sampling is that *vote entropy* VE considers the predicted posterior probability of an ensemble member $\boldsymbol{p}_e(\boldsymbol{c}|\boldsymbol{x})$ only implicitly. Precisely, it is utilized only for getting a vote from each ensemble member e for a specific class. When normalizing those votes, we receive a new distribution, which we evaluate based on the *entropy* formula to draw samples as follows:

$$\boldsymbol{x}^{+}_{VE} = \underset{\boldsymbol{x}\in U}{\operatorname{argmax}} \left(-\sum_{i=1}^{n_\Omega} \frac{\sum_e D(\boldsymbol{p}_e, c_i)}{n_e} \cdot \log_2 \frac{\sum_e D(\boldsymbol{p}_e, c_i)}{n_e} \right) \tag{3.2}$$

$$\text{where } D(\boldsymbol{p}_e, c) = \begin{cases} 1, & \text{if} \operatorname{argmax}(\boldsymbol{p}_e) = c \\ 0, & \text{otherwise} \end{cases} \qquad \text{and } \boldsymbol{p}_e(\boldsymbol{c}|\boldsymbol{x}) \text{ abbreviated as } \boldsymbol{p}_e$$

While *entropy* sampling relies purely on posterior probabilities (either from one model or averaged over multiple models) to select most informative points, as soon as the separation hypothesis is roughly set, this strategy might lead to a suboptimal choice of points. This is due to giving highest sampling scores to points close to the current decision borders, but such points will always remain to be complex for classification since they incorporate features of multiple classes. Thus, the separation hypothesis will only profit marginally from the inclusion of such points into the training set. For this reason, the basic idea underlying *vote entropy* sampling is that pure uncertainty is of subordinate importance as long as a model still predicts the correct class for such points. *Vote entropy* implements this mindset by measuring the disagreement of predictions among ensemble members e (and intentionally discarding uncertainty encoded in posterior probabilities). If all members vote for the same class (i.e., low VE), we would consider the respective point to be confidently predicted (even if the *entropy* score of averaged posterior probabilities over all committee models, as measure more related to aleatoric uncertainty, might suggest otherwise). But if those ensemble members vote for different classes, the sample is considered ambiguous, and labeling this point might have a great impact on the classification model, i.e., we identified it to populate a region of high epistemic uncertainty.

Figure 3.4*(b)* visualizes this principle for an ensemble classifier with 4 members, where each formulates a decision border in accordance with the current training data set, but that is still different from the others. This is possible since the current training samples are situated far away from the optimal decision border, thus giving room for a plethora of possible separation hypotheses. This situation is likely to occur in our crowd-based setting where the initialization data set is comprised of points freely picked by crowdworkers, which are expected to populate the class centers in feature space (cf. Section 3.1). The disagreement values by means of VE correspond well with those regions in feature space where selecting training points from would help to constrain the variance of separation hypotheses.

3.3.2 Fostering an Equally Class-Distributed Training Set

Both measures (E & VE) are well suited for detecting the points the classifier is the most uncertain about. However, if point sampling is solely based on these metrics, it is likely that classes that are underrepresented in the data set will be underrepresented in our point queries as well, if we assume that the relative number of points in the vicinity of class borders (or in regions of disagreement for *vote entropy* sampling) equals that of the complete data set. As a consequence, when points of smaller classes are not drawn, a refinement of the decision border(s) to effectively separate such underrepresented classes is likely to be not prioritized and might only happen implicitly by improving other class borders. Therefore moving the decision border(s) in feature space to a cluster representing this underrepresented class is also not prioritized, and when points of such a class are not next to the class border (or in the region of maximum disagreement for *vote entropy* sampling), they might never get sampled.

Thus, special measures for giving more emphasis to underrepresented classes are in order. This can be achieved by deriving a dynamic weighting factor $w_c(i)$ for each class c at each iteration step i by considering the ratio between the total number of samples currently included in the training data set n_L and the number of samples for a specific class n_c:

$$w_c(i) = \frac{n_L(i)}{n_c(i)} \tag{3.3}$$

These weight values are then multiplied by the predicted score of each class (E: $p(c|\boldsymbol{x})$, VE: $\sum_e D(\boldsymbol{p}_e, c)/n_e$), normalized and inserted into the *entropy* formula. In this way, we always increase the class score of the rarest class with respect to the other classes. The working principle of the weighting scheme can be interpreted to be twofold. While we aim to raise sampling scores of points from rare classes, at the same time scores of well-represented classes are to be diminished in order to "free" sampling capacity for samples from rare classes.

For the latter, the idea is to reduce sampling scores of points that would have been selected normally in an unweighted sampling scheme because they are likely to belong to well-represented classes. For instance, for the *entropy* sampling situation, consider an equal distribution of individual class scores for a given point and a three-class problem with class A having only one labeled instance in the current training set and classes B and C each having 5 labeled instances. Then, the weight value for class A is 11 and for the remaining two classes it is 2.2 each according to Equation 3.3. Applying the weighting scheme to this specific point (that probably belongs to a well-represented class) would change the equal score distribution to one having a strong indication of class A, so that the sampling score s (based on E or VE) is decreased. Thus, it is more unlikely than before that this point will be queried, i.e., we manipulated sampling by decreasing the score of a point that would have been selected otherwise to allow sampling of a representative of a point from an underrepresented class. Vice versa, if all classes are equally represented in the current iteration step (i.e., equal weights), there is no need of tampering with sampling scores.

As for the other aspect, i.e., boosting scores of scarce classes, the idea is to focus labeling resources on all samples that incorporate at least a weak hint to be a representative of a scarce class. For example, considering the same class distribution in our training set (and thus weights) as before. When we are confronted with a specific sample having class scores of $p(c_A|\boldsymbol{x}) = 0.1$ and $p(c_B|\boldsymbol{x}) = p(c_C|\boldsymbol{x}) = 0.45$, we would alter the distribution to resemble more an equal distribution to emphasize inclusion of this point, which might stem from class A, into our training data set. On the other hand, if the class scores give absolutely no indication of class A and the two other classes are equally likely (i.e., $p(c_A|\boldsymbol{x}) = 0$; $p(c_B|\boldsymbol{x}) = p(c_C|\boldsymbol{x}) = 0.5$), we would not change the resulting score, thus not interfering with the classifier's selection. Since our weighting scheme emphasizes sampling of points that can lay farther away from the current (maybe imperfect) decision border, it can also help to put focus on finding new, so far unsampled clusters. Respective points are otherwise never queried (considering the *entropy* sampling scheme), which is a well-known issue of AL (Matasci et al., 2012).

3.3.3 Guaranteeing Diversity in Sampled Batches

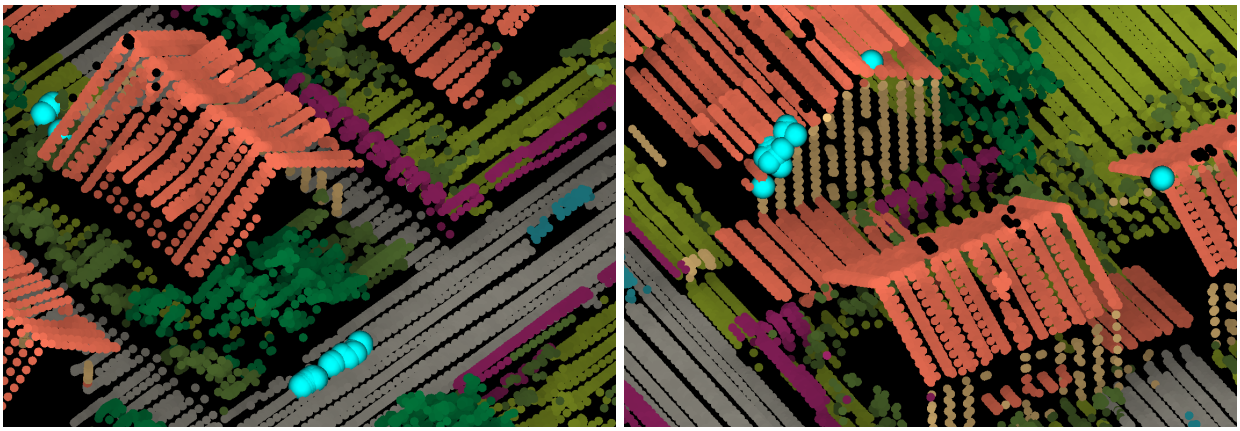
The weakness of the aforementioned sampling strategies is that they are specifically designed for querying one instance at a time. From the view of an optimal selection of points, the classification rule should be updated with each new training sample to give the best possible estimate of the next point to be labeled. But sampling one instance per iteration step is statistically questionable and just not efficient for most classifiers. Thus, a batch of points is commonly selected at once. The easiest approach to this would be to choose the n^+ points with the maximum sampling score. If this is done relying on *entropy*, the samples that are picked might be rather similar to each other (with respect to their representation in feature space), as can be seen from Figure 3.4*(c)*. This means that, except for the sample lying closest to the decision border, quasi-duplicates are selected that might not provide significant value to refining the separation hypothesis. Therefore, statistical significance is solved, but efficiency still remains an issue. To boost the convergence of the AL loop, a set of points that is as informative *and* as diverse as possible should be selected. For the latter to achieve, a clustering algorithm of feature space can be utilized to get n^+ clusters, where we would like to sample only one point per cluster, so that all points of the batch are sufficiently different. This can be implemented by the *k-means* algorithm (Lloyd, 1982), which sets k cluster centers (in our case $k = n^+$)

in a way that the euclidean distance between each data point in U and the distributed cluster centers $\boldsymbol{\mu}$ is minimal.

$$\sum_{\boldsymbol{x}_i \in U} \sum_{j=1}^{n^+} s_i \left\| \boldsymbol{x}_i - \boldsymbol{\mu}_j \right\| \rightarrow \min \tag{3.4}$$

Please note that we also introduce an individual weight for each data point, which is given by the sampling score s (E/VE or their weighted variants) of each point (Zhdanov, 2019). If pure *k-means* sampling was applied, the focus would be solely on the diversity criterion, but likely the clusters would mainly populate high-density class concentrations with low score values. Consequently, when sampling from those clusters, only an insignificant improvement of the current classification belief is to be expected. Incorporating the sampling score into the optimization process will help to ensure that most clusters are close to the decision borders, but most similar points are still summarized into one cluster, thus avoiding sampling duplicates. Figure 3.4*(d)* illustrates the improvement in terms of batch diversity when relying on feature space clustering (for simplicity, a pure *k-means* clustering is depicted). After clustering, we select the point with the highest sampling score from each cluster and refer to this method as *Diversity in Feature Space (DiFS)*.

In case of geospatial data, similarity of individual data points can also be considered as spatial similarity (Crawford et al., 2013). Generally, we would assume that neighboring points in object space also show similar features and thus are close to each other in feature space as well. To this end, an exemplary depiction of *entropy* sampled points is given in Figure 3.6. We can clearly observe that sampled points within one iteration step form clusters in object space. Thus, an alternative method to the *k-means* based *DiFS* method for guaranteeing diversity of sampled points is to enforce the diversity criterion in object space, referred to as *Diversity in Object Space (DiOS)*. This can be achieved by allowing one of the aforementioned greedy query functions to add a point to the current batch only if the distance to all previously sampled points in that iteration step is greater than some threshold d_{DiOS}. Setting this threshold, on the other hand, requires adjusting another parameter that can be hard to interpret. At what distance are two points too similar to be sampled with respect to their informativeness or their representation in feature space? It is worth emphasizing that this sampling add-on is intentionally restricted to operate within one AL iteration step each, so that in the final training data set points with a distance d_{DiOS} smaller than the threshold can be included. This alleviates the impact of the threshold because if the query function was prohibited from sampling an actually informative point in the current iteration step, in one of the next iteration steps there is still the option to add it to the training set.



(a) *(b)*

Figure 3.6: Entropy sampled points (depicted in *cyan*) of two exemplary iteration steps of a batch-mode AL loop. The point cloud data is colored according to GT. Class color coding is depicted in Figure 3.3.

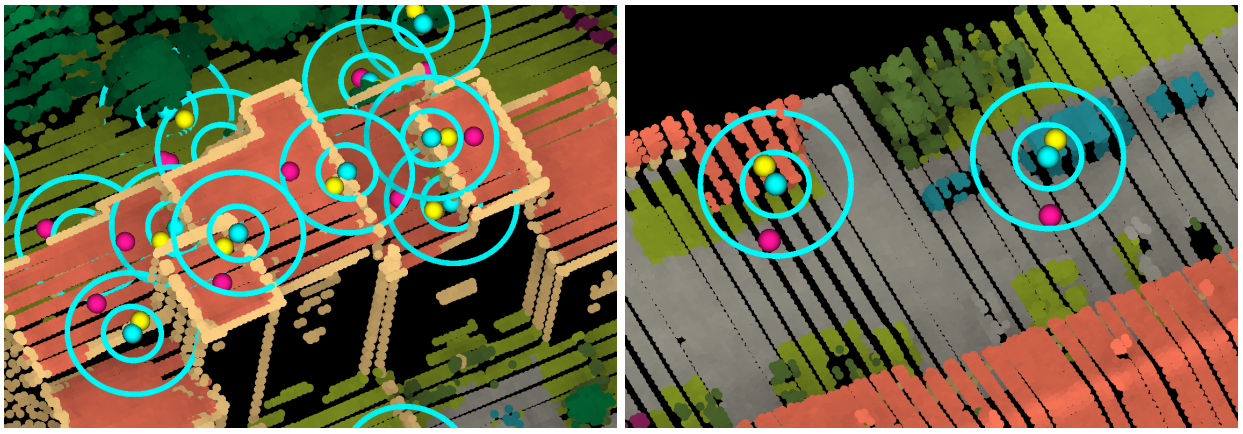
3.3.4 Addressing Imperfect Oracles in AL

While the aforementioned sampling strategies (*entropy* & *vote entropy*) and sampling add-ons to ensure diversity in batches (*DiFS* & *DiOS*) were concerned about the best selection of points from the view of the machine, this section is concluded by also addressing that, in contrast to many AL-related publications (Marcus and Parameswaran, 2015), we are dealing with a realistic imperfect crowd oracle. Thus, in favor of getting correct answers from crowdworkers, we might have to sacrifice informativeness. The idea is to consider the choice, that is the scoring of the machine, as a prior for the final selection of points.

As already discussed in Section 3.2 and observed from Figure 3.5, the classifier tends to select points that are situated on class borders in object space. There we have the highest informativeness values because the features incorporate characteristics of multiple classes. But we assume that interpretability for the machine is also related to human interpretability, so that those spots the current state of the ML algorithm is not sure in its decision are also demanding for crowdworkers.

For human interpretation, however, subjective understanding and definition also play a major role (Raykar et al., 2010). Often we are confronted with ambiguous points that are hard to assign to one distinct class. Therefore, we argue that increasing the distance to the class borders is related to *Reducing Interpretation Uncertainty (RIU)* and thus also refer to this method as such. Precisely, we consider a point selected by the machine as the seed point, but query a certain other point within a specific radius d_{RIU} around that point instead. In the end, the point with the lowest score in this region and thus hopefully with the highest interpretability gets selected.

When implementing this idea, the selection of points can be adjusted as visualized in Figure 3.7. As expected, the classifier selects points on class borders where it might be hard to decide for one specific class. For instance, the points on the borders between classes *Façade* and *Roof* in Figure 3.7*(a)* can be very well assigned to each of the two classes, depending on personal understanding of class affiliation. If we move away from the class borders, such ambiguities vanish, and a human operator can confidently and quickly decide for a specific class. But on the other hand, if the distance of the point to the class border becomes too large, the label might not bring any benefit for the training of the ML model in the end (cf. Figure 3.7). Thus, we are confronted with a trade-off between human interpretability and informativeness for the ML model.



(a) *(b)*

Figure 3.7: Working principle of our *Reducing Interpretation Uncertainty (RIU)* technique to alleviate label ambiguities for the oracle, depicted for two different scenes. Instead of the point that was actually queried (*cyan*), an alternative point within a certain radius (1.5 m for *yellow* and 4 m for *magenta*) that is supposed to be easier for human interpretation is sent to the oracle for labeling. Class color coding is depicted in Figure 3.3.

3.4 Active Transfer Learning

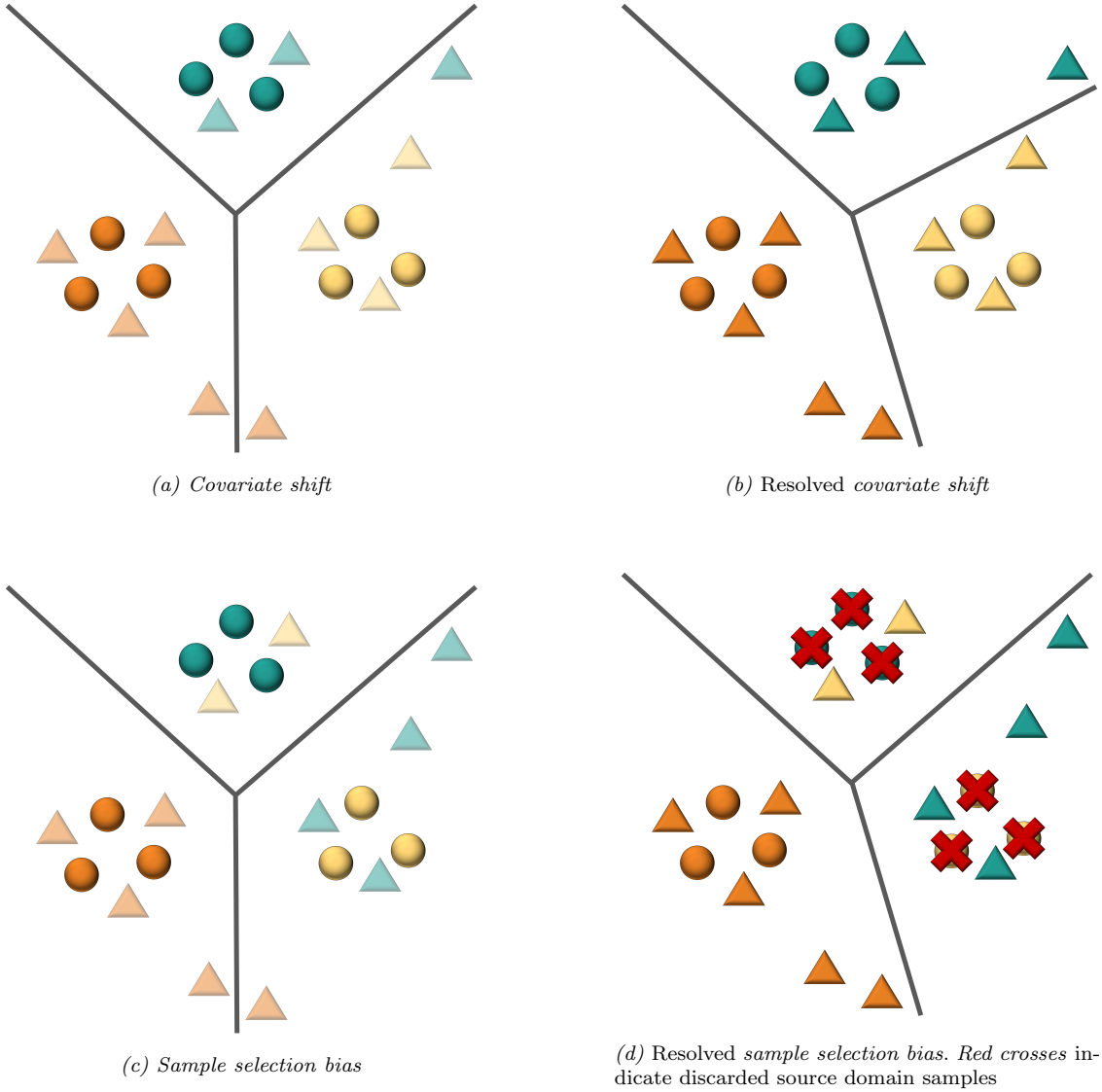
The previous section provided a guideline on how to minimize labeling effort with AL enabling a classifier to ascend to a satisfactory level of accuracy. However, the question of generalizability remains, i.e., whether a classifier trained for a specific data set will also output predictions of comparable accuracy for another data set as for the one it was originally built for. Often classifiers do not generalize well when supposed to operate on a new domain, the target domain D_T. The reasons for that can be manifold, but can be traced back to two distinct issues occurring in transfer learning tasks, that is, the *sample selection bias* (Heckman, 1979) and the *covariate shift* (Shimodaira, 2000) problem. In case of *covariate shift*, we assume that the marginal distributions inherent in the source domain D_S and the target domain D_T are different but the class conditional probabilities are sufficiently similar, i.e., $\boldsymbol{p}^S(\boldsymbol{x}) \neq \boldsymbol{p}^T(\boldsymbol{x})$ while $\boldsymbol{p}^S(\boldsymbol{c}|\boldsymbol{x}) \approx \boldsymbol{p}^T(\boldsymbol{c}|\boldsymbol{x})$. This can be interpreted as representatives of a specific class being described by a sufficiently similar feature vector in both domains, but some samples are missing to define the separation hypotheses in D_T more truly or more generally. This issue can be solved by just including respective samples of D_T to refine the current class borders, as exemplarily visualized in Figure 3.8*(a)&(b)*.

However, the more general (and severe) case of domain gap is the *sample selection bias*. In this case, we are additionally confronted with samples being characterized by a similar feature vector but carrying different class labels in D_S and D_T, i.e., $\boldsymbol{p}^S(\boldsymbol{x}) \neq \boldsymbol{p}^T(\boldsymbol{x})$ and $\boldsymbol{p}^S(\boldsymbol{c}|\boldsymbol{x}) \neq \boldsymbol{p}^T(\boldsymbol{c}|\boldsymbol{x})$. Figure 3.8*(c)* depicts an example of such a scenario. Like for the *covariate shift* issue, some samples of D_T can be included to improve the decision borders. But even if new samples for class *yellow* and *green* are added, the classifier is left confused since samples of D_S are misleading for D_T. We can solve this shift problem by removing such deprecated samples from D_S to yield a proper set of decision hypotheses for D_T (cf. Figure 3.8*(d)*).

The approach we deploy to solve both types of domain gap problems was originally purposed for semantic segmentation of aerial imagery (Persello and Bruzzone, 2012; Persello, 2013). However, we aim to transfer it for the classification of 3D point clouds employing a realistic crowd oracle. Thus, the method discussed in the following and visualized in Figure 3.9 can be considered an extension or rather generalization of the already presented AL pipeline (cf. Figure 3.1 & Algorithm 1). Its initialization is the same as for a basic AL loop. The unlabeled point cloud is presented to the crowd, which generates a sparse initial data set (cf. Section 3.1) that is then used for training our ML model. After this initialization, we start sampling most informative points, receive labels from crowdworkers and employ them for training the ML model. We repeat this procedure n_i times to iteratively tune our classifier for the domain in which we started this process (i.e., D_S).

The straightforward way to adapt a classifier to a new domain by means of AL, or more precisely ATL, is to just continue the started iteration but to change the pool of unlabeled instances U the classifier is allowed to draw points from through the query function from the source to the target domain. This is an easy yet effective solution for the aforementioned *covariate shift* problem. For this, any sampling strategy (e.g., *entropy* sampling or *vote entropy* sampling) can be used, but in case of pool-based batch-mode AL, Persello (2013) stresses the importance of a diversity criterion. This is to encourage that samples are not only drawn from the direct vicinity of the current decision border, since this separation hypothesis might be suboptimal with respect to D_T, so that the actual most informative samples might be situated farther away. However, this is a scenario similar to a basic AL run where, depending on the initialization data set, decision borders can also only be estimated suboptimally. But in the case of ATL, we hope that at least some of the class-separating borders are positioned quite well. Otherwise, i.e., if the data distributions of D_S and D_T were completely different, the transfer process would be pointless. Anyways, newly sampled points are then used together with all samples from D_S to train the ML model in the course of the ATL loop.

To overcome the more general form of domain gap, i.e., the presence of *sample selection bias*, we do not only add n^+ samples of D_T through our sampling function, but we also remove n^- samples of D_S that are the most in contradiction with D_T. The idea is that as soon as labeled samples from D_T become available,



(a) Covariate shift

(b) Resolved *covariate shift*

(c) Sample selection bias

(d) Resolved *sample selection bias. Red crosses* indicate discarded source domain samples

Figure 3.8: Comparison of different data shift problems in transfer learning (*left*) and possible solutions to them (*right*) for a 3-class 2D feature space setting. Samples from the source domain are indicated by *bullets* and samples from the target domain by *triangles*. The current decision borders under optimization are marked in *gray*. *Transparent* target domain samples were not considered to build the decision borders based on source domain samples (*left*).

we can compare them against representatives of the same class from D_S. To avoid having to define a specific similarity measure including an appropriate threshold for this similarity, the predictions of two ML models are utilized: one being solely trained by source domain samples and the other one being trained by the combined training set of source *and* target domain samples. Please note that we could theoretically also oppose the predictions of a source domain classifier to one using *only* target domain samples, but such a comparison would be infeasible as the two classifiers would disagree significantly even without the domain gap. This is due to the training pool of the target domain classifier only including very few most informative points of a later stage in an AL loop, that might focus only on one specific class or on some rather fine-grained details across multiple classes. Thus, we refrain from the second option and discard target-conflicting

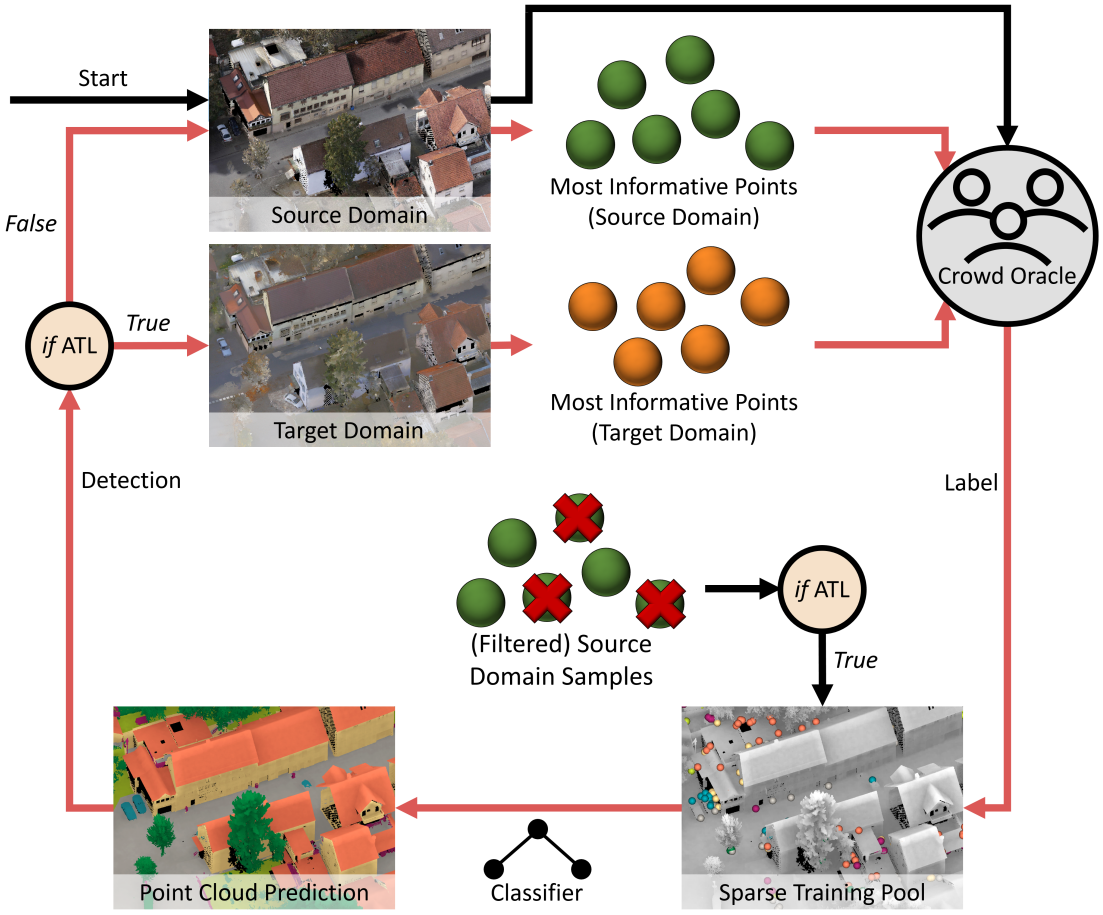


Figure 3.9: Overview of our crowd-driven ATL loop.

samples from D_S included in the labeled source training set L_S by measuring the disagreement between the predicted posterior probabilities of the two classifiers, as outlined in Equation 3.5.

$$\boldsymbol{x}^- = \underset{\boldsymbol{x} \in L_S}{\operatorname{argmax}} \Big(\left| \boldsymbol{p}^S(\boldsymbol{c}|\boldsymbol{x}) - \boldsymbol{p}^{S \cup T}(\boldsymbol{c}|\boldsymbol{x}) \right| \Big) \tag{3.5}$$

Hence, in ATL we need to decide on both the number of samples n^+ from D_T to be added to the training set and the newly introduced hyperparameter n^- of samples from D_S to be discarded. The latter is implicitly set by $n^- = q \cdot n^+$ (or $q = n^-/n^+$), with q being defined by the user as the ratio of deletion vs. inclusion. Depending on the batch size n^+ in tandem with q and the number of iteration steps, there are scenarios where the training set will be quickly cleansed of samples from D_S and learning will be solely based on D_T. However, this can be a handy property whenever training samples from a source domain have been included that are actually not meaningful for the target domain and are thus to be timely eliminated. But generally, we would assume that there is at least a small portion of information in the training samples from D_S which is worth learning from and that we do not wish to ignore.

But even for a dedicated expert, it is often hard to tell whether given source domain data will be actually helpful for the classification of another target domain point cloud (Waldhauser et al., 2014). Thus, the procedure is designed in such a way that samples of a given source domain can be included either way, and if they are helpful for D_T, the ATL loop will be boosted. But if they are contradicting D_T, they can be quickly eliminated (provided proper parametrization). In this manner, scenarios can be avoided where an ATL run requires more iteration steps than an AL run launched from scratch on D_T.

3.5 Stopping the Loop

As already emphasized earlier, our aim is to establish an automated hybrid intelligence system that minimizes human interaction and thus costs as much as possible. Considering both automation and minimizing costs, an effective stopping criterion of the AL loop comes into focus. A compulsory and thus the easiest variant would be a scenario where we have a limited labeling budget that implicitly defines the number of affordable labels and thus the number of iteration steps (with known batch size n^+). A sophisticated termination criterion, on the other hand, can help when the budget for labeling is considerable and when the goal is rather to conduct the loop for as many iteration steps as necessary to reach a stable (and hopefully highly accurate) performance while avoiding superfluous costs. In other words, we would like to identify the optimal point where continuing the iteration would alter the results only marginally. Thus, improving the current state of the classifier would be disproportionately expensive, especially compared to early iteration steps, where typically only a few labels significantly improve classification accuracy.

The inherent difficulty in defining a stopping criterion is that we cannot assume to have a representative labeled test data set on which we could evaluate our classifier - otherwise, the AL idea of avoiding such a necessity would be violated. Nevertheless, deriving predictions for a preferably extensive and thus representative point cloud is possible. This might either be the remaining *unlabeled* training data set U or an independent *unlabeled* test data set. Thus, the congruence between the predictions of two iteration steps can be evaluated. If there is a significant change between the current prediction and that d_{stop} iteration steps earlier, it will probably be worthwhile to continue the iteration. Precisely, inspired by the approach of Bloodgood and Vijay-Shanker (2009), we measure the overall congruence C_o by simple comparison of predicted labels, but also derive a measure C_{ac} that is more sensitive to underrepresented classes. For the latter, for each class, we compute an individual congruence value by comparing whether all points currently predicted as this specific class were assigned to the same class in the considered previous iteration step. Afterwards, the mean value over all class-specific congruence values is calculated to have a single class-sensitive indicator. Actually stopping the loop is achieved by comparing the standard deviation of congruence values (either C_o or C_{ac}) computed over the last n_{stop} iteration steps (that can be interpreted as derivative of congruence values) against a user-defined threshold t_{stop}. This avoids the need to specify an absolute congruence value, which can be highly task and data specific. The threshold defines the aggressivity of the stopping criterion, i.e., whether the loop should stop at the earliest iteration step from which no significant change is expected anymore, or whether the loop should continue until we can confidently assume convergence.

As for the interpretation of the stopping criterion, congruence values computed over all n_{stop} considered iteration steps have converged to a stable niveau if the standard deviation is close to zero. In such cases, we assume that only few predicted class labels are still changing, i.e., probably those of ambiguous instances close to decision borders. But overall, the predictions have reached a stable state. However, if there is a significant standard deviation, this means that there is a change and congruence values are either increasing or decreasing - in both cases we would like to continue the iteration. The increasing case is the one we would generally prefer since it means we are approaching convergence. By enhancing the training data set, we assume that the classifier can be trained more and more confidently (low epistemic uncertainty) and thus congruence of class labels should also increase. If congruence decreases, however, this means that a region (or regions) in feature space is queried that changes the overall separation hypothesis significantly. For instance,

this might happen if a new cluster of a class was discovered in feature space, which will probably cause prolonging the loop.

The stopping threshold t_{stop} is to be set in consideration of the other hyperparameters, i.e., n_{stop} and d_{stop}. The latter determines the "long-term memory" of the predictions. The larger d_{stop} the more likely a discrepancy will be observed in comparing the predictions of respective classifiers due to the more different underlying training data sets, but the more confidently we can assume convergence if the predictions are similar. While d_{stop} is concerned with absolute predictive difference, the number of considered congruence values from previous iteration steps n_{stop}, in contrast, addresses the change of congruence/difference values. The smaller n_{stop} the more uncertain the derived change value expressed through the standard deviation (due to a lower number of observations), but the more likely it is to reach early a value close to zero (and thus probably $< t_{stop}$), since it is more prone to local minima.

Regardless of the parametrization, generally speaking, a stop will always happen with a delay of a few iteration steps, because a sufficient number of predictions must be present to detect congruence. Thus, the last few iteration steps that triggered the stopping criterion were actually needless from a performance-based point of view. Additionally, due to our intention to operate without reference labels, we can only detect convergence of predictions, which are not necessarily required to be on a level of satisfactory accuracy.

3.6 The Crowd as AL Oracle: Working with Non-Experts for 3D Point Labeling

While the previous sections discussed how both *human* and *machine processors* can be fused into one intelligence system, we will now focus on those two parts in more detail, starting with the human component, in our case the crowd. Before doing so, we briefly review different oracle types in an AL setting for semantic segmentation and locate the crowd in this taxonomy (Section 3.6.1). Unlike most literature in AL, in our hybrid intelligence system we are confronted with an imperfect human oracle, inevitably leading to labeling errors. Minimization of such errors can be achieved either by easy-to-use tools for the task at hand (Section 3.6.2), measures for quality control (Section 3.6.3), an increased incentive to contribute (Section 3.6.5), or by easing the labeling process by means of a different data modality (Section 3.6.4).

3.6.1 Oracle Types in AL

The type of oracle typically encountered in AL-related publications is a simulated oracle. The straightforward and most common variant is an omniscient or GT oracle $\mathcal{O}_O$, which always returns a true label for all points that are queried within an AL loop (Marcus and Parameswaran, 2015). Obviously, such an accuracy level of an oracle is not achievable when working with human operators - even if they are experts (Walter et al., 2016). Therefore, we advocate for oracles that at least try to simulate typical error behaviors of real crowdworkers. This incorporates both a so-called noisy oracle and an oracle with systematic errors (Lockhart et al., 2020). A noisy oracle $\mathcal{O}_N(\sigma)$ refers to a simulated crowd oracle that outputs $\sigma\,\%$ of random noise for all queried points, i.e., for $\sigma\,\%$ of AL points, a class label differing from the true one is randomly assigned. Such noisy oracles are to be expected whenever a human operator is tasked with labeling assignments, where noise can stem from the operator's tiredness, guessing when points are hard to interpret, or simply due to not paying attention. However, we would expect that our ML model incorporated into the AL process is able to cope with a certain amount of random noise in training labels, since most models generalize well.

The more severe kind of labeling error comes from a confused oracle $\mathcal{O}_S(\sigma)$ that systematically assigns false labels to queried AL points. Such an oracle - intentionally or in good faith - also labels $\sigma\,\%$ of points incorrectly, but it follows some concrete mapping rules (e.g., always confuses *Car* with *Urban Furniture* or *Roof* with *Façade*). Presence of such systematic errors can significantly impact the classification process and

can also lead to a "confused" ML algorithm. When real crowdworkers are employed as oracle ($\mathcal{O}_C$), we need to expect both kinds of errors.

3.6.2 Designing Labeling Tools for the Crowd

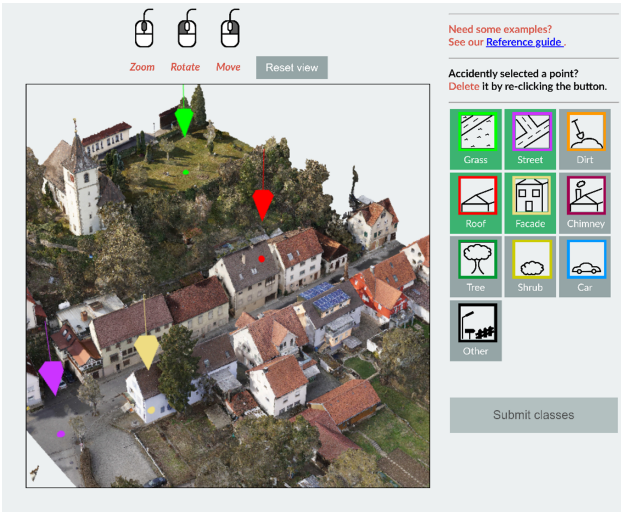
Before focusing on the more sophisticated methods for improving the accuracy of crowd-generated labels, we briefly present the tools that are required to accomplish the crowd-related tasks arising in our AL-based framework, as outlined in Algorithm 1. In general, all these tools should be designed to be as easy to use as possible, without requiring going through extensive intros or reading a specific documentation, since the main goal of most crowdworkers in a paid crowdsourcing scenario is to quickly make money. Therefore, extensive instructions are likely to be ignored either way, as stressed by Endres et al. (2010).

Additionally, we are not confronted with geospatial experts, but lay workers who are not familiar with the concept of aerial imagery, even less with ALS point clouds. Although an interested crowdworker could have some experience with this type of data through usage of navigation and map services such as *GoogleMaps* or *GoogleEarth*, we need to assume that actively navigating in a point cloud (i.e., zooming, rotating, panning) might already overwhelm some workers. Thus, such interactions with the data are also minimized to an absolutely necessary minimum. For all these reasons, the endeavor of designing suitable tools in a crowd-sourced data acquisition scenario is a complex task itself, but one that is often overlooked (Kittur et al., 2008; Vondrick et al., 2012) and only scarcely addressed (Sorokin and Forsyth, 2008; Kovashka et al., 2016).

As can be seen from Algorithm 1, the crowd is supposed to provide both an initialization data set as well as labels for points that are queried in the course of the AL loop. Thus, we start the workflow by presenting the RGB-colorized point cloud to crowdworkers (in case of extensive point clouds, it is split into sub-clouds, each being shown successively). They are asked to mark one point for each class the system operator (i.e., the expert who outsources labeling) defines when launching the AL loop. A snapshot of the implemented tool, which we refer to as crowd task *Type A*, is given in Figure 3.10*(a)*. On the left is the data panel depicting the colorized point cloud along with an explanation of controls that are required for the first task of finding one representative of each class. To better communicate to crowdworkers that they are facing 3D data, and to resolve possible occlusions, the point cloud rotates slowly around the z-axis. On the right, crowdworkers find a button for each requested class. Our aim is to provide them with the definition of classes without the need for complex descriptions (that they would probably not consider anyway). To achieve this, the button incorporates a pictogram along with the class name. The actual task of each crowdworker is to push one of the buttons and to click a corresponding 3D point in the data panel (ray tracing is applied to deduce the 3D point from the clicked pixel). This will prompt the selected point to be marked by means of the specific class color as well as by an arrow marker. Selecting the same class again will delete the previously chosen point and allows for editing the selection.

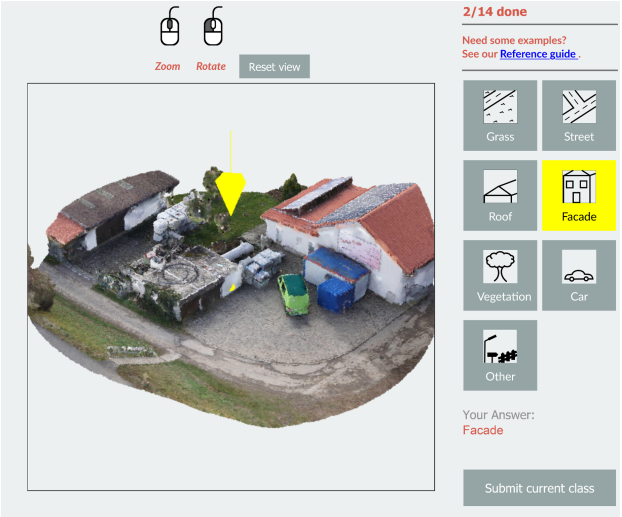
As soon as one representative for every class is found, the crowdworker can submit the result. Thus, our *Type A* tool places relatively high demands on the 3D handling capabilities of our workers (navigation, selection, etc.), but the actual interpretation task is easy. This is due to allowing crowdworkers to freely choose points that are easy to interpret, so that they do not need to deal with edge cases. However, this poses a challenge for quality control because there is no real opportunity to include checks in the task. Also, an integration method such as majority voting after data acquisition is not feasible as it is unlikely that multiple crowdworkers will select the same point.

In order to nevertheless provide the classifier an (ideally) faultless initialization data set (cf. Algorithm 1), we have designed a second tool to detect errors produced in the previous campaign utilizing our *Type A* tool. As can be seen from Figure 3.10*(b)*, the overall design for *Type B* is basically the same as for *Type A*. However, the complexity is significantly reduced as crowdworkers in fact are not required (but allowed) to interact with the data, but only need to decide whether a presented point is assigned to the correct class. For judging, crowdworkers are provided with the indicated point and a 2.5D neighborhood with a radius



(a) Type A: Selecting one point for each class

(b) Type B: Checking labels of selected points



(c) Type C: Labeling selected AL points

Figure 3.10: Compilation of our tools required for the human crowd component within our AL-based framework. Each tool also offers a short introduction video, a task description, and a feedback option, not visualized here. A complete snapshot of the layout of tool *Type A* can be found in the Appendix in Figure A2.

of 20 m, which is again slowly rotating to avoid occlusions in the line of sight (e.g., the tree fragments in Figure 3.10*(b)*). This should be an easy task, both in terms of 3D handling and understanding. Since it can be quickly completed, crowdworkers are asked to label multiple points. Points that are marked as false can then be eliminated, so that crowdworkers using the *Type B* tool are acting as "filters" for the results produced in *Type A* campaigns. The last tool, *Type C*, is designed for labeling points queried within the AL iteration (cf. Section 3.1) and will thus be employed most frequently. It is built very similar to *Type B*, the only difference is that crowdworkers are not asked for a binary decision (*Correct*/*False*), but are expected to solve a multi-class classification problem. The main advantage of the latter two tools is that quality control measures can be easily implemented, as outlined in Section 3.6.3.

3.6.3 Automated Quality Control

In our pursuit to minimize crowd labeling errors, we draw tools from both methods identified by Zhang et al. (2016): i) *quality control on task designing* and ii) *quality improvement after data collection*. As already mentioned, these measures are only valid for our crowd tasks of *Type B* and *Type C* (cf. Section 3.6.2), i.e., where distinct instances are to be categorized. Most crucially, all methods for quality control need to be suited to run automatically, i.e., without manual checks or corrections by an operator, since we aim for a fully automated pipeline.

To meet the need of *quality control on task designing*, a measure easily employed is to present a specific point multiple times for labeling in order to verify that the crowdworker is (at least) working consistently, not necessarily with the correct label. This consistency check can be considered as minimum requirement for successful task submission. However, this procedure does not detect crowdworkers intentionally selecting the same class label for all points (without even considering the data). But such workers can be intercepted by combining consistency tests with check points for which we know the true categorization, i.e., verifiable questions, as recommended by Kittur et al. (2008).

Such check points can be mixed at random positions into real payload tasks. Crowdworkers are only informed *that* there are check points, but not *which* points are actually such points, i.e., they appear just like payload points. Afterwards the accuracy of labels given by crowdworkers for those points is assessed, and we would assume that the remaining point labels are of the same quality. Consequently, only the results of crowdworkers passing all (or a defined number of) checks will be stored and will cause the respective crowdworker to be paid. However, this procedure requires that some points are actually labeled. But in scenarios where we suppose that each crowdworker is allowed to conduct only one labeling task in each iteration step, it is sufficient to have one set of check points in each iteration step, which can be easily and quickly provided by an operator (or, theoretically also by crowdworkers). If we select a set of check points that are rather unremarkable, it might even be possible to use them in multiple steps, because it is unlikely that crowdworkers will remember these points, at least if they work on only a limited number of jobs. Thus, only a few labeled check points might be sufficient for a whole iteration. To summarize, we aim to automatically eliminate gross labeling errors by the two aforementioned measures for *quality control on task designing*.

But even crowdworkers passing the check points might label the actual payload point suboptimally. Especially in our *Type C* tasks, i.e., labeling selected AL points that are often rather complex for interpretation (at least in later iteration steps), a single crowdworker might fail in deriving the true label. This issue can be tackled through *quality improvement after data collection* by applying the principle of the *Wisdom of the Crowds*. In our case, this can be realized by assigning a task to multiple crowdworkers and afterwards aggregating the results automatically by majority voting, again avoiding engagement of an expert in this process. However, in the context of paid crowdsourcing, labeling instances multiple times means a corresponding multiplication of costs. Thus, the number of multiple acquisitions should be kept low, which raises the question of *how many is enough*, as recently answered by Walter et al. (2022b) for car detection in 2D shadings. Since this parameter n_{cw} is rather task-specific, its choice is also required for the concrete task at hand of categorizing points. The specification is to be achieved by conducting a study with real crowdworkers performing the task of focus.

The most straightforward way would be to launch multiple crowd campaigns for a given task, differing by the number of crowdworkers n_{cw}, whose acquisitions are then considered for integration, yielding an overall quality measure for each campaign. More cost-efficiently, we would launch only one crowd campaign, where each job is completed n_{mult} times to receive a pool of results, with n_{mult} being the maximum number of crowdworkers to be considered ($n_{cw} \leq n_{mult}$). From this pool of n_{mult} acquisitions, for each number of workers n_{cw} ($\in [1, n_{mult}]$) to be evaluated, we could draw the respective number of results randomly to derive the corresponding quality value, ignoring remaining acquisitions. Please note that this is equivalent to launching multiple campaigns with a different number of crowdworkers n_{cw}, where the complete pool of all

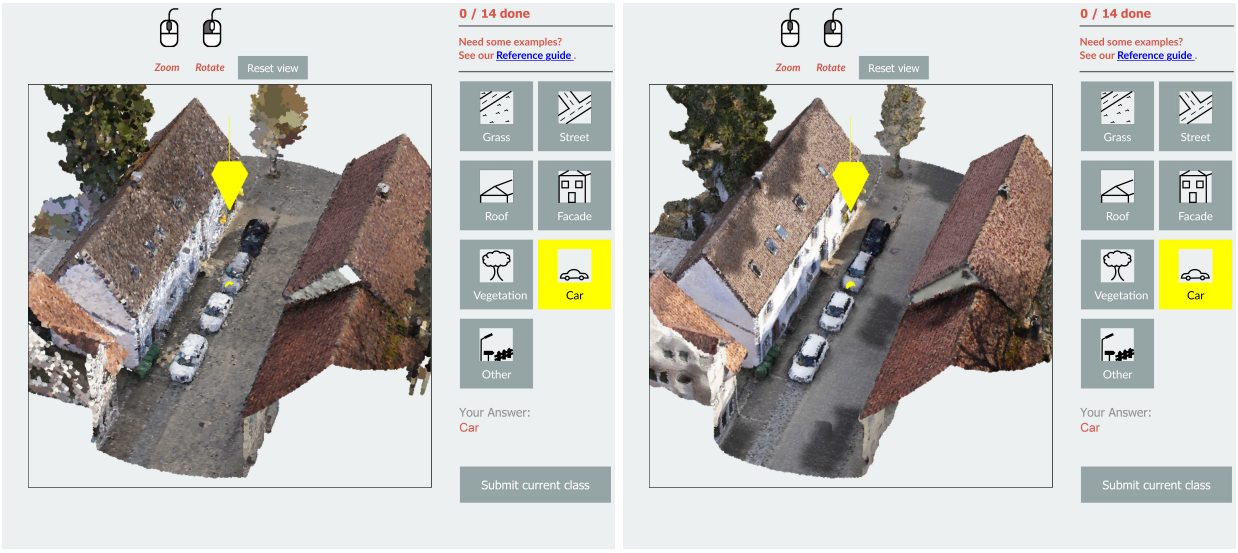
available crowdworkers on a respective platform is sampled vs. the subset of n_{mult} crowdworkers that have contributed to establishing the pool of multiple acquisitions. However, findings based on both approaches would be distorted by results from crowdworkers who perform exceptionally well or badly. For instance, if we randomly pick three low-performing crowdworkers out of ten predominantly high-performing workers, we obtain an integrated result that is far from representative. Thus, we opt to form all possible combinations from the pool of n_{mult} performed crowd jobs for all specific numbers of workers n_{cw} for which we would like to know the expected accuracy. The number of possible combinations is restricted to:

$$n_{comb} = \left(\begin{array}{c} n_{cw} \\ n_{mult} \end{array} \right) = \frac{n_{cw}!}{(n_{cw} - n_{mult})! \cdot n_{mult}!} \tag{3.6}$$

Afterwards, we integrate the results of those crowdworkers and compute the mean value over all combinations. Running such a pre-study before launching large-scale campaigns is worthwhile in order to ensure quality of results by leveraging the *Wisdom of the Crowds*, while minimizing the costs of a given project by avoiding asking an abundant number of workers.

3.6.4 Which 3D Data Representation is Best Suited?

Although we are interested in the semantic segmentation of 3D point clouds and thus require labeled point instances, other data representations can be a viable alternative to be presented to crowdworkers. For instance, projections of 3D data onto a 2D plane to resemble image data, as presented by Walter et al. (2022b), can help non-experts to interpret 3D data, but carry the risk of information being lost due to occlusions. For example, consider a façade point in an urban canyon is queried in the AL loop. In such a case, an orthogonal projection would be suboptimal and crowdworkers would most likely label this point as roof instead. While there might be other projection planes that would depict this point in an easy-to-interpret and occlusion-free manner, it is hard to automatically derive such mappings. Thus, it is rather inevitable to actually present 3D data as such - optimally with an RGB colorization being the most familiar to non-expert labelers.



(a) *(b)*

Figure 3.11: A comparison of feeding either colored point cloud data *(a)* or 3D textured mesh data *(b)* into our labeling tool *Type C* (i.e., categorizing a selected point).

However, instead of colorized point clouds, presenting textured mesh data might be a suitable alternative. Both representation forms are opposed in Figure 3.11. Please note that the automatic creation of such mesh data sets is beyond the scope of this work. The interested reader is referred to Linsen and Prautzsch (2001), Kazhdan et al. (2006), Labatut et al. (2007) and Hiep et al. (2009). In the geospatial domain, a meshed representation is typically obtained by first identifying a subset of 3D points (either LiDAR or photogrammetric points) required to sufficiently describe surfaces. When we are dealing with smooth surfaces, only a few support points are sufficient, whereas many points are required for describing complex and rough objects (e.g., urban furniture). This subset of points defines the set of nodes. Such nodes are connected via edges to form faces with three edges each in case of triangle meshes. Consequently, such a data representation can also help to minimize the memory requirements, as smooth surfaces can be described by few large faces approximating a dense point-based surface description. If imagery data is available as well, the interpretability of 3D meshes can be drastically boosted by enriching the faces with mapped image crops, i.e., by texturing the mesh.

Thus, compared to a colorized point cloud, where colors are limited to discrete point locations, a much more natural and realistic depiction can be achieved, as can be seen in Figure 3.11. However, this is not only due to the discretized nature of the point cloud, but also due to challenges in the post-processing colorization technique, which is required whenever color information is not inherent in the data, as for ALS data. In this regard, projection of widely available (temporally disjoint) orthophoto imagery, as proposed by Charaniya et al. (2004) and Lodha et al. (2006), is a suboptimal choice by design, as occluded points such as façade points obtain color values of the occluding structure (probably roof). If concurrently captured oblique imagery is available, there are many possibilities to obtain more true RGB tuples at point positions. For instance, the ALS point cloud in Figure 3.11 is colorized by interpolating color values from a photogrammetric dense point cloud. Although the result exceeds by far mapping of an orthophoto, still issues arise whenever there are objects that are not included in either the photogrammetric or ALS point cloud, since then interpolation artifacts occur (e.g., the unnaturally colorized tree at the end of the street in Figure 3.11*(a)*). Alternatively, 3D LiDAR points can be projected into all images in whose viewing frustum this point is included to receive an RGB tuple of the hit image pixel. Besides the question of the appropriate aggregation rule of all obtained color values (from all images) for a specific point, possible occlusions are a more severe problem.

Most efficiently, such occlusions can be detected if there is a closed and watertight surface that can be used for ray tracing, i.e., we require a meshed surface. Since a realistic colorization of 3D point clouds is a demanding topic and an optimal solution would already lead to 3D mesh geometry, we advocate for textured 3D meshes as this data representation offers a most realistic continuous colorized surface. We argue that this drastically helps non-experts like crowdworkers in the interpretation of 3D data. The idea is to emulate a representation similar to *Google*'s geospatial services such as *Google Maps* and *Google Earth*, where 3D data is represented by means of 3D textured meshes for geospatial non-experts worldwide. In case of our AL scenario, to visualize 3D data we can utilize the mesh and point cloud data interchangeably. For instance, in our *Type C* labeling tool (cf. Section 3.6.2) where we highlight a point sampled within the AL iteration, we can use either a crop from the point cloud or from a 3D textured mesh (cf. Figure 3.11) to give the crowdworker a visual impression of the surrounding context to enable interpretation of the presented point, resulting in crowd oracles relying either on point cloud data $\mathcal{O}_{CP}$ or mesh data $\mathcal{O}_{CM}$. From a geometric point of view, generating the mesh surface equals a (mild) generalization that can help to suppress noise, but fine-grained details can also be lost. Thus, it may happen that a point selected from the point cloud modality is not included as a node in the mesh geometry. However, facing the significant gain in interpretability due to the fine texturing, this issue is negligible.

3.6.5 Stimulate Motivation by Gamification

Another option for fostering data collection through crowdsourcing is to gamify this collection process. This means we aim to increase the appeal of participating, with the intention to improve the quality of work as well as to increase the amount of work being completed. Ideally, crowdworkers are intrinsically motivated and

(a) Introduction page *(b)* Main game interface

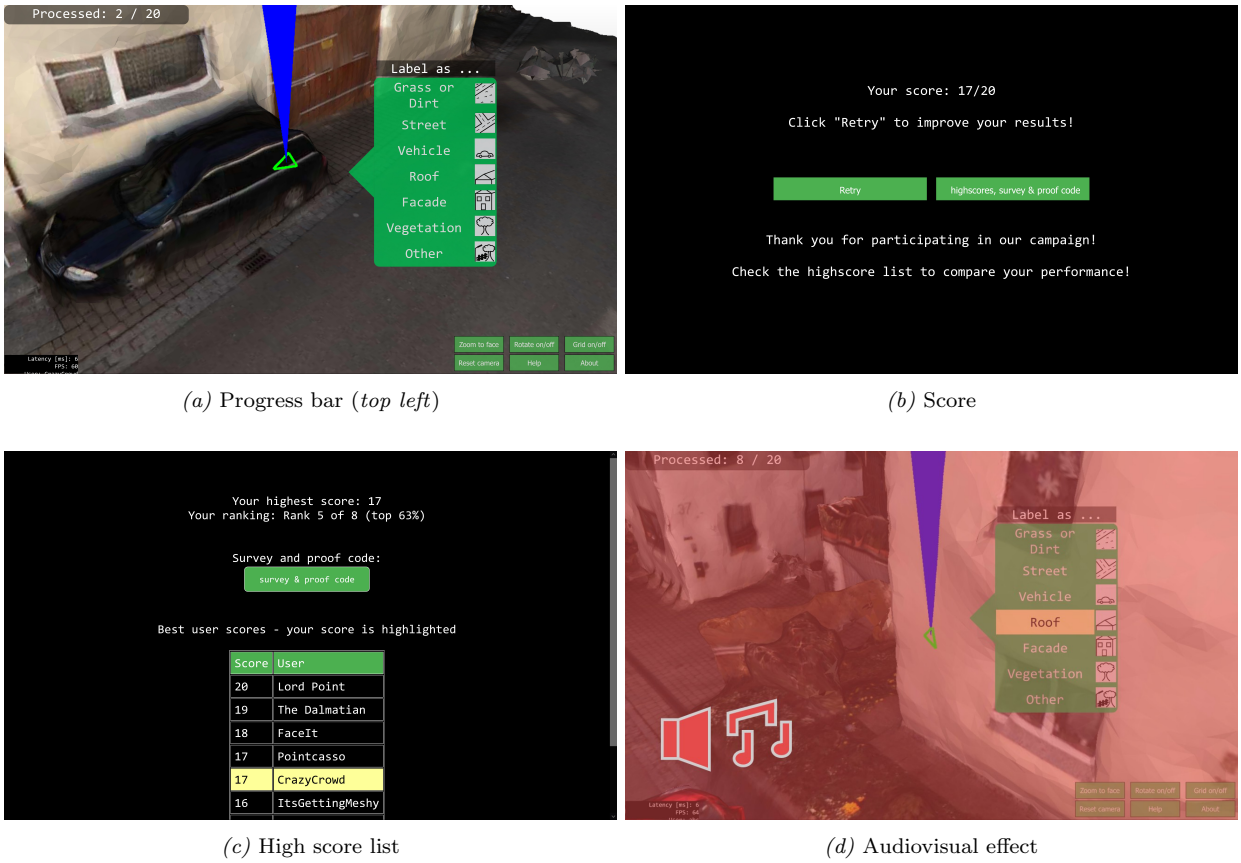
Figure 3.12: Snapshots of our labeling tool with a game-like appeal compared to the basic labeling tool as presented in Figure 3.10*(c)*. Based on this interface, gamification add-ons are implemented (cf. Figure 3.13). Best viewed digitally or in Figure A3 in the Appendix.

enjoy taking part in our campaigns without utilizing gaming elements. But especially in paid crowdsourcing, where the crowdworkers' motivation is mainly of extrinsic nature (i.e., receiving the payment), campaigns can benefit from gamification. To position the concept of gamification in relation to the other sections of this work, we are dealing with a hybrid of creating easy and enjoyable tools (cf. Section 3.6.2) and a measure for *quality control on task designing* (cf. Section 3.6.3), since a higher quality can be expected when crowdworkers are intrinsically motivated. The latter is, of course, limited by the crowdworkers' abilities. To achieve the aspired intrinsic motivation, the challenge is to create a game that is actually appealing, enjoyable, and entices a worker to play *just one more time* (von Ahn and Dabbish, 2008), while completing the rather tedious task of labeling.

In case of geospatial data and especially 3D data, we are fortunately dealing directly with virtual worlds, such as digitized cities by means of discrete point clouds or continuous 3D textured meshes. The latter data modality is indeed typically used in more sophisticated games. Thus, the difficult task of setting up a proper gaming environment is solved directly by using 3D geospatial data from daily practice, which might already stimulate interest of workers (cf. Figure 3.12*(a)*). Eventually, we aim for gamified versions of our labeling tools presented in Section 3.6.2. However, as already mentioned, they differ in complexity. While *Type A* requires navigation in 3D data, *Type B* and *Type C* are much easier to handle.

When playing a game, navigation within the environment is often part of the joy. Yet, this requires a certain skill level, which might not always be met, and certainly a short period of acclimatization to the controls. But crowdworkers performing a job only once are typically not in the mood of learning a new game. Since most of the labeling tools we designed do not require navigation (*Type B* and *Type C*, cf. Section 3.6.2), we decided to first launch a gamified labeling tool without such navigational elements, opting instead for a dynamic camera motion to a selected point to be labeled, or from one point to the next. Precisely, we aim to develop a gamified labeling tool of *Type C*, i.e., label pre-selected entities (e.g., within the course of the AL loop).

We first focus on implementing a more game-like design than the one seen in Figure 3.10*(c)*, which is visualized in Figure 3.12*(b)*. To be able to purely evaluate the impact of gamification elements for 3D data annotation, crowdworkers are tasked with labeling a certain amount of mesh triangles (instead of 3D points) to avoid confusion based on discrepancies between the discretized point cloud and the meshed representation (cf. Section 3.6.4). Additionally, crowdworkers are also given the option to complete as many jobs as they wish. This helps to evaluate the will of workers to conduct more work (for the same payment).



(a) Progress bar (*top left*)

(b) Score

(c) High score list

(d) Audiovisual effect

Figure 3.13: Overview of our implemented gamification elements built upon the labeling tool presented in Figure 3.12. Best viewed digitally or in Figure A4 & Figure A5 in the Appendix.

We enhance this labeling tool with a variety of different gamification elements, as depicted in Figure 3.13. A simple add-on already employed in our non-gamified tools is an indication of crowdworker's progress (cf. Figure 3.13*(a)*). This element is intended to keep crowdworkers motivated, as they can work towards a goal, i.e., completing labeling of all points. In this process, the progress count is supposed to minimize the risk of crowdworkers getting tired and bored when they have lost count of the number of points that were already labeled. As a second measure, we score the labeling result of each worker and openly communicate it to the worker upon completion of the job (cf. Figure 3.13*(b)*). This comes in tandem with stressing again the option to retry. In paid crowdsourcing, this appeals mainly to the will of crowdworkers to maximize their income, since a higher score can lead to a higher payment as well, as we have intentionally vaguely stated in the introduction of the gamified tool (cf. Figure 3.12*(a)*). Thus, this element could lead to crowdworkers doing more work for free, *and* it might also increase the quality since they try hard to achieve a high score. A drawback of a score function for sure is the necessity of already labeled data. Generally, we have to assume that such labeled data are not available at all or only sparsely. The latter applies when an operator provides some check points in the context of our quality insurance measures (cf. Section 3.6.3). However, to evaluate whether such elements have a positive impact and whether it pays off to provide labeled data, we assume in our experiments that labeled data are abundant.

A gaming element directly linked to the previous one are high score lists (cf. Figure 3.13*(c)*). The principle is exactly the same, but now we not only communicate the score achieved, but we also relate it to the score of the other workers. Again, this directly appeals to the wish of maximizing income by expecting that a

higher rank could cause a bonus payment, etc. Finally, our last add-on roots from a different concept, which is again to increase joy but also to give immediate feedback (cf. Figure 3.13*(d)*). This is accomplished by audiovisual effects, i.e., whenever a face has been labeled correctly, the screen flashes green and a prosperous sound is prompted, otherwise the screen flashes red and a losing sound is played.

3.7 Classifiers for 3D Semantic Segmentation

After discussing the human component of our hybrid intelligence system, we now turn to the machine part, the ML model responsible for 3D point cloud segmentation. To demonstrate the general independence of our crowd-driven AL pipeline from the specific classifier employed, we rely on one representative each from both the feature-driven (cf. Section 3.7.1) and the data-driven (cf. Section 3.7.2) domain. In either case, as input we assume to have a point cloud $\mathcal{P}$ being constituted by $n_{\mathcal{P}}$ individual point instances $\boldsymbol{p} = (x \quad y \quad z) \in \mathbb{R}^3$, that may carry additional attributes (such as RGB values). Ultimately, we aim to assign to each 3D point $\boldsymbol{p}$, a unique class label $c \in \{c_1, ..., c_{n_\Omega}\}$, with n_Ω being the number of classes.

3.7.1 Hand-Crafted Features & Random Forest

Defining Geometric and Radiometric Features

For successful classification of ALS point clouds by means of a feature-driven classifier, under careful review of respective literature (cf. Section 2.8.1), we aim to define a set of both geometric and radiometric features to be employed. Since a meaningful description of geometric properties of individual 3D points is infeasible, neighboring points are extracted, which poses the question of how to define the vicinity of each point. For this, we rely on a spherical neighborhood definition (Lee and Schenk, 2002) parametrized by the radius r. This means that all points $\boldsymbol{p} \in \mathcal{P}$ fulfilling $d(\boldsymbol{p}_c, \boldsymbol{p}) < r$, with $\boldsymbol{p}_c \in \mathcal{P}$ being the current point of focus for which a set of features is to be computed, are part of the set of neighboring points $\mathcal{N}$. In order to avoid testing this condition for all $n_{\mathcal{P}}$ points, and to boost the recovery of neighbors, as a pre-processing step, an *octree* structure is utilized, as described by Girardeau-Montaut et al. (2005).

Although Weinmann et al. (2015a) demonstrated that a dynamic neighborhood definition based on the local geometry can be beneficial for the segmentation result (especially in cases of varying point densities), we argue that in case of large-scale ALS point clouds, where typically a rather uniform point density is present, constant neighborhood definitions are a viable alternative minimizing computational effort. For increasing spatial contextuality, we compute (most) geometric features at different scales (i.e., we evaluate the same set of features for multiple values of r), where smaller values of r are intended to capture detailed geometric nuances and larger radii are meant to describe rather the more general, smoothed local context.

For each point (and scale), a first set of geometric features is generated based on the 3D covariance matrix often referred to as structural tensor, which can be obtained from all $n_{\mathcal{N}}$ points located in a specific neighborhood under consideration according to Equation 3.7.

$$\boldsymbol{Cov} = \frac{1}{n_{\mathcal{N}}} \sum_{i=1}^{n_{\mathcal{N}}} (\boldsymbol{p}_i - \bar{\boldsymbol{p}})^T (\boldsymbol{p}_i - \bar{\boldsymbol{p}}) \qquad \text{with } \bar{\boldsymbol{p}} \text{ being the neighborhood's center of gravity} \tag{3.7}$$

From this symmetric, positive-definite matrix, the principal components, i.e., the eigenvectors and their respective lengths, the eigenvalues $\lambda_1, \lambda_2, \lambda_3$ stating the variation of data along the respective eigenvector, can be derived. Eigenvalues are required to be sorted so that $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$. This whole process can be thought of as fitting an ellipsoid to the supporting neighborhood $\mathcal{N}$ (Mallet et al., 2011), where the impact of points increases quadratically with their distance to the core point (Kim and Sohn, 2010; Blomley et al., 2014). Eigenvectors define the orientation and the eigenvalues indicate the extent of the ellipsoid. Please note that this formulation is in accordance with least squares plane fitting, where the eigenvector

Feature Category	Designation	Abbreviation	Definition
Geometric features			
eigenvalue-based	*Linearity*	L	$(\lambda_1 - \lambda_2)/\lambda_1$
	Planarity	P	$(\lambda_2 - \lambda_3)/\lambda_1$
	Sphericity	S	λ_3/λ_1
	Surface Variation	SV	$\lambda_3/(\lambda_1 + \lambda_2 + \lambda_3) = e_3$
	Omnivariance	O	$\sqrt[3]{e_1 \cdot e_2 \cdot e_3}$
	Anisotropy	A	$(\lambda_1 - \lambda_3)/\lambda_1$
	Eigenentropy	E	$-\sum_{i=1}^{3} e_i \cdot ln(e_i)$
	Eigenvalues Sum	ES	$\lambda_1 + \lambda_2 + \lambda_3$
plane-based	*Verticality*	V	$1 - n_z$
	Verticality (from robustly fitted plane)	V_{rob}	$1 - n_z^{rob}$
	Roughness	R	$\|\boldsymbol{p} \cdot \boldsymbol{n} - d\|$, with $d = \boldsymbol{q} \cdot \boldsymbol{n}$
height-based	*Height Above Ground*	Δz	$z - z_{DTM}$
LiDAR-inherent	*Echo Ratio*	ER	echo number/number of echos
Radiometric features			
LiDAR-inherent	*Reflectance/Intensity*	Re/I	
	Mean Reflectance/Intensity	$\bar{Re}/\bar{I}$	$\text{mean}(\{\boldsymbol{Re}_{\mathcal{N}} \vert \boldsymbol{I}_{\mathcal{N}}\})$
imagery-based	*Color Values*	H, S, V	
	Mean Color Values	$\bar{H}, \bar{S}, \bar{V}$	$\text{mean}(\{\boldsymbol{H}_{\mathcal{N}} \vert \boldsymbol{S}_{\mathcal{N}} \vert \boldsymbol{V}_{\mathcal{N}}\})$

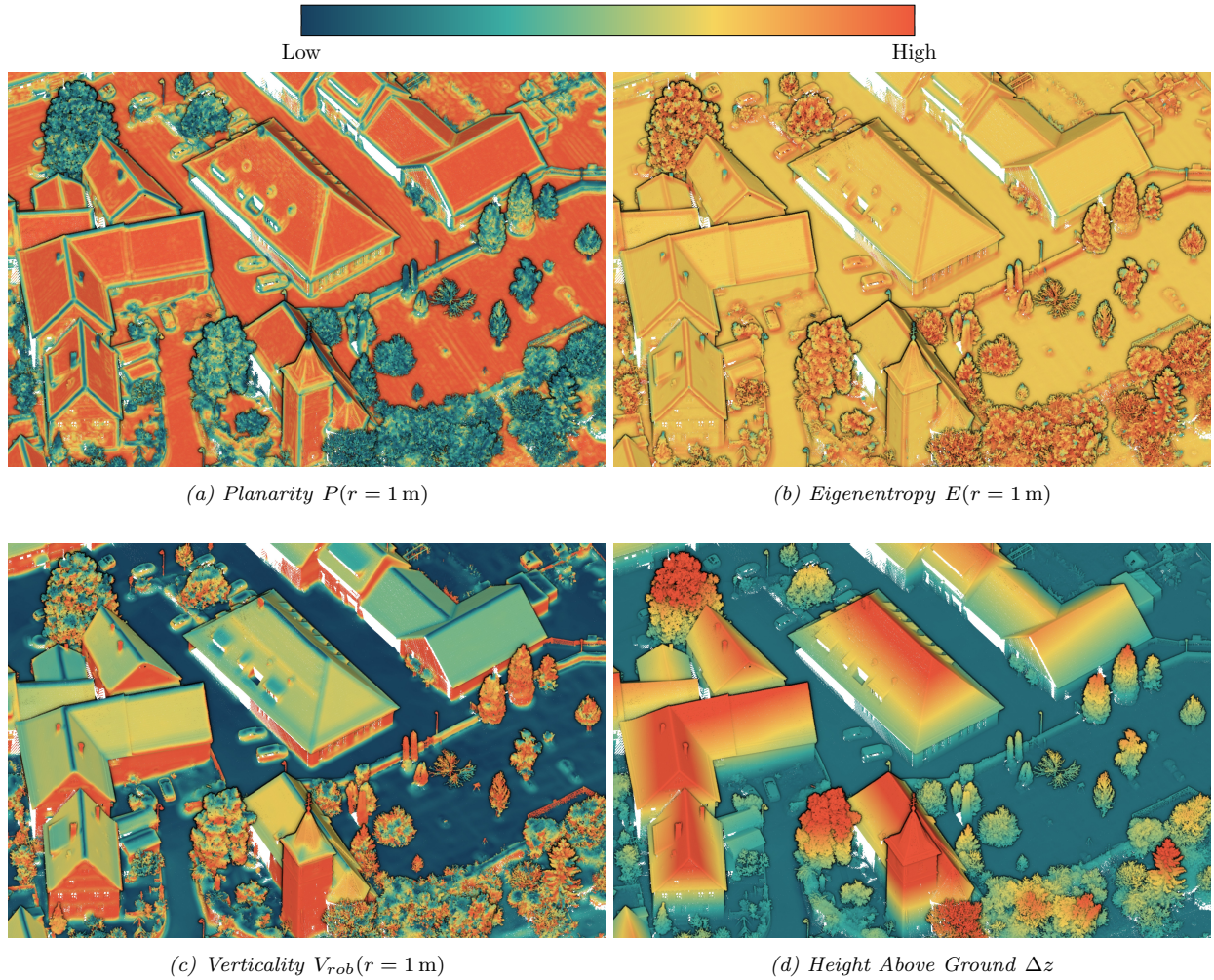
Table 3.1: Features used as input for the RF classifier. Relative importance of individual features can be found in Figure A6 in the Appendix.

with the smallest eigenvalue represents the normal vector of the plane (Shakarji, 1998; Waldhauser et al., 2014). Derived eigenvalues can be employed for defining a set of features (cf. Table 3.1) which exhibit typical values for certain structures (Jutzi and Gross, 2009; Dittrich et al., 2017). For some features, a normalization of eigenvalues is required ($e_{1/2/3} = \lambda_{1/2/3} / \sum_{j=1}^{3} \lambda_j$). A favorable property of the eigenvalues, and hence the features based on them, is their invariance to rotation, scale, and shift (Maas and Vosselman, 1999; Jutzi and Gross, 2009), allowing robust comparability.

Linearity, *Planarity* and *Sphericity* (also denoted as dimensionality features) each rely on the ratio of certain eigenvalues to describe the resemblance of $\mathcal{N}$ to a linear, planar and spherical distribution, respectively (cf. Figure 3.14*(a)*). This is also the case for *Surface Variation*, indicating spherical point distributions and *Anisotropy*, allowing cylindrical and plane support regions to be distinguished from spherical ones (cf. Table 3.2). *Omnivariance* and *Eigenentropy* are rather noise-robust geometric descriptors (Dittrich et al., 2017),

point distribution	ratio of eigenvalues	L	P	S	SV	O	A	E
cylindrical	$\lambda_1 \gg \lambda_2 \approx \lambda_3$	≈ 1	≈ 0	≈ 0	≈ 0	$\ll 1/3$	≈ 1	$\ll -\ln(1/3)$
planar	$\lambda_1 \approx \lambda_2 \gg \lambda_3$	≈ 0	≈ 1	≈ 0	≈ 0	$< 1/3$	≈ 1	$< -\ln(1/3)$
spherical	$\lambda_1 \approx \lambda_2 \approx \lambda_3$	≈ 0	≈ 0	≈ 1	$\approx 1/3$	$1/3$	≈ 0	$-\ln(1/3)$

Table 3.2: Characteristic values of eigenvalue-based features derived from eigenvalues λ_{1-3}, as described in Table 3.1.



(a) Planarity $P(r = 1\,\mathrm{m})$ *(b) Eigenentropy* $E(r = 1\,\mathrm{m})$

(c) Verticality $V_{rob}(r = 1\,\mathrm{m})$ *(d) Height Above Ground* Δz

Figure 3.14: Exemplary evaluation of geometric features from feature groups in Table 3.1 for the H3D data set (cf. Section 4.2.2).

capable of providing typical values for all point configurations at once, resulting in a clear separability between plane surfaces (e.g., roofs or streets), irregular point distributions (e.g., vegetation) and linear point configurations (e.g., lamp posts or antennas), as depicted in Figure 3.14*(b)*. Finally, the *Sum of Eigenvalues* allows to distinguish between different neighborhood characteristics of the same configuration, i.e., cylindrical, planar or spherical (and is hence not listed in Table 3.2), as for instance, a more widespread irregular configuration (i.e., lower volumetric density) yields a higher value for this feature than a more dense layout.

While the aforementioned invariance of such eigenvalue-based features is generally a desired property of features, it is not necessarily an advantage for the interpretation of ALS point clouds, as objects of specific classes in our environment adapt a typical orientation in space (e.g., vertical surfaces). Therefore, we also measure the *Verticality* (proven to be a distinctive feature by Mallet et al. (2008), Chehata et al. (2009) and Weinmann et al. (2013)) depending on the z-component of the local normal vector $\boldsymbol{n}$, based on one hand on the plane derived within the computation of covariance features, and on the other hand on a more robustly fitted plane to minimize the impact of fine-grained details or noise. The latter can be achieved,

for instance, by RANSAC-supported (Fischler and Bolles, 1981) plane fitting (cf. Table 3.1 and depicted in Figure 3.14*(c)*). The set of plane-based features is complemented by deriving the local *Roughness*, which expresses the distance of a 3D point $\boldsymbol{p} \in \mathcal{N}$ to the best-fitting estimated plane from $\mathcal{N}$ (cf. Table 3.1, where $\boldsymbol{n}$ is the normal vector and $\boldsymbol{q}$ is the support vector of the plane) (Chehata et al., 2009). Thus, we expect it to highlight planar surfaces (e.g., roofs and streets) over unordered structures (e.g., vegetation).

Since height-based features have also proven to be distinctive (Chehata et al., 2009; Weinmann et al., 2013; Niemeyer et al., 2014), the last purely geometric LiDAR-independent feature derived is the height of individual 3D points above ground level Δz, visualized in Figure 3.14*(d)* (Kim and Sohn, 2010; Niemeyer et al., 2012). To do so, we require a reference height of a DTM whose estimation in fact also incorporates a classification step (*Ground* vs. *Non-Ground*). However, it can be obtained before the actual targeted semantic segmentation by purely geometric considerations (Sithole and Vosselman, 2004), such as implemented within the *SCOP++* software (Pfeifer et al., 2001). Alternatively, it may also be available from NMAs (Charaniya et al., 2004). If respective tools or data are not accessible, the local ground level can also be approximated by relying on the minimum height value within a cylindrical neighborhood (Chehata et al., 2009; Mallet et al., 2011; Guo et al., 2015). However, this approach is limited for steep surfaces.

The aforementioned features are based solely on respective 3D coordinates, which in ALS are the result of polar measurement where the required range values are obtained either from analyzing the received echo via hardware components or from the analysis of the densely discretized received echo waveform (Mandlburger et al., 2019). The latter allows deriving even more descriptors (Mallet et al., 2011). However, many laser scanning systems make use of online waveform processing, with the recorded echos being the result of Full-Waveform (FWF) processing, but where the original echo waveform is not necessarily accessible. Thus, we limit ourselves to universally valid features based solely on available returns, so that they are applicable to both evaluation concepts. Since the emitted laser pulse illuminates an extended area in object space (due to beam divergence), a single emitted laser pulse can hit multiple objects/targets, resulting in multiple echos. Hence, laser scanning is notorious for its capability of "penetrating" vegetation. This property is exploited for deriving the *Echo Ratio* (cf. Table 3.1), being a good indicator of vegetation points (cf. Figure 4.4*(a-c)* in the next chapter, in which we discuss features that are inherently obtained due to the measurement principle).

In addition to the pure detection of the presence of echos, the strength of the reflected signal is also recorded. The power of the received echo is highly dependent on the target's spatial extension, its reflectivity, and its directionality of scattering. However, it is also determined by the target range and beam divergence (thus defining the effective footprint), the size of the aperture and also by atmospheric conditions. Analytically, this functional dependency is expressed by means of the Radar/LiDAR equation (Wagner et al., 2006). Pure intensity values, typically found in ALS point clouds, are strictly in accordance to this equation, although it would be more desirable to have a property solely determined by target characteristics, i.e., solely by the backscatter cross section. Therefore, manufacturers such as *RIEGL* often apply range corrections to raw intensity measures calibrated with respect to a flat white surface orthonormal to the laser beam and of extent of the footprint. This measure, referred to as *Reflectance*, as well as *Intensity* values are exemplarily visualized in Figure 4.4*(d-f)* in the next chapter. Since pure *Intensity*/*Reflectance* values are often noisy, or more generally, radiometric measures often tend to be noisy, Guo et al. (2015) and Becker et al. (2017) suggest applying spatial smoothing of respective measures within a defined support region. Hence, for each point, we compute the mean value of those readings, considering the same neighborhoods $\mathcal{N}$ as for the geometric features (cf. Table 3.1).

Since humans are capable of distinguishing between different objects not only based on the observed geometry, but also on individual RGB colors, we also aim to include this kind of information in our feature vector. However, color information is not inherent in LiDAR data, thus requiring mapping of RGB values from imagery, leading to the challenges discussed in Section 3.6.4 (Chapter 4 discusses how this challenge is met for our individual data sets). Providing a colorized point cloud, we follow the suggestion of Becker et al. (2017) and transform RGB values to the HSV (Hue, Saturation, Value) color space (Smith, 1978). Just

as for the *Intensity/Reflectance* values, we also apply gaussian smoothing for those radiometric features (cf. Table 3.1).

Finally, all features described in Table 3.1 and evaluated for each utilized scale are concatenated into the feature vector $\boldsymbol{x}$ of size $n_{feats} = 15 \cdot n_S + 6$ (6 features are independent of a neighborhood $\mathcal{N}$), with n_S being the number of scales. All elements x_f of the feature vector $\boldsymbol{x}$ are standardized according to $x'_f = (x_f - \mu_f)/\sigma_f$, with μ_f denoting the mean and σ_f denoting the standard deviation of each feature before being used as input for a classifier.

Random Forest Classification

For the actual subsequent classification step, we choose an RF classifier for it has proven to provide a good trade-off between accuracy and efficiency (Weinmann et al., 2015a). The RF classifier is a supervised, non-probabilistic, discriminative classifier and represents a *Bagging* (***B**ootstrap **Agg**regat**ing***) model (Breiman, 1996). The basic idea is to combine a set of n_e typically weak classifiers, Decision Trees in case of RF, to form a strong overall classification hypothesis by simple majority voting according to Equation 3.8.

$$\boldsymbol{p}(\boldsymbol{c}|\boldsymbol{x}) = \frac{1}{n_e} \sum_{e \in \mathcal{E}} \boldsymbol{p}_e(\boldsymbol{c}|\boldsymbol{x}) \tag{3.8}$$

The RF as ensemble of Binary Decision Trees (each predicting a pseudo probability $\boldsymbol{p}_e(\boldsymbol{c}|\boldsymbol{x})$) is constructed by the combination of Bootstrapping (Efron, 1979) with the Classification and Regression Trees (CART) algorithm for building Binary Decision Trees in an automatic manner (Breiman et al., 1984). This means that n_e times an independent subset of the available training data is sampled randomly (*Random* Forest) with replacement, each subset being used as input for establishing a Decision Tree by means of CART. In CART, first only 1/3 of available training data is used for generating the structure of a Decision Tree. To establish a first split of data, a random set of n_{feats}^{split} features (typically defined as $n_{feats}^{split} = \log_2(n_{feats})$) is selected, in contrast to the basic model of Decision Trees only relying on one feature at a time and therefore only allowing for axis-parallel decision borders. Those features are used to define separating hyperplanes $\boldsymbol{w}^T \cdot \boldsymbol{x} + w_0 = 0$, with $\boldsymbol{w}$ being the normal vector of the hyperplane with its components $\in [-1, 1]$ and $w_0 \in [\min(\boldsymbol{w}^T \cdot \boldsymbol{x}), \max(\boldsymbol{w}^T \cdot \boldsymbol{x})]$ denoting the offset to the origin. Respective values for both parameters are selected randomly. To assess whether such a randomly set hyperplane is a reasonable choice, the impurity of the current split E_{split} (Equation 3.9), expressed through *weighted entropy* of the resulting subspaces, is evaluated and minimized (please note that this formulation equals maximizing the information gain G, i.e., $G = E_{prev} - E_{split}$, with E_{prev} denoting the impurity of the parent node).

$$E_{split} = -\frac{n_1}{n_1 + n_2} \sum_{i=1}^{n_\Omega} p_1(c_i) \cdot \log_2 [p_1(c_i)] - \frac{n_2}{n_1 + n_2} \sum_{i=1}^{n_\Omega} p_2(c_i) \cdot \log_2 [p_2(c_i)] \tag{3.9}$$

In Equation 3.9, n_1 and n_2 represent the number of samples of each subspace of the (reduced) feature space separated by the hyperplane, and $p_1(c)$ and $p_2(c)$ can be approximated by simply creating normalized class histograms for the two subspaces. In case of a class-imbalanced training set (which is typically the case for ALS data), individual class weights can be considered by giving each sample a weight inversely proportional to the relative occurrence of its assigned class in the complete training set. This weight is taken into account when creating the normalized class histogram, thus affecting E_{split} and finally the selected hyperplane. After determination of the first hyperplane, the process is repeated in a recursive manner for each child of the current node until full purity. To abort the process, the maximum tree depth d^{max} and the minimum number of samples $n_{samples}^{min}$ at a node justifying another separation can be used. Following building the structure of a tree, the remaining and so far untouched 2/3 of the training set is run through the tree. For each leaf of the tree, a normalized histogram of classes is constructed for all training points populating this node, which is the very pseudo-probability utilized in Equation 3.8. Thus, the output of the

RF classifier can be interpreted in a probabilistic manner (although being a non-probabilistic classifier). Due to the multitude of random decisions made in the process of construction of each tree, the resulting output of the RF is well generalizable and robust to noise (Breiman, 2001).

Hence, the RF incorporates a heuristic similar to the *Wisdom of the Crowds* principle, where diverse non-expert individuals (i.e., weak individual Decision Trees) independently draw conclusions (based on independently sampled training sets) which are aggregated (via majority vote) to form a hypothesis outperforming those of individuals (Bernardo, 2022).

In order to employ the RF in the AL scenario as presented in Algorithm 1, different requirements must be met:

i) The classifier is supposed to be able to learn from a sparsely labeled data set. For a feature-based classification scheme, this requirement is easily satisfied since we only need to provide our classifier with tuples of feature vectors and corresponding labels. In case of a sparsely labeled training set, this consequently only leads to a smaller number of such training samples, as unlabeled points are ignored. Nevertheless, they still contribute passively to feature computation since they may be part of a local neighborhood $\mathcal{N}$. Thus, feature vectors of labeled points are independent of whether we are confronted with a fully or sparsely annotated data set.

ii) We are relying on our classifier to reliably assess uncertainty since the sampling of points is based on it. Especially, we are interested in epistemic uncertainty as this is the one we are able to minimize by means of AL. We argue that an RF is able to capture epistemic uncertainty (Shaker and Hüllermeier, 2020) since it is an ensemble model based on *Bagging* and results in different feature space separation hypotheses. This is firstly due to randomly sampling an independent subset of available training data, and secondly due to randomly selecting features used for separation in each tree. Thus, multiple valid but deviating models will be obtained, and the disagreement between these different models can be considered as measure of epistemic uncertainty. The generated tree-wise posterior probabilities are equally suited to support *vote entropy* sampling (cf. Equation 3.2) or *entropy* sampling (cf. Equation 3.1) based on the averaged probabilities (cf. Equation 3.8).

iii) For guaranteeing diversity within batches (cf. Section 3.3.3), we require features for each point to measure the similarity with previously sampled points. Since we are dealing with a feature-based classifier, providing features is straightforward.

3.7.2 Submanifold Sparse Convolutional Neural Network

As mentioned earlier, we aim to contrast a representative of the data-driven domain to our RF classifier. Among the vast pool of recently introduced CNNs for semantic segmentation of 3D data (Griffiths and Boehm, 2019; Guo et al., 2021; He et al., 2021), we choose the SCN (Graham, 2015; Graham et al., 2018) as it achieves top-notch results for demanding state-of-the-art benchmark data sets (Thomas et al., 2019), therefore also being embedded in *Google's TensorFlow 3D* framework (Huang, 2021). Furthermore, it performs favorably for our task at hand of semantically interpreting 3D geospatial data (Li et al., 2021), which is further underlined by attempts on geospatial benchmark data sets like V3D (Schmohl and Sörgel, 2019) and ISPRS Hessigheim 3D Benchmark on Semantic Segmentation of High-Resolution Point Clouds and Meshes (H3D) (Kölle et al., 2021a) while minimizing computation time. This property is especially beneficial since the classification model will be trained and used for prediction repeatedly in the course of the AL loop (cf. Algorithm 1).

As CNNs were originally developed for 2D raster data, i.e., regular grid data, as a first step we voxelize the input point cloud $\mathcal{P}$. This means a regular 3D grid structure with a specific cell size d_v is superimposed over the point cloud data. Thus, a rather sparse 3D voxel structure can be obtained (cf. Figure 3.15), where

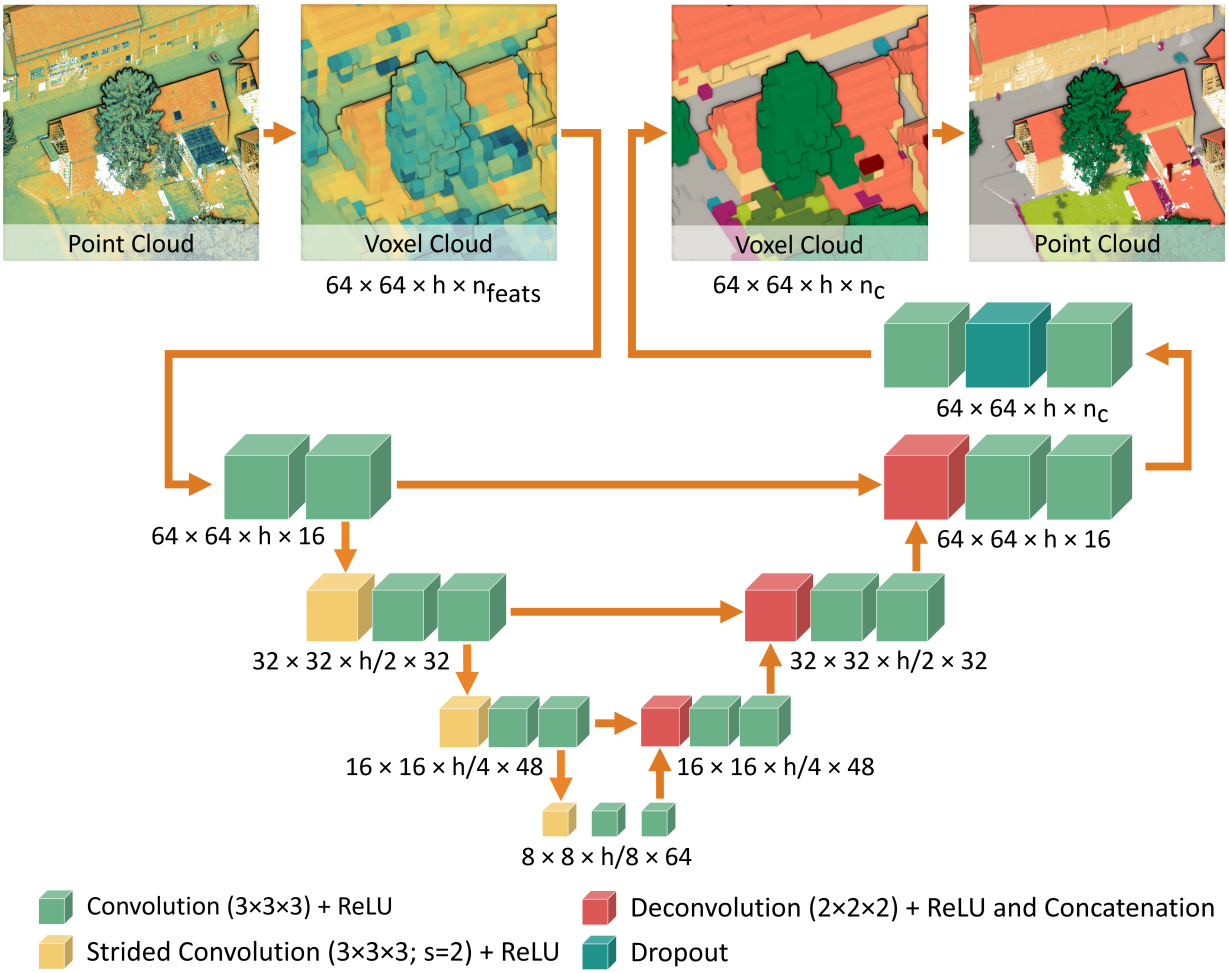


Figure 3.15: The SCN relies on voxelized point clouds (in this case colored according to *Reflectance*), which are fed into a *U-Net* architecture to obtain voxel-wise labels to be subsequently mapped to the original point cloud data. Respective dimensions of each level state the current shape of the data, where the first three values correspond to the resolution and the last value to the number of filter responses from sparse convolutions (i.e., features). The more coarse the resolution of data, the more filter kernels can be employed to gain a deep understanding of data.

point positions are implicitly encoded in the form of active or inactive grid cells. Please note that if no point-wise (or more precisely voxel-wise) input features are available, a pseudo-feature is generated assigning the value 1 to each active cell to allow for geometric feature extraction by means of meaningful convolution and thus classification. In case of presence of such input features (like RGB colors or LiDAR inherent features), the aggregated voxel feature is obtained by averaging over all 3D points included in the respective cell. For the labels, this translates to a majority vote (or, more precisely, plurality vote). Although voxelization poses the risk of losing information due to discretization, we argue that it is a necessary step to achieve reasonable performance - especially with respect to AL, where the classifier needs to be trained multiple times. In this regard, voxelization is closely related to spatial subsampling (where a minimum distance between each point is guaranteed), as often pursued in context of feature-driven classifiers (Brodu and Lague, 2012).

In case of geospatial ALS data, we are rather dealing with an extensive surface (i.e., the earth's surface) with superstructures such as vegetation or buildings. Hence, it significantly differs from other 3D data like indoor data or data representing individual closed objects (e.g., dogs, cats, chairs, etc.), as most of the resulting 3D grid cells are empty (inactive), i.e., there are no 3D points in the respective cells. Thus, to minimize both computational effort and memory consumption, the sparsity of data should be exploited, which is the first challenge in CNN-based processing of 3D data.

The second challenge is how to apply the core functionality of CNNs, the convolution, to 3D data. In case of the SCN, both challenges are met by building upon an alternative formulation of convolving a specific kernel W over given input data X (either being the actual raw input data or the result of an intermediate layer of the CNN) according to $Y = X * W$ as matrix multiplication $\boldsymbol{Y} = \boldsymbol{X} \cdot \boldsymbol{W}$, which is typically used for efficient implementation of convolutions for GPUs (Chellapilla et al., 2006; Chetlur et al., 2014). For the matrix multiplication, $\boldsymbol{W} \in \mathbb{R}^{f^3 \cdot n_{ch} \times n_f}$ comprises all learnable weight parameters (including biases) and contains one column for each of the n_f filters of size f (or more precisely $f^3 \times n_{ch}$) applied to the input with n_{ch} channels. $\boldsymbol{X} \in \mathbb{R}^{n_{conv} \times f^3 \cdot n_{ch}}$ includes the reorganized input data (or output of the previous layer), where each row corresponds to one of the n_{conv} specific convolutions executed at each input location to cover the complete data. This reformulation can be applied to any CNN to minimize the number of basic mathematical operations required (multiplications and additions), and indeed is commonly used to boost GPU-based processing (Chellapilla et al., 2006).

The reformulation of a basic CNN into a *Sparse* Convolutional Neural Network is based on filtering out convolutions operating *only* on empty cells. This means with given filter size, such obsolete convolutions can be identified and respective rows in $\boldsymbol{X}$ can be eliminated. To assign the result to the true position in the resulting activation map, hash tables are created (Graham, 2015). Please note that the previously described steps only serve to reduce the computational load. Nevertheless, even with sparse input, the number of active cells increases with the number of layers.

Finally, a *Submanifold* Sparse Convolutional Neural Network (SCN) also omits convolutions, but this time only those convolutions are performed where the filter matrix is centered on non-empty input cells (but convolutions are performed by matrix multiplications nevertheless), and the result is assigned to those active cells by means of hash tables (Graham, 2015; Graham et al., 2018). This leads to a drastic decrease of n_{conv} compared to a *Sparse* Convolutional Neural Network. However, this formulation does not correspond to the conventional definition of convolution anymore, but is to be considered an approximation for the sake of boosting performance. But the authors argue that aggregation operations (e.g., pooling or strided convolution) allow propagation of information in the network, so that independently processed cells communicate implicitly.

These submanifold sparse convolutions are the core feature of the SCN architecture, visualized in Figure 3.15, which is a slightly adapted version of the one presented by Graham et al. (2018). As previously motivated, we aim at semantic segmentation, meaning that our model is to predict posterior probabilities $\boldsymbol{p}(\boldsymbol{c}|\boldsymbol{x})$ for each point instance. This can be achieved by an encoder-decoder architecture preserving the original shape of the input, such as the state-of-the-art *U-Net* architecture (Ronneberger et al., 2015). As can be observed from Figure 3.15, the *U-Net* consists of two symmetrically arranged branches, an encoding and a decoding branch, each being constituted by multiple (depth) levels. The basic idea is to abstract input data to a deep representation/understanding (situated at the bottom of the U) where spatial information is mainly lost. At this point a classification model would predict the class label (e.g., *dog* in image) without identifying those input values (i.e., pixels) that contributed to this decision. This means we strive to only extract the essence of the input data, while fine-grained details and noise are suppressed. This deep representation is afterwards upsampled back to the original input shape, but instead of input channels, posterior class probabilities are derived. Recalling the example of identifying dogs in images, now we would be able to detect those pixels that contributed to the decision that this image represents a dog (i.e., "dog pixels" in the sense of semantic segmentation).

In each level, a series of batch normalizations (Goodfellow et al., 2016, p. 309), submanifold sparse convolutions, and activation functions are applied (cf. Figure 3.15). Precisely, each level consists of two sparse convolution blocks, each preceded by a batch normalization and followed by an activation layer. In this sequence, batch normalization of the output of individual filters (i.e., features) before inserting into the next convolutional block serves to achieve faster convergence of the training process by encouraging a uniform loss landscape. The subsequent convolutional layer aims to infer a meaningful descriptor for the provided data, where kernels in early layers can be interpreted as low-level descriptors (e.g., edge filters in context of 2D imagery) and later kernels as high-level object descriptors (e.g., focusing on detecting specific objects such as a dog's snout in context of 2D imagery). To allow for a non-linear separation of data, non-linearities by means of the non-linear Rectified Linear Unit (ReLU) activation function $h(\mathrm{x}) = \max(0, \mathrm{x})$ (Goodfellow et al., 2016, p. 187) are introduced.

The gradual reduction of the resolution of data in the encoding branch to allow for employment of an increasing number of filters to gain a deep understanding is accomplished by max-pooling layers in the original *U-Net* definition. However, reducing the resolution and thus increasing the receptive field can also be achieved by setting the stride of a convolutional layer to a value > 1, in our case to 2. This means that convolution is performed along all axes only for every second cell, effectively halving axis dimensions. Such a strided convolution layer is used as first layer in each encoding level (also preceded by batch normalization and followed by a ReLU layer). In the decoding branch, restoring the shape of each opposing level of the encoding branch is accomplished by deconvolution layers, intended to learn individual up-convolutional filters (also preceded by batch normalization and followed by a ReLU layer). This allows for avoiding the definition and possible restrictions of conventional approaches like nearest-neighbor or bi-linear interpolation. Spatial information/context is restored by explicit connections between the encoding and decoding branch. Precisely, the output (i.e., the activation maps) of an encoding level is concatenated with the corresponding deconvolved state of the network and used as input for the layers of a specific level. For better generalization and to avoid overfitting, we add two final convolutional layers with 50 % dropout (Goodfellow et al., 2016, p. 251) in between before finally mapping the number of channels (16, cf. Figure 3.15) to the number of classes n_c.

For actually training the network and determining kernel values, labeled point cloud data is propagated through the network, and the resulting posterior probabilities are compared against GT labels. However, memory consumption prohibits the usage of a complete point cloud (of typical spatial extent) at once. Hence, from the complete data set, we create subsets of sufficient size to characterize representatives of individual classes in context with their typical environment, e.g., in our case $32\,\mathrm{m} \times 32\,\mathrm{m}$ in horizontal dimension. In conjunction with the voxel size of $d_v = 0.5\,\mathrm{m}$ that we will employ in our experimental section (cf. Section 5.1.2), this leads to a voxelized sample dimension of $64 \times 64 \times h$, where h represents the required number of voxels to cover the whole vertical extent of the subset (cf. Figure 3.15). Please note that this varying input size does not affect individual filter kernels, but merely the shape of data within our network.

The pool of samples is enhanced by sampling with overlap (e.g., 30 % in our case) and offline data augmentation for better generalization by rotating the input point clouds at intervals of 30 ° before voxelizing each permutation. This way, a pool of training samples is formed from which subsets are used to constitute mini-batches. These mini-batches are sequentially propagated through the network, and the loss L is computed according to Equation 3.10. We employ an element-wise cross-entropy loss (Goodfellow et al., 2016, p. 73) by comparing the elements $y_c(\boldsymbol{x})$ of the one-hot-encoded reference label (array of size $n_\Omega \times 1$ with value 1 at index of true class and 0 otherwise) to the predicted posterior probability $p(c|\boldsymbol{x}, \boldsymbol{\Theta})$ for each class c, that is based on the network parameters $\boldsymbol{\Theta}$, and summing the cell-wise loss over all active cells $\boldsymbol{x} \in \mathcal{V}^m$ in the current sample m, again summed over all M samples of the current mini-batch, and then normalized by Z (please note that in this section $\boldsymbol{x}$ refers to the input feature vector of voxels rather than points).

$$L = -\frac{1}{Z} \sum_{m=1}^{M} \sum_{\boldsymbol{x} \in \mathcal{V}^m} \sum_{c=1}^{n_\Omega} w(\boldsymbol{x}) y_c(\boldsymbol{x}) \log(p(c|\boldsymbol{x}, \boldsymbol{\Theta})) \quad \text{with} \quad Z = \sum_{m=1}^{M} \sum_{\boldsymbol{x} \in \mathcal{V}^m} w(\boldsymbol{x}) \tag{3.10}$$

To account for typical class imbalance in ALS data, samples are weighted by the reciprocal relative occurrence of the class they belong to, i.e., $w(\boldsymbol{x}) = 1/f(GT(\boldsymbol{x}))$, as suggested for the *U-Net* architecture by Ronneberger et al. (2015). Based on the mean loss of the mini-batches, the network parameters are updated by means of stochastic gradient descent with momentum and weight decay. After every mini-batch has gone through the optimization step once, i.e., one epoch has passed, the procedure is repeated until convergence is reached (when the best previously achieved validation accuracy is not exceeded for a specific number of epochs). The size of the mini-batches is a compromise between having smooth updates based on a large enough number of representative samples alleviating individual noise and, on the other hand, memory consumption. After learning the network parameters, prediction can be easily accomplished by voxelization of a given point cloud (without forming subsets) and propagating it as a whole through the network, so that the data shapes stated in Figure 3.15 (first three values) no longer hold true, but learned filters can be applied to any input shape. To obtain point-level predictions, voxel-level posterior probabilities are inherited to all points included in respective voxel cells.

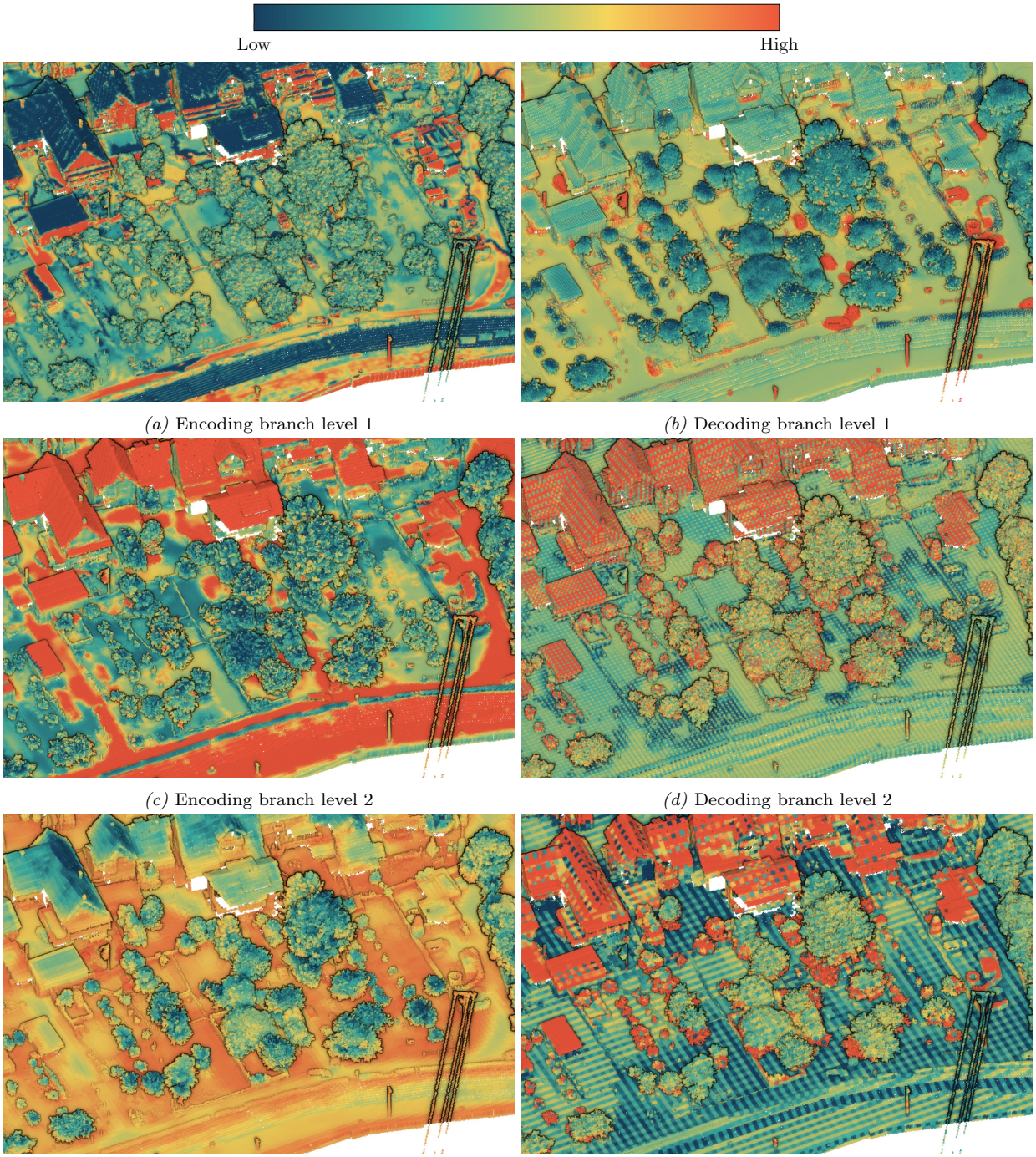
When this SCN classifier is to be employed in AL, just like for the RF classifier in the previous section, several requirements must be met:

i) The training is to work with sparse labels, i.e., feature learning needs to operate on input point clouds where only few points are actually labeled and where a background class label is assigned to all non-labeled points. This affects the label assignment to voxels, since the determination of voxel-wise GT values is no longer based on plurality vote of all points in the voxel, but solely on all *labeled* points (provided that at least one non-background label is present). The learning process itself does not differ from PL, but the loss needs to be modified such that background voxels do not contribute to loss computation, i.e., active cells $\boldsymbol{x} \in V^m$ in Equation 3.10 now only describe active *and* labeled cells. Nevertheless, unlabeled active cells are still a crucial part of the training procedure and contribute to training due to their passive presence for learning geometric descriptors.

ii) Our network needs to be able to reliably assess its predictive uncertainty. However, CNNs tend to underestimate their uncertainty in case only one single network is employed because they lack the ability to estimate epistemic uncertainty (Gal and Ghahramani, 2016). A solution to this end are Bayesian CNNs, which are however often intractable to be evaluated. Thus, an approximation of such Bayesian CNNs by *Monte Carlo dropout ensembles* or *deep ensembles* is a viable alternative (Jospin et al., 2022). Since Schmohl and Sörgel (2019) have demonstrated that utilizing a SCN as base engine for a *deep ensemble* can significantly boost performance, we opt to exploit it also within our AL scheme. Hence, in each iteration step, multiple networks are trained, each being parametrized exactly the same and trained with the complete available training set, as favored over training on subsets by means of *Bagging* (Lakshminarayanan et al., 2017). The models, however, differ due to random initialization of the network parameters. Thus, each network derives a posterior probability $\boldsymbol{p}(\boldsymbol{c}|\boldsymbol{x})$, which can either be used for *vote entropy* sampling (cf. Equation 3.2) or can be inserted into the *entropy* sampling (cf. Equation 3.1) formula as the average of n_e individual class scores of the members of the ensemble $\mathcal{E}$ (cf. Equation 3.11).

$$\boldsymbol{p}(\boldsymbol{c}|\boldsymbol{x}) = \frac{1}{n_e} \sum_{\boldsymbol{e} \in \mathcal{E}} \boldsymbol{p}_e(\boldsymbol{c}|\boldsymbol{x}) \tag{3.11}$$

iii) For allowing diverse sampling of points, as described in Section 3.3.3, feature vectors are required. In contrast to the RF classifier, such feature vectors are not explicitly specified in case of the SCN, although its predictions root from (automatically derived) features as well. To constitute explicit feature vectors for all our points, we propagate them through a trained network and concatenate filter responses (i.e., features) of different *U-Net* levels. In reference to the multi-scale feature computation for the RF classifier (cf. Section 3.7.1), the result can also be thought of as a multi-scale descriptor with



(a) Encoding branch level 1

(b) Decoding branch level 1

(c) Encoding branch level 2

(d) Decoding branch level 2

(e) Encoding branch level 3

(f) Decoding branch level 3

Figure 3.16: Filter responses from selected filters at different levels of our SCN. Subfigures are arranged in an order to match the U-shape of the SCN.

features from different levels of (deep) understanding. Specifically, individual filter responses from the last convolutional layer of each encoding branch and from the deconvolutional layers in the decoding branch (representing a compact, upsampled understanding of the previous level each) are concatenated to form a $16+32+48+48+32+16 = 192 \times 1$ shaped feature vector. For the respective upmost levels in the encoding and decoding branch, retrieving features is straightforward since the resolution is the same as that of the input.

For the lower resolution levels, however, we restore the original resolution without relying on any dedicated upsampling (such as deconvolution layers or other interpolation methods) to avoid distorting the original filter response. This means that deep features are simply assigned to all cells in the original resolution that have been summarized to the respective cell at a deeper level, leading to a "voxelated" depiction (cf. Figure 3.16). We can observe that upsampled filter responses from deeper encoding levels are smoother (i.e., less checkerboard-like) than their counterparts from decoding levels (cf. Figure 3.16*(c)&(e)* vs. Figure 3.16*(d)&(f)*), although being propagated from the same lower resolution level. However, this is an unfair comparison because, in the decoding branch, we retrieve features at the deconvolutional layer, where we are basically dealing with the spatial resolution of the previous deeper, lower resolved level. Apart from that, in the decoding branch essentially we can watch the SCN trying to sharpen the spatial awareness of features from a state of highest abstraction with minimum spatial awareness at the bottom of the U, which however can only be achieved by consulting information from the encoding branch through explicit links.

Although the interpretation of such deep features is a daring endeavor, we argue that a general trend is observable. Features of the encoding branch are similar to standard features often used for conventional classifiers such as RF. For instance, an exemplary filter response of level 1 of the encoding branch resembles the angle of inclination of a fitted plane (cf. Figure 3.16*(a)*), and level 2 is similar to a measure of flatness (cf. Figure 3.16*(c)*). However, since convolutions are always performed over all input channels, trails of other features such as RGB color information or *Reflectance* can be observed as well. In level 3 of the encoder branch, a feature similar to *Height Above Ground* is obtained (cf. Figure 3.16*(e)*), which can be considered a higher level feature compared to solely measuring inclination or flatness, since a spatial context awareness is already required.

Deeper levels of the decoding branch are hard to interpret - at least for a human operator. However, a trend can be observed from pure description of data in the encoding branch to a higher level understanding. For instance, level 3 of the decoding branch (cf. Figure 3.16*(f)*) highlights buildings, but also lower parts of high vegetation, i.e., other predominantly vertically oriented noisy surfaces (façades are also noisy due to balconies, etc.). Going one level up, buildings are still emphasized, but fewer parts of vegetation (cf. Figure 3.16*(d)*). Or in other words, the model tries to become a filter for individual objects respectively classes. Finally, the capability of kernels to operate as class filters is achieved in level 1 of the decoding branch, allowing separation of different classes, for instance, cars are clearly highlighted in Figure 3.16*(b)*.

3.8 Automatization via the CATEGORISE (Crowd-Based Active Learning for Point Semantics) Framework

After presenting the key components of our scheme for 3D point cloud classification, namely the crowd and the machine, connected by AL, this section gives an idea of how to cast the individual components into one fully automated hybrid intelligence system as we have claimed to design. Furthermore, some more technical details on the implementation of this system are covered in the following, which are, however, crucial for full automation. A key part of this anticipated full automation is completely avoiding the need for an expert to engage in labeling tasks, which are to be outsourced to the crowd. However, conducting crowdsourcing campaigns often causes additional overhead, such as actively recruiting crowdworkers, instructing and mon-

itoring them, and finally paying them. Such tasks can quickly become just as time-consuming as the actual labeling task, thus requiring special focus when designing the hybrid intelligence system.

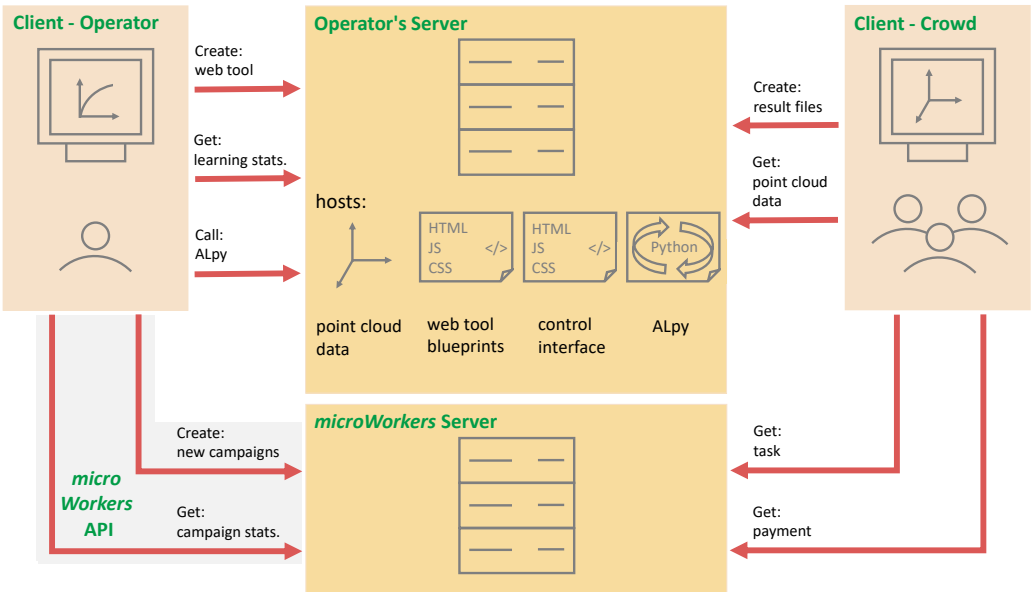
To power such a system, we first focus on its architecture and the required technologies, as displayed in Figure 3.17*(a)*. Generally, it encompasses two different actors or clients (the crowd and an operator) and two different servers. For one, this is the operator's server. It hosts all required input data (i.e., point clouds and optionally meshes) as well as our *Python*-based implementation of both AL routines and respective classifiers (referred to as *ALpy* in Figure 3.17*(a)*). Whenever the operator decides to launch a new AL iteration, a respective web tool is tailored to the operator's needs (*Type A* to start, followed by *Type B* and *Type C* in the course of the iteration; cf. Section 3.6.2). Such are, for instance, the desired class catalog, the point size for visualization in the web tool, the data to be labeled as provided by the operator at the beginning of the iteration (for a *Type A* campaign) but later automatically determined by *ALpy*, quality control settings (cf. Section 3.6.3), etc.

All our web tools are implemented in *JavaScript* and built upon the *Three.js* library (Cabello, 2019), with *HTML* and *CSS* being used for creating and styling the tool's basic layout. Setting up these tools is achieved by means of the control interface (cf. Figure 3.17*(a)*), which is also implemented in *JavaScript* and hosted on the operator's server too. After proper preparation of the labeling tool for the specific crowd campaign at hand, the creation of a campaign on a paid crowdsourcing platform is triggered through the operator's interface. As a platform, we rely on *MW*, which offers a dedicated *API* to allow for automation of crowd campaigns. Thus, utilizing the *MW API*, which can be prompted through *JavaScript* generated *HTTP* requests, we can automatically create a crowd campaign without the need to tediously going through a respective form on the *MW* homepage to set up the campaign. Instead, such crowd related parameters (e.g., *MW* worker group, payment, etc.) can be set together with all remaining system parameters, for instance concerning the AL loop or the classifier, in our dedicated control interface.

MW, which can be understood as a mediator between the employer and the crowd, will post the campaign hosted on its server as an open call to all registered crowdworkers (provided no filters were set by the operator such as "only workers from the U.S.") after manually checking the validity of the campaign (for long-term users this check is omitted) and after receipt of payment from the employer (payment for crowdworkers and *MW* fee). Crowdworkers see a brief description of the task (such as the one in Figure 3.17*(b)*; *bottom left*) with some additional information such as payment and time required. When a worker chooses to accept the task, one of the open jobs is reserved for him and he is linked to the operator's server, where he is provided with the corresponding labeling tool loaded with the 3D data of this specific job. After completing the task, the result will be stored on the operator's server (via *php*) and the crowdworker will be informed about his performance on the check points (only applicable to *Type B* and *Type C* campaigns; cf. Section 3.6.3). If the minimum requirement is met, and thus implicitly a first step of quality control is enforced (cf. Section 3.6.3), the crowdworker receives a code as proof of completion of the task (for *Type A* campaigns the code of approval is directly awarded). This code can be submitted to *MW* to request the payment. Therefore, no interaction between the operator and the crowd is required at all.

The operator, on the other hand, can monitor the progress of the current campaign and the whole AL loop (cf. Figure 3.17*(b)*), and may query some statistics such as job duration and quality (all requests are handled via *AJAX*). After completion of each campaign, integration routines are launched as the second step of automated quality control (only applicable to *Type B* and *Type C* campaigns; cf. Section 3.6.3), and the obtained labeled data is fed into *ALpy* to continue the iteration. If the stopping criterion, which can be adjusted at any time based on the current training progress (cf. Section 3.5), is not met and the operator has not terminated the iteration manually, this process can be repeated for the next iteration step(s).

As can be seen, no real involvement of the operator is actually required, although we give him complete control over the iteration. Thus, the operator can choose to just launch the iteration (based on a set of system parameters) and wait for its completion while optionally monitoring the progress. Hence, from the operator's



(a)

(b)

Figure 3.17: System architecture of the CATEGORISE framework *(a)* and its operator interface *(b)*.

point of view, this system runs just like any other computer program, although it combines *electronic* and *human computation units* in a hybrid manner, thus being a representative of a hybrid intelligence system.

Chapter 4

Data & Evaluation Metrics

For testing our methodology developed in the previous chapter, we rely on a variety of different data sets (cf. Section 4.2), that is, ALS point clouds captured from various platforms (thus incorporating specific characteristics), but also 3D textured meshes as alternative modality. Before presenting our data sets in detail (Section 4.2), we briefly describe ALS (Shan and Toth, 2008; Vosselman and Maas, 2010), the main acquisition technique used for all our data sets (cf. Section 4.1). Furthermore, we discuss appropriate and strict evaluation metrics to measure the performance of our approach (cf. Section 4.3).

4.1 Airborne Laser Scanning

ALS has become a standard method for mapping the earth's surface and is used worldwide by NMAs to produce nationwide surface data, but is equally suited for fine-grained mapping of a limited area of interest (Haala et al., 2022). It is a polar measurement principle where a stimulated light pulse, i.e., Laser (Light Amplification by Stimulated Emission of Radiation) pulse is emitted, normally at a wavelength of the near-infrared spectrum of typically 1064 nm or 1550 nm (Vosselman and Maas, 2010, p. 21, 25). This pulse traverses space until it hits an object. Due to beam divergence of the emitted light, the system illuminates a cone (with its top centered in the aperture) rather than an infinitesimal line (Shan and Toth, 2008, p. 23). This very property explains the multi-target capability (Vosselman and Maas, 2010, p. 28) to "penetrate" semi-transparent structures such as vegetation. If the illuminated area at the target, i.e., the footprint, is larger than individual structures (e.g., leaves), multiple objects are hit, each resulting in an echo. Please note that the detectability is limited by the resolution of range measurement, determined by the pulse length.

Based on the target's extension and material properties, summarized by the backscatter cross-section, a certain amount of the signal (which can be non-existent due to the directionality of reflection at specular surfaces) is reflected towards the position of the aperture of the device (Wagner et al., 2006). The system resolves the incoming signal and detects significant echos either by hardware components or by signal processing techniques based on the recorded waveform (on the fly or in post-processing) (Wagner et al., 2006; Shan and Toth, 2008, p. 245). Through measuring the time of flight of each pulse (or parts of it) for allowing precise range measurements (Shan and Toth, 2008, p. 3) and through recording the strength of the incoming signal as a material-specific property (i.e., *Intensity*), the illuminated surface patch can be attributed and positioned in 3D space, resulting in a 3D point as product of signal integration over this patch. However, this requires that the position and orientation of the scanner are determined by a GNSS/IMU unit (Shan and Toth, 2008, p. 96).

To map not only individual points, in cross-track direction the laser beam is deflected by moving mirror elements (Vosselman and Maas, 2010, p. 16; Shan and Toth, 2008, p. 102) to scan a line on the ground, which,

together with the forward motion of the platform, leads eventually to strip-wise scanning of surfaces. The resolution or point density in object space, i.e., the distance increment on the ground, is determined by the angular resolution within one scan line, which is defined by the angular momentum of the mirror together with the pulse repetition rate and the height above ground. Distance of points in along-track direction depends on the mirror's angular momentum (or more precisely, the number of scan lines accomplished per time interval) in conjunction with the speed of flight (Vosselman and Maas, 2010, p. 26). When conducting ALS campaigns, areas of interest are typically mapped with overlap between individual strips to guarantee completeness (Vosselman and Maas, 2010, p. 30). In the subsequent strip adjustment process, this also allows to identify identical structures in multiple strips to minimize discrepancies between different strips due to systematic errors present in the raw trajectory, in scanner measurements or in the mounting calibration to finally yield an adjusted flight block with improved relative accuracy (Shan and Toth, 2008, p. 259). As for GNSS/(IMU) supported photogrammetric bundle block adjustment of imagery, ground control information is not mandatory, but can help to improve absolute accuracy.

Contemporary flight campaigns and thus adjustment processes integrate classic photogrammetric bundle block adjustment into the strip adjustment process to further stabilize the measured trajectory (Glira et al., 2019; Haala et al., 2020, 2022). In such a hybrid strip adjustment, the estimated exterior orientation parameters of the images can serve as discrete supports for the more continuous GNSS/IMU trajectory observations. This yields a precise co-registration of imagery and LiDAR data, and thus allows for a more accurate colorization of individual LiDAR points by the concurrently captured imagery. This property is also beneficial for the hybrid creation of 3D textured meshes (from both LiDAR data and imagery) as another representation type of 3D data, which might be especially suited for presentation to non-geospatial experts (e.g., *Google Maps* or *Google Earth* users).

4.2 Data Sets

4.2.1 ISPRS Vaihingen 3D Semantic Labeling Contest (V3D)

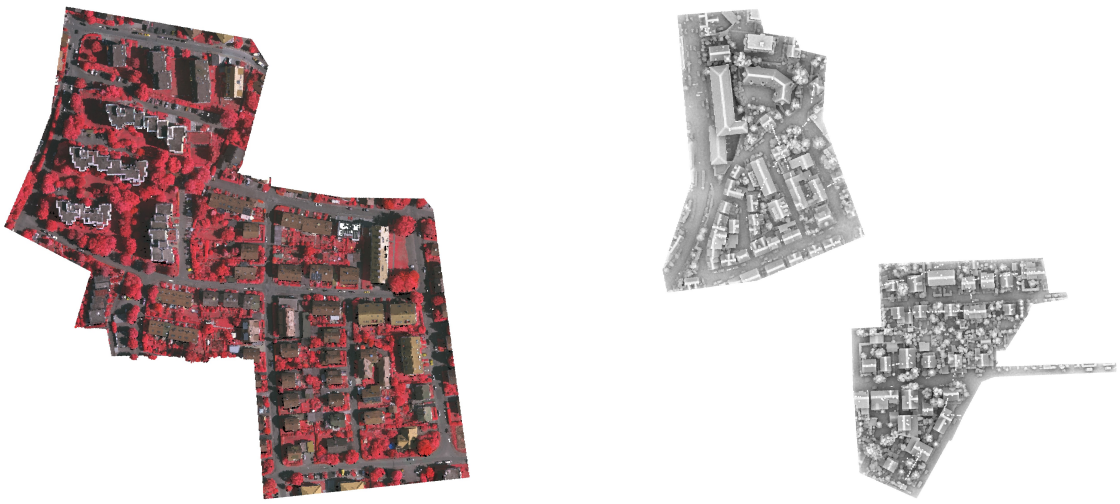


Figure 4.1: Overview of the V3D data set. While the training set (*left*) is depicted according to available TOP CIR information, the spatially disjoint test point cloud is visualized in a shaded manner (*right*).

Since the establishment of the ISPRS Vaihingen 3D Semantic Labeling Contest (V3D) data set (Niemeyer et al., 2014) for benchmarking in 2014, it has become the quasi-standard for demonstrating semantic seg-

mentation approaches for 3D point clouds in the geospatial domain. The point cloud was recorded in August 2008 by means of a *Leica ALS 50* system flown at 500 m above ground with a field of view of ±45 °, originally intended to be a geometric baseline in context of another benchmark endeavor for assessing quality of metric camera systems (Cramer, 2010; Haala et al., 2010). It depicts a typical suburban scene in southern Germany (Vaihingen an der Enz), mainly populated by single-family buildings and apartment blocks on flat terrain. Flight parameters were chosen to achieve a point density of at least 5 pts/m^2, resulting in an actual point density of about 8 pts/m^2 due to strip overlap (see also Table 4.1).

The data set was manually semantically segmented by Niemeyer et al. (2014) into a total of 9 distinctive classes (cf. Table 4.1 and Figure 4.4*(g)*), aiming to depict the highest level of detail with respect to the inherent resolution. Apart from typical ALS features (*Intensity* & (poor) echo readings; cf. Figure 4.4*(a)&(d)*), radiometric information can be added by relying on False Color Infrared (CIR) True Orthophoto (TOP) data provided for the International Society for Photogrammetry and Remote Sensing (ISPRS) 2D Semantic Labeling Contest (Gerke et al., 2014; Rottensteiner et al., 2014). That imagery was acquired by an *Intergraph/ZI DMC* camera configured to achieve a Ground Sampling Distance (GSD) of 9 cm and was mapped to the point cloud data via orthogonal projection (cf. Figure 4.1).

4.2.2 ISPRS Hessigheim 3D Benchmark on Semantic Segmentation of High-Resolution Point Clouds and Meshes (H3D)

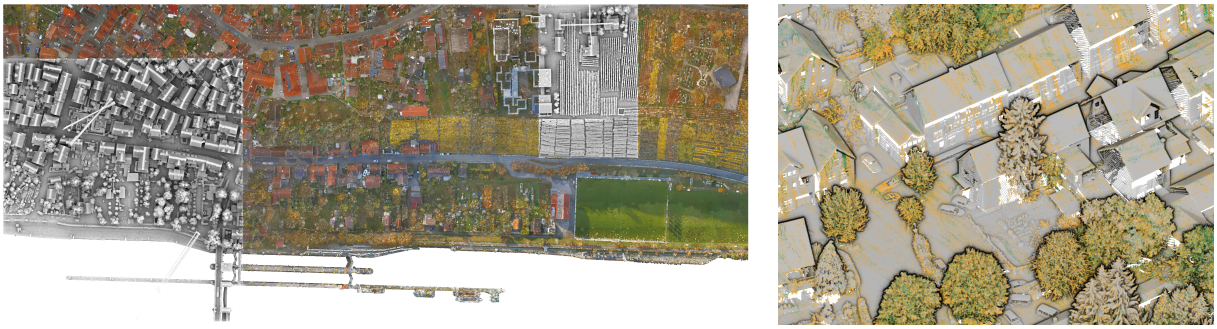
The ISPRS Hessigheim 3D Benchmark on Semantic Segmentation of High-Resolution Point Clouds and Meshes (H3D) data set (Kölle et al., 2021a) was introduced only recently to enrich the spectrum of 3D benchmarks by a multi-temporal and multi-modal data set. In total, 4 different epochs are available, that is, 3 ALS point clouds captured from an Unmaned Aerial Vehicle (UAV) platform (in context of a project aiming at highly precise deformation measurement; Haala et al., 2022) and one point cloud provided by the State Office for Spatial Information and Land Development Baden-Wuerttemberg (LGL) with typical properties of NMA point clouds (see also Section 4.2.3). The spatially congruent point clouds (except for the less wide epoch March 2018, cf. Figure 4.2) represent an area of the village of Hessigheim situated in southern Germany, comprising suburban scenes with single-family and multi-family buildings and leisure facilities (such as sports fields), but also agricultural land (e.g., small crop fields and steep vineyards).

As for the UAV campaigns, we limit ourselves to usage of the first two epochs (March 2018 & November 2018) in this work (cf. Figure 4.2*(a)&(b)*). For both, a *Riegl Ricopter* platform equipped with a *VUX-1LR* laser scanner and two obliquely mounted *Sony Alpha 6000* cameras were employed. The system was flown at a height above ground of 50 m with a field of view of the scanner of ±70 °. Thus, vertical surfaces such as façades are well depicted (cf. Figure 4.2*(a)&(b)*). Both point clouds map the scene at an extraordinary point density of more than 800 pts/m^2 and provide range-corrected intensity values (i.e., *Reflectance*) as well as multi-echo readings with apart from first echos, mainly second and third echos (cf. Table 4.1 and Figure 4.4*(b)&(e)*). Colorization of the point clouds is accomplished by nearest neighbor interpolation of RGB tuples from the 3D textured mesh (as also provided in the H3D benchmark) and approximates occlusion-aware projection of LiDAR points to images (Kölle et al., 2021a). Thus, compared to the V3D and Stuttgart 3D Point Cloud (S3D) data sets, the colorization is more true due to a real 3D colorization by imagery captured at the same time with the exact same scene geometry.

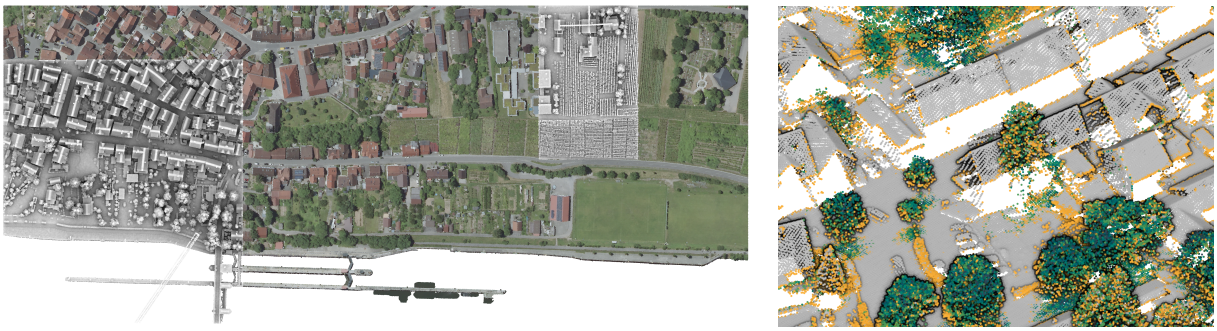
These two UAV epochs differ in terms of i) the spatial extent (epoch November 2018 depicts an area more than two times larger than epoch March 2018), ii) phenological changes and iii) the atmospheric conditions, which mainly impacts radiometric properties. Apart from different illumination, thick fog was present in November 2018, severely affecting RGB colorization from the concurrently captured images. This can be clearly seen in Figure 3.9, where *source domain* is a subset of epoch March 2018 and *target domain* of epoch November 2018. The presence of fog also resulted in recording multiple echos for streets, façades and roofs due to additional returns caused by aerosols (compare right side of Figure 4.2*(a)&(b)*).

The third epoch (March 2016) incorporates features of a typical ALS point cloud of NMAs. The data set was captured by a *Riegl LMS-Q780* scanner at a height above ground of 600 m and a field of view of ±20 °. The mean point density is about 16 pts/m^2. This configuration leads to scarce depiction of vertical surfaces due to the nadir-like measuring perspective (see few façades in Figure 4.2*(c)*; *right*). Multi-echo readings profit from the larger footprint area due to the higher flying altitude and thus lead to observing echo numbers > 3 in contrast to the other epochs (cf. Figure 4.2*(c)*; *right*). Since no concurrently captured imagery is available for this epoch, we both i) interpolated colors from epoch November 2018 via nearest neighbor transfer and ii) used the point cloud colorization obtained by orthogonal projection of an orthophoto

(a) H3D - Epoch March 2018



(b) H3D - Epoch November 2018



(c) H3D - Epoch March 2016

Figure 4.2: Compilation of H3D's data sets/epochs (except for epoch March 2019). Each visualization depicts an overview of the complete data set, where the training set is colored according to available RGB data while the test set is presented as shading (*left*). Respective close-ups are colorized according to the number of returns of an emitted laser pulse (*right*, cf. Figure 4.4 for color scheme).

as available in the H3D benchmark (visualized in Figure 4.2*(c)*; *left*). The orthophoto was acquired by a *DMC II 140* camera configured to achieve a GSD of 20 cm and was taken on June 11, 2017.

The semantic class catalog (cf. Table 4.1 and Figure 4.4*(h)*) is developed to mimic V3D's labels while introducing classes for objects not depicted there due to the coarser resolution (e.g., classes *Urban Furniture* and *Chimney*), but also accounting for H3D's special characteristics such as agricultural land (class *Soil/Gravel*) and a ship lock (class *Vertical Surface*). Semantic labels of all epochs were manually set by a team of student assistants employed in a multi-stage process of labeling from scratch and quality checks. Since one of the goals of this work is to minimize labeling effort by experts, we aim to give an estimate of the labeling costs, exemplarily for H3D's epoch November 2018. Annotating the complete point cloud from scratch and checking each point two times (checks were conducted by different student assistants each) took about 1490 hours. That is for an area of about 0.207 km^2 resulting in an average time effort of 0.431 min/m^2 and makes with a salary of \$14.69 /h an amount of \$0.106 $/m^2$, respectively. Please note that, on one hand, labeling costs highly depend on the complexity of the scene, the defined class catalog and the desired accuracy level, defining the number of multiple acquisitions or checks. On the other hand, they vary with the skill and salary of the workers. Nevertheless, this calculation is supposed to give an impression of effort required to obtain data for real-world projects. For comparison, Zolanvari et al. (2019) report a similar workload of 0.194 min/m^2 for labeling the DublinCity data set (13 classes).

In addition to labeled point clouds, H3D also provides labeled and textured 3D meshes. A favorable property of H3D's meshes, in contrast to other data sets (Gao et al., 2021), is that the jointly adjusted imagery and the concurrently captured LiDAR point clouds are utilized to generate a 3D surface model as completely and realistically as possible by means of the two sensors (such processing is supported by combination of *OPALS* (Pfeifer et al., 2014) and *SURE* software (Rothermel et al., 2012)). Precisely, geometric completeness greatly benefits from including LiDAR measurements in cases where surfaces are occluded by semi-transparent objects such as vegetation, which cannot be reconstructed when purely relying on imagery. The same applies to urban canyons, which are likely to be depicted in only one image, making image-based reconstruction impossible. Texturing this jointly built geometry with image crops from simultaneously captured and adjusted oblique imagery, finally, leads to the truest representation of scenes.

4.2.3 Stuttgart 3D Point Cloud (S3D)

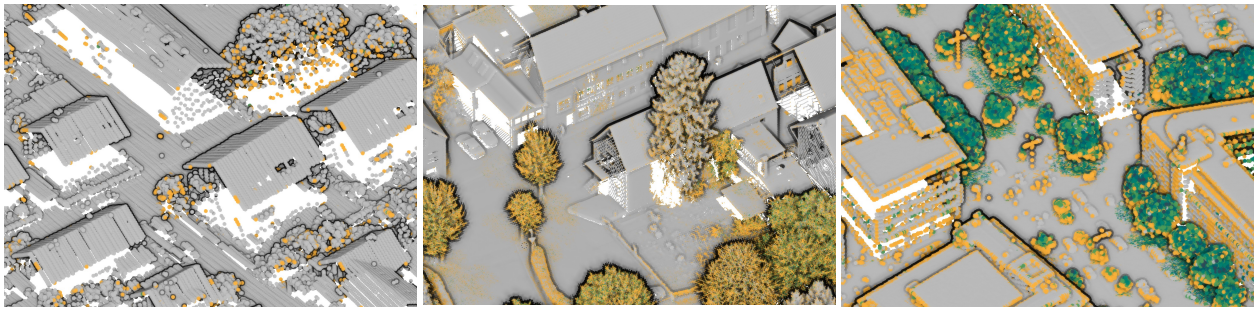
In addition to the comparably small-scale point clouds presented above (V3D & H3D), we also aim to employ a point cloud of much greater spatial extent (about 30 times as big as the V3D data; cf. Table 4.1) and thus more object variety. As for epoch March 2016 of the H3D data set, we have received NMA point cloud data from the LGL depicting the city center of Stuttgart, Germany. This Stuttgart 3D Point Cloud (S3D) includes urban commercial and administrative buildings, infrastructural facilities (such as the main station) and a variety of different types of residential buildings, but also green spaces like parks, gardens and steep vineyards. Data capturing took place in spring 2016 with a *Riegl LMS Q780* system utilizing an effective field of view of ±20 °, flown at a height of 600 m above ground (same flight parameters as H3D's epoch March 2016).

Again, typical ALS features are incorporated (cf. Table 4.1 & Figure 4.4*(c)&(f)*). Like for H3D's NMA epoch March 2016, multi-target capabilities profit from a larger illuminated footprint and the more favorable time of year, and exceed those of V3D and UAV-based H3D epochs. The point density is comparable to the one of V3D (and naturally to H3D's epoch March 2016). The generation of GT labels was completed by a private contractor commissioned by the LGL, where the classification error is supposed to be $< 1\,\%$. After manually adding of class *Tree* by Schmohl et al. (2022), with a total of 4 classes (cf. Table 4.1 & Figure 4.4*(i)*), S3D's class catalog is still rather coarse compared to our other data sources, but well suited for tasks such as DTM generation (please note that class *Urban Furniture* serves as quasi-class *Other*). Again, we colorized the point cloud by orthogonal projection of an RGB TOP with a GSD of 20 cm, based on *DMC II 140* imagery captured in summer 2017.

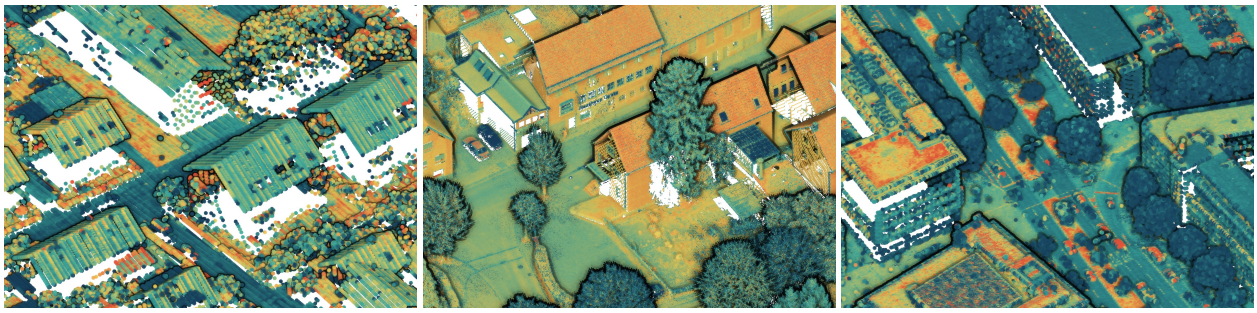
Figure 4.3: Overview of the S3D data set. While the training set is colored according to available RGB information, the test split is visualized in a shaded manner.

Data Set	Split	# Points [M]	Area [km^2]	Density [pts/m^2]	Attributes		Classes
					LiDAR	Color	
V3D	train test	0.754 0.412	0.084 0.046	8	Intensity, Echo Number, # Echos	CIR	*Powerline*, *Low Vegetation*, *Impervious Surface*, *Car*, *Fence/Hedge*, *Roof*, *Façade*, *Shrub*, *Tree*
H3D (Mar. 2018)	train test	73.909 51.745	0.062 0.031	> 800	Reflectance, Echo Number, # Echos	RGB	*Low Vegetation*, *Impervious Surface*, *Vehicle*, *Urban Furniture*, *Roof*, *Façade*, *Shrub*, *Tree*, *Soil/Gravel*, *Vertical Surface*, *Chimney*
H3D (Nov. 2018)	train test	159.031 83.283	0.140 0.067	> 800	”	”	”
H3D (Mar. 2016)	train test	2.809 1.428	0.140 0.067	16	Intensity, Echo Number, # Echos	”	”
S3D	train test	38.511 40.417	1.819 2.179	16	Intensity, Echo Number, # Echos	RGB	*Urban Furniture*, *Ground*, *Building*, *Tree*

Table 4.1: Properties of relevant point cloud data sets utilized within this work.



(a) V3D - Number of Returns *(b)* H3D - Number of Returns *(c)* S3D - Number of Returns



(d) V3D - *Intensity* *(e)* H3D - *Reflectance* *(f)* S3D - *Intensity*

(g) V3D - Semantic Labels *(h)* H3D - Semantic Labels *(i)* S3D - Semantic Labels

Figure 4.4: Comparison of point clouds from our 3 test sites (*from left to right*: V3D, H3D - epoch March 2018 and S3D).

4.3 Evaluation Metrics

While the presented data sets represent appropriate and diverse playgrounds to test our approach, for a strict and quantitative evaluation, respective quality metrics need to be defined. Generally, we aim to assess the quality of class predictions that stem either from crowdworkers or from an ML algorithm, thus allowing to use the same metrics. These are based on a comparison of the assigned class label of each of the n_p points to expert-level GT labels that we assume to be error-free. But although benchmark data sets such as V3D or H3D exhibit a high quality of labeling overall, label noise is undoubtedly included. To measure the quality of our results, we rely on the Overall Accuracy (OA), obtained as the ratio of correctly predicted instances to the total number of predictions.

For a finer, class-aware analysis of results, confusion matrices are calculated, where each column comprises the sum of all GT labels of one of the classes that get distributed more or less correctly among the different classes (represented in respective rows) during the prediction process. To present the results more clearly, we normalize our confusion matrices by the number of GT labels per class (i.e., sum of columns). From the number of true positives (TP), false positives (FP) and false negatives (FN) for each class c recorded in the confusion matrix, class-dependent accuracies such as Recall R (Producer's Accuracy) and Precision P (User's Accuracy) can be derived. While Recall indicates the number of points of a specific class that were correctly identified as belonging to that class in relation to the total number of GT points of that class, Precision checks the number of points that were correctly predicted to belong to a specific class relative to the total number of points assigned to that class. These two measures can be combined into one overall quality metric for each class, namely the F1-score, and, to receive an overall class-sensitive measure, we compute the arithmetic mean of F1-scores over all n_Ω classes, referred to as mF1. Formally, this leads to (Goutte and Gaussier, 2005):

$$\begin{gathered} \mathrm{OA} = \sum_{c=1}^{n_\Omega} \frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FP}_c} \\ \mathrm{P}_c = \frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FP}_c} \qquad \mathrm{R}_c = \frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FN}_c} \qquad \mathrm{F1}_c = 2 \cdot \frac{\mathrm{P}_c \cdot \mathrm{R}_c}{\mathrm{P}_c + \mathrm{R}_c} \qquad \mathrm{mF1} = \frac{1}{n_\Omega} \sum_{c=1}^{n_\Omega} \mathrm{F1}_c \end{gathered} \tag{4.1}$$

Additionally to those accuracy-related measures, we identify the number of labeled instances as a key factor for our system, since our main goal is to minimize the total labeling effort while maintaining an accuracy similar to that of a full annotation, i.e., as in PL. In other words, we would accept our system falling below top-notch accuracy if a slightly worse performance could be reached with significantly less labeling effort, i.e., if our system was more budget-friendly. Consequently, we anticipate that we will be confronted with a trade-off between model accuracy and labeling effort: Generally speaking, the more (informative) labels for training, the more accurate the result of the ML model, but on the other hand, the more labeling jobs and thus costs for payment of crowdworkers arise.

Chapter 5

Experiments & Results

Following the outline of our methodology (Chapter 3) and the presentation of the data sets of interest (Chapter 4), we can finally design a variety of experiments. The purpose is to come up with a configuration capable of semantic segmentation of 3D point clouds, with special emphasis on an *efficient* solution. We will approach this task in two stages, in accordance to our overall design blueprint of bringing together *electronic processing units* and *human processing units*, in our case the crowd. At first, we will focus on finding an optimal configuration and parametrization of the machine part, but we will also consider real-world conditions, that is, learning from noisy oracles (cf. Section 5.1). Afterwards, we conduct various crowdsourcing campaigns in order to test the general suitability of employing crowdworkers for 3D data annotation and how to improve the quality of crowdsourced labels (cf. Section 5.2). Finally, we merge both findings to power a hybrid intelligence system in the form of a fully automated AL loop for 3D data interpretation for various data sets (cf. Section 5.3).

5.1 Evaluating and Optimizing the Machine Component

Setting up the machine part requires a variety of choices, including finding the right AL sampling strategy, choosing a proper classifier, and also appropriate parametrization. Thus, in the following sections, we pursue to gradually reduce uncertainties in the set-up by identifying an optimal and robust set of methods and parameters, while freezing the other components that are currently not under investigation. Following this idea, we start with an in-depth evaluation of the key component of the AL loop, that is the selection of most informative points (cf. Section 5.1.1), followed by a comparison of the performance of classifiers from both the feature-driven and data-driven domain (cf. Section 5.1.2). Afterwards, we evaluate the impact of realistic imperfect oracles on the AL system and attempt measures to facilitate the labeling process (cf. Section 5.1.3). Lastly, we test our findings on all our proposed data sets to outline the accuracies theoretically possible to achieve with our proposed system (cf. Section 5.1.4).

5.1.1 Comparing Sampling Strategies

In order to find the best sampling strategy and its proper parametrization, we start by applying the methods outlined in Section 3.3 to the V3D data set. In each run, the model is allowed to draw samples from the official V3D training set and performance is evaluated by means of applying the resulting model to the dedicated test set. Since the choice of the ideal classifier is postponed to the following section, we solely rely on an RF classifier that is parametrized by an ensemble of $n_e = 100$ Decision Trees with a maximum depth of $d^{max} = 18$ and a minimum number of samples at a node to justify a new split of $n^{min}_{samples} = 7$. This configuration has proven to be robust and efficient for semantic segmentation tasks (Niemeyer, 2017).

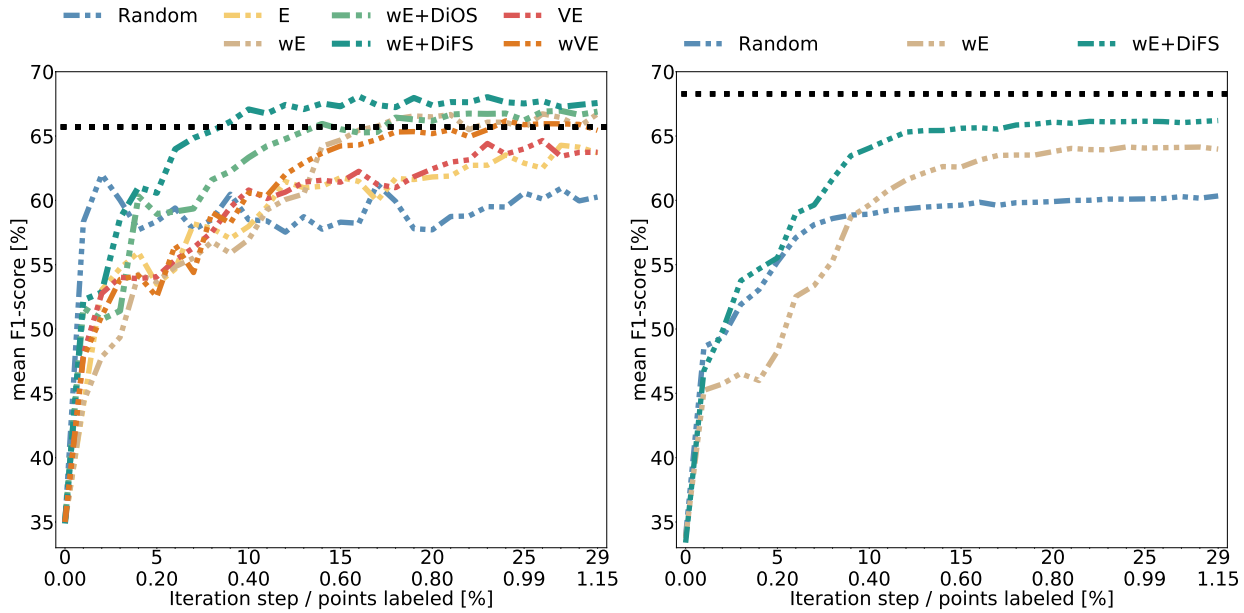
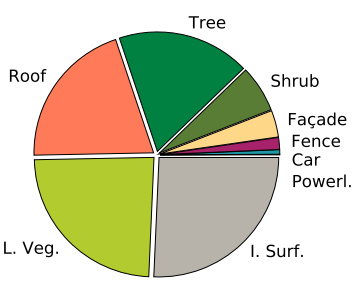


Figure 5.1: Comparison of the performance of different AL sampling strategies in combination with both an RF classifier *(a)* and our SCN classifier *(b)* when applying the resulting model to the V3D test data set. *Dotted black lines* represent the result of PL.

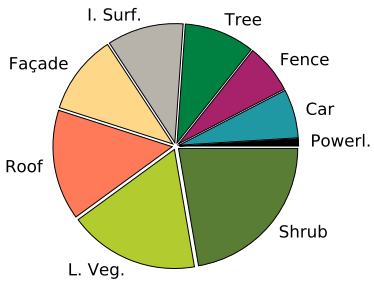
For feature computation, we are guided by the suggestions of Niemeyer et al. (2014), Schmidt et al. (2014), Blomley et al. (2016) and define the spherical neighborhood radius $r \in \{1, 2, 3, 5\}$ m. Respective features are derived with the *CloudCompare* software (Girardeau-Montaut, 2021) in command line mode. Furthermore, we assume to deal with an omniscient oracle $\mathcal{O}_O$ and run our iterations for a fixed number of $n_i = 30$ iteration steps, where $n^+ = 300$ points are added in each step (the effect of n^+ will be discussed at the end of this section). The required initial training set is built in such a manner that it includes 10 typical representatives for each class. This means that we select points that are far away from class borders and that are likely similar to the points that real crowdworkers would select using our labeling tool *Type A* (cf. Section 3.6.2).

The resulting AL learning curves by means of different sampling schemes are depicted in Figure 5.1*(a)* (an evaluation of additional sampling strategies can be found in the Appendix in Figure B1). As baseline solution, we also report the result of PL, where the model roots from the completely labeled training set. Another baseline, which however focuses on gradually improving the classifier, is *random sampling*. This sampling scheme achieves a surprisingly high mF1, especially related to the number of labels required, fluctuating around 60 % after only very few iteration steps. This is a result of selecting a mixture of both most informative and less informative points by chance. With such a training set, we are able to achieve a reasonable level of accuracy, which seems to be bounded, however, since most informative points are underrepresented. This is due to *random sampling* generating a training set following approximately the same distribution of points as the underlying data set where the informative points close to class borders are comparatively scarce.

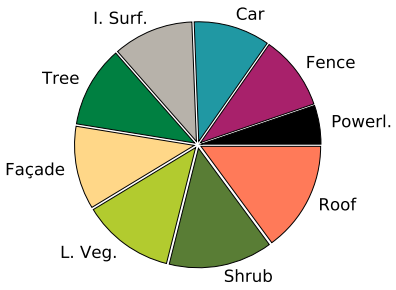
This motivates applying a smarter sampling strategy. To this end, we compare *entropy* and *vote entropy* sampling (cf. Section 3.3.1). As aspired, such methods improve the long-term accuracy by margin, but in early iteration steps, they are outperformed even by *random sampling*. This is due to the fact that *random sampling* is capable of selecting most informative points by chance from the beginning. On the other hand, more directed sampling strategies choose instances close to the decision borders (*entropy*) or in ambiguous



(a) V3D's training set; $E = 2.481$



(b) Entropy sampling; $E = 2.931$



(c) Weighted entropy sampling; $E = 3.126$

Figure 5.2: Comparison between the class distributions in the original training data set *(a)* vs. those obtained by respective AL runs (*(b)*&*(c)*) each after 30 iteration steps. Furthermore, we report *entropy* values of those distributions.

regions (*vote entropy*) based on the current suboptimal separation hypotheses, which is founded on a highly sparse training set. This can be interpreted in such a way that the directed sampling strategies require several iteration steps to approach the actual decision border. Upon reaching it, however, *random sampling* can no longer compete. In terms of *entropy* vs. *vote entropy* sampling, accuracies differ only marginally, so they can be assumed equally efficient.

Further improvement of the accuracy curves can be achieved by applying the weighting scheme to both sampling strategies, i.e., wE and wVE (cf. Section 3.3.2), again yielding close to similar accuracies. This allows reaching a result even outperforming the PL baseline solution. The explanation for why this weighting scheme is so effective emerges from considering the class distribution of the obtained training sets. Figure 5.2 gives a visual impression of the class distribution in the original V3D training set as well as in the ones obtained by means of our AL loops. As we can observe, the inherent class imbalance of ALS point clouds can be significantly alleviated by basic AL sampling strategies, such as *entropy* sampling, but is practically eliminated by the weighted variants, such as *weighted entropy* sampling. This is also expressed by means of the *entropy* values of the class distributions in Figure 5.2, where large *entropy* values correspond to a high level of class balance - the maximum value for a true equal distribution would be $-\log_2(1/9) = 3.170$ (valid for a 9 class scenario such as in V3D). Since classification models can be significantly impacted by class imbalance of training data (Elkan, 2001; Criminisi and Shotton, 2013; Weinmann et al., 2015c), such an almost class-balanced training set, as shown for the final one obtained after 30 iteration steps generated via wE sampling (cf. Figure 5.2*(c)*), can be beneficial for training respective models. Thus, theoretically, for training such models, no strategy is necessarily required to increase the impact of underrepresented classes to obtain an unbiased classification result.

While strategies to reach a level of accuracy close to or even better than PL could be verified, we still need a relatively large set of labels since top accuracies are only achieved in late iteration steps. For instance, the weighted variants result in close to linearly ascending accuracy curves. From an economic point of view, a much steeper increase in early iteration steps would be beneficial. However, in our batch-mode AL scenario, in each iteration step we add $n^+ = 300$ samples to our training data set, which are up to now selected purely according to their score values (E or VE). But to guarantee diversity in the sampled batches and hopefully, boost the convergence of our AL runs, we presented the *DiOS* and *DiFS* sampling add-ons (cf. Section 3.3.3). Both are applied to *weighted entropy* sampling as this strategy turned out to produce top results as discussed previously (although performing quite similarly to *vote entropy* sampling).

Figure 5.1*(a)* demonstrates that both sampling add-ons are well suited for boosting the convergence, but *DiFS* outperforms *DiOS* by margin and is thus to be preferred in conjunction with *weighted entropy* sampling, converging already after about 10 iteration steps. The success of *DiFS* over *DiOS* is due to *DiOS*

basically brute-forcing diversity in object space by means of the d_{DiOS} hyperparameter, i.e., the minimal distance between samples in one batch. It is designed as a global parameter, but in many cases, a more data-oriented measure might be required, i.e., in some cases where points are close in object space, but indeed informative each and represent no quasi-duplicates. In such cases, *DiOS* eventually prohibits sampling all of those points - although this restriction is deliberately relaxed by enforcing d_{DiOS} only within one iteration step. Practically, this means if, for instance, 30 most informative points are closer than d_{DiOS}, it would also take 30 iteration steps to include them all in the training data set. *DiFS*, on the other hand, is capable of dynamically adapting to the data. If most informative points are really different with respect to their representation in feature space, i.e., if they belong to 30 different feature space clusters, *DiFS* is able to sample all at once while guaranteeing diversity. However, for a fair comparison of the two methods, we should also consider different values for the *DiOS* hyperparameter, i.e., d_{DiOS}, which we have set to 5 m in our experiments here. Although choosing this value seems rather obscure and might be also highly data- and class-dependent, we found it to be relatively robust (cf. Appendix Figure B2). *DiFS*, on the other hand, does not incorporate a dedicated hyperparameter, also underlining the preference of this strategy.

Principally, the claim of such diversity strategies is to allow for greater batch sizes, which we have set to $n^+ = 300$ to produce the results depicted in Figure 5.1*(a)*. This value has turned out to be rather robust, i.e., no matter the number of iteration steps, a similar number of labeled samples leads to similar accuracies (cf. Appendix Figure B3). This means that a greater batch size can help to minimize the number of required iteration steps and thus training cycles, which can be beneficial whenever classifiers with a time-consuming training process are employed. However, even with a diversity-based sampling strategy, greater batch sizes pose the risk of increased labeling effort due to separation hypotheses that may already be outdated as a result of infrequent classifier updates.

5.1.2 Comparing Classifiers

Building upon the preceding experiments on the choice of the query function, we now shift the focus to the analysis of another component in the AL loop, that is the classifier for 3D point cloud segmentation. As outlined before, we aim to compare a feature-driven RF classifier to our data-driven SCN approach (cf. Section 3.7). For the experiments reported in the following, we hold on to the basic parametrization, an AL loop running for $n_i = 30$ iteration steps and sampling $n^+ = 300$ points in each step. As for the sampling strategy, we advocate for *weighted entropy* and again enhance it by *DiFS* to guarantee diversity of samples. With regard to the SCN-related parameters, we employ a voxel size of $d_v = 0.5\,\text{m}$ and allow the network to train in each iteration step until convergence determined by the early stopping criterion (where training is aborted when validation accuracy does not improve more than 0.3 percentage points (pp) over 15 epochs). Respective training data is prepared by forming subsets of $32\,\text{m} \times 32\,\text{m} \times h$ with 30 % overlap and 30 ° angular increments in data augmentation subdivided into mini-batches of 32 subsets (cf. Section 3.7.2). Furthermore, we form a *deep ensemble* of $n_e = 5$ networks. To reduce computation time, networks of each iteration step start their training cycle based on the result of the previous iteration step and use the current decayed learning rate (the initial rate is set to 0.1 and reduced by a factor of 0.7 when validation accuracy does not improve more than 0.3 pp over 4 epochs). These parameter settings are based on the work of Schmohl and Sörgel (2019) and were identified as trade-off between accuracy and efficiency. Please note that for both our SCN and RF classifier, we select reasonable default parameterizations as we are more interested in optimizing training data instead of respective classifiers, as recommended by Ng (2021).

Generally, the two classifiers yield similar accuracies (cf. Figure 5.1), with the SCN-based loops following a smoother pattern. This is due to the incremental learning scheme, i.e., the continuation of training each time based on the result of the previous iteration step with the current learning rate versus completely re-training the classifier on the current training set as for the RF classifier. Interestingly, final results of *random sampling* are almost identical, emphasizing the general idea that the impact of the classifier is of subordinate importance compared to setting up a meaningful training set. Again, for the SCN, *DiFS* greatly helps to both improve final classification accuracy as well as boost accuracy in early iteration steps, resulting in less

manual labeling and thus cost. As for the RF classifier, the final classification level is reached after about 10 iteration steps. However, both *wE* and *wE+DiFS* reach a lower accuracy compared to the RF-based runs and fail to surpass the PL learning result, which is, however, harder to beat than the one of the RF (cf. Figure 5.1).

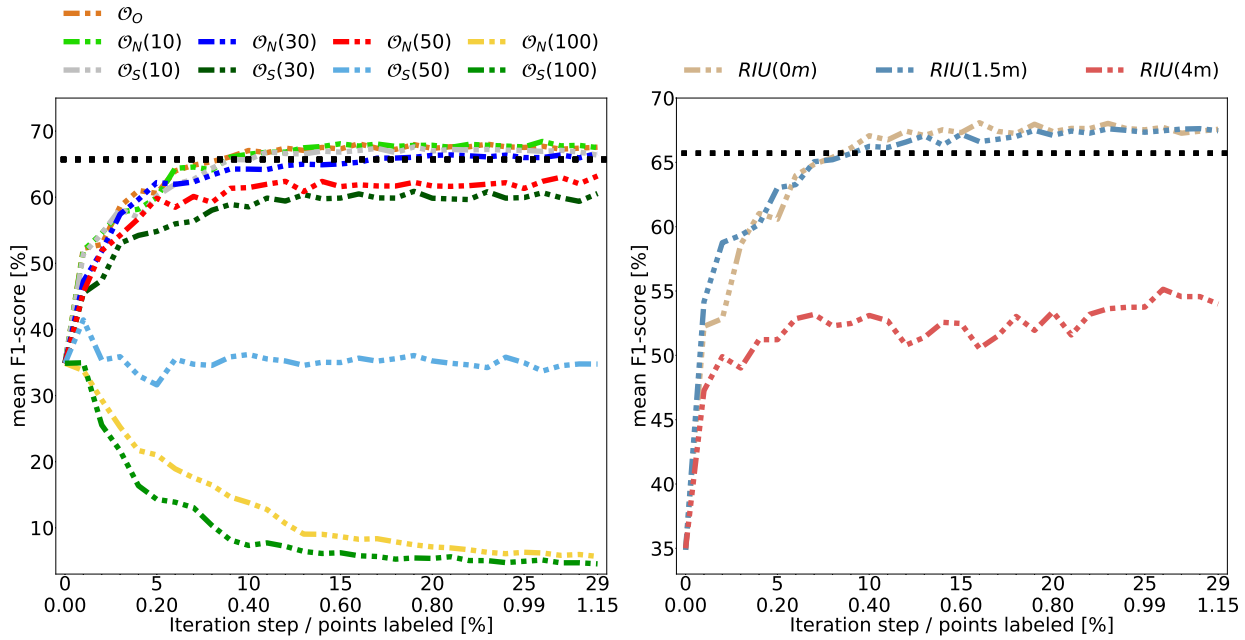
Although reachable results for both classifiers are quite similar, this comes with a significant difference in computational effort. CNN-based approaches are per design suboptimal for AL (at least from an economic point of view), since in each iteration step features need to be completely relearned or at least refined based on the currently available training set. In contrast to such a label-depending feature computation, on the other hand, for the RF-based classifier, feature computation can be performed completely independent of available label information and thus only needs to be performed once as such hand-crafted features are static throughout the whole iteration. Thus, a CNN-based approach is doomed to be more computationally intensive. A learning cycle using an RF powered loop with $n_e = 100$ Decision Trees trained in parallel on an upper midrange CPU is completed in less than a minute. An AL loop relying on a SCN classifier with $n_e = 5$ successively upper medium GPU-trained ensemble members results in a computation time that is higher by a factor of 50. These timings were measured for the V3D data set, but for data sets of greater voxel volume (either due to a greater spatial extent or due to a greater object richness), computational effort multiplies with the number of active voxel cells (cf. Section 3.7.2). In case of feature computation for the RF, the pure extent of a data set does not affect computation time as this rises with the number of instances in the respective point cloud. Nevertheless, computing features for all instances only once is still negligible in extra computation effort compared to the additional load for the SCN of re-computing features in each iteration step. Thus, this gap in computational effort becomes even greater.

Independent of these classifier-dependent computational costs, compared to the effort of manual labeling by humans, pure computation time is likely to be of subordinate importance. It only becomes critical when those humans stay idle during the training period of the classifier. But when using a crowd platform, crowdworkers are recruited just in time as *human processing units*, thus alleviating this issue.

5.1.3 Considering Imperfect Oracles

In company of the sampling strategy (cf. Section 5.1.1) and the classifier (cf. Section 5.1.2), the third component under consideration is the oracle generating the labels and thus even allowing AL to work. So far, we have assumed omniscient GT oracles $\mathcal{O}_O$, but this is an unjustified mindset when real humans are acting as oracles, even more when non-experts are working with unfamiliar data. Thus, before employing real crowdworkers later on, we start by simulating imperfect oracles, both by means of random noise (i.e., $\mathcal{O}_N$) and systematic mislabeling of classes (i.e., $\mathcal{O}_S$) referred to as confused oracles (see also Section 3.6.1). For the latter oracle, we need to define a mapping function that is built in such a way that similar classes are mixed up, but it also roots from our experience with crowdsourced data annotation, reported in Section 5.2. For V3D, the mapping is outlined in Table 5.1.

Using such oracles within an AL loop sampling points according to *wE+DiFS* and utilizing an RF classifier yields the results visualized in Figure 5.3. Naturally, the highest accuracy is obtained with the omniscient oracle $\mathcal{O}_O$. For imperfect oracles, both a noisy oracle $\mathcal{O}_N$ and a confused oracle $\mathcal{O}_S$, each with an error rate $\sigma = 10\,\%$ affects the classification accuracy only marginally. This is an important observation as it grants our crowdworkers we intend to employ an error rate of 10 %. When increasing this error rate, our RF classifier in the AL loop is remarkably robust against random noise. With 30 % random noise, the performance is only slightly impacted and even 50 % noise yields a reasonable level of accuracy still. A confused oracle $\mathcal{O}_S$, on the other hand, diminishes performance at a much higher pace than random noise. For instance, a confused oracle $\mathcal{O}_S$ with $\sigma = 30\,\%$ performs even worse than a noisy oracle $\mathcal{O}_N$ with $\sigma = 50\,\%$. Conducting the same experiments on the H3D data set (cf. Appendix Table B1 & Figure B4) verifies these findings expressing proper generalizability. Thus, systematic errors, as occurring with confused oracles $\mathcal{O}_S$, should be limited as much as possible, which is why we also apply the *RIU* technique (cf. Section 3.3.4).



(a) *(b)*

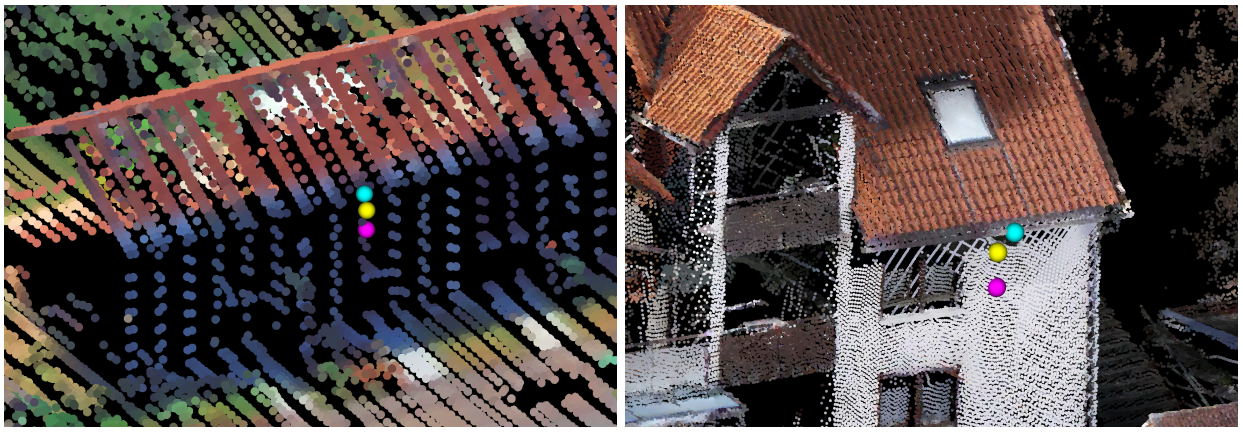
Figure 5.3: Evaluating the impact of the oracle component in AL. We compare different imperfect oracles (*(a)*; arguments in brackets represent the error rate σ [%]) as well as measures to counter labeling errors *(b)*. Both simulations are based on an RF classifier with $wE + DiFS$ sampling for the V3D data set. Respective AL loops are conducted for $n_i = 30$ iteration steps utilizing a batch size of $n^+ = 300$.

With *RIU*, we aspire to minimize labeling errors with respect to ambiguous regions by moving away from the class borders to allow for a more clear annotation task (cf. Section 3.3.4). But since this moving away also yields less informative points, this might severely harm the performance of the classifier. To assess the potential effect of this method on both the performance of our classifier and the labeling accuracy of the crowd, we simulate different AL loops, with different values for d_{RIU} (cf. Figure 5.3), and we will also employ a real crowd oracle $\mathcal{O}_C$ for labeling queried points for an exemplary iteration step (cf. Section 5.2.5). Regarding the performance of the classifier, we can observe that a maximum allowed distance to the supposed class border of $d_{RIU} = 1.5\,\text{m}$ yields almost the same accuracy values as strictly focusing on most informative points as queried by the classifier without the *RIU* add-on. Thus, we can support crowdworkers by easing the labeling process to minimize errors due to ambiguous regions without loss in classification accuracy of our ML model.

However, the impact of different d_{RIU} values is also correlated with the resolution of the data set. For instance, when dealing with a high-resolution data set such as H3D, a small value for d_{RIU} might already be sufficient to resolve label ambiguities, while for a more coarsely resolved data set, crowdworkers might not benefit from small d_{RIU} values at all (cf. Figure 5.4). Thus, a higher resolution (as in H3D) allows to reduce d_{RIU} values, but at the same time requires this reduction since structures become more detailed and even

true label	Powerl.	L. Veg.	I. Surf.	Car	Fence	Roof	Façade	Shrub	Tree
confused with	Roof	Fence	Façade	I. Surf.	Shrub	Façade	Roof	Tree	Shrub

Table 5.1: Assumed behavior of a confused oracle $\mathcal{O}_S$ regarding the V3D data set.



(a) *(b)*

Figure 5.4: Comparing different values for d_{RIU} to ease labeling for crowdworkers. The higher the resolution of the data set, the smaller d_{RIU} can be chosen to indicate unambiguous points. Thus, instead of the machine selected seed point (*cyan*), either a distance of $d_{RIU} = 1.5\,\text{m}$ (*magenta*) in case of the V3D *(a)* or of $d_{RIU} = 0.75\,\text{m}$ (*yellow*) in case of the H3D *(b)* data set could be suitable.

a small distance can cause fine structures such as urban furniture to be skipped, resulting in a diminished classification performance (cf. Appendix Figure B5). From Figure 5.3*(b)* (and Figure B5 in the Appendix), we can also observe that if d_{RIU} is chosen too large, e.g., $d_{RIU} = 4\,\text{m}$, the accuracy of the classifier suffers significantly. Allowing such a big distance to the queried seed point, a representative too far away from the region of interest is chosen, often located on completely different objects (with probably different underlying classes), if we are confronted with objects that have a small spatial extent (cf. Figure 3.7). Naturally, such points incorporate feature vectors that are too dissimilar to that of the queried seed point to be informative. Therefore, learning from such points fares even worse than *random sampling* (cf. Figure 5.1), where at least by chance some informative points are included in the training set. To summarize, when *RIU* is properly parametrized, the resulting slight degradation of the theoretical performance of the classifier can be accepted in our endeavor to help the crowd generate more correct training labels and thus achieve a better overall performance of our hybrid intelligence system. Thus, only the verification of crowdworkers really profiting from *RIU* to reduce their labeling errors needs to be furnished, which will be the focus of Section 5.2.5.

5.1.4 Simulation of the Hybrid Intelligence System

After analyzing individual components of the methodology regarding the machine part of our hybrid intelligence system, we will now concentrate on a comprehensive study evaluating different classifiers and sampling strategies on our various data sets (cf. Section 4.2), where we will especially focus on ISPRS' benchmark data sets for semantic segmentation of 3D point clouds, i.e., V3D and H3D. Those exemplary AL runs are based on simulated oracles, which serve the purpose of providing an estimate of accuracies to be expected when working with real crowdworkers, as we strive to do in Section 5.2. The respective initialization data sets are constituted as discussed in Section 5.1.1, i.e., a total of 10 typical points far away from class borders are sampled for each class. All AL related entries in Table 5.2 - Table 5.4 are the accuracies obtained after $n_i = 30$ iteration steps with a batch size of $n^+ = 300$. As baseline, we always rely on the results achieved for the unseen test data set based on the respective classifier trained by the provided completely labeled training data set, i.e., PL. If not mentioned otherwise, both classifiers applied, i.e., RF and SCN, are parametrized as presented in the previous sections.

Method	Sampl. Method	Oracle	F1-score									**mF1**	**OA**
			Powerl.	L. Veg.	I. Surf.	Car	Fence	Roof	Façade	Shrub	Tree		
RF													
PL			48.39	**83.16**	**91.93**	72.68	14.94	**95.17**	**64.30**	40.60	**80.73**	65.76	**84.25**
AL	*wE*	$\mathcal{O}_O$	49.98	80.50	89.99	70.68	14.49	94.50	52.45	43.55	77.11	63.69	81.00
	wE+DiFS	$\mathcal{O}_O$	61.90	80.53	90.24	**73.12**	**28.58**	94.14	57.08	**43.55**	78.99	**67.57**	82.43
	wE+DiFS+RIU	$\mathcal{O}_O$	67.35	79.37	89.50	70.32	28.53	92.77	60.45	39.62	79.24	67.46	81.59
	wE+DiFS+RIU	$\mathcal{O}_N$	**68.85**	79.44	90.16	69.43	27.44	92.64	58.06	36.66	77.00	66.63	81.17
SCN													
PL			42.11	**81.40**	**91.11**	72.15	**41.22**	**94.10**	**59.65**	**48.87**	**83.88**	**72.92**	**83.86**
AL	*wE*	$\mathcal{O}_O$	65.17	78.29	88.96	68.86	25.32	88.39	49.58	34.49	76.81	63.99	79.07
	wE+DiFS	$\mathcal{O}_O$	60.57	79.31	88.59	72.28	24.92	91.21	55.34	43.44	80.16	66.20	81.13
	wE+DiFS+RIU	$\mathcal{O}_O$	**63.02**	79.52	89.62	**75.03**	26.33	91.18	54.41	38.45	78.27	66.20	80.91
	wE+DiFS+RIU	$\mathcal{O}_N$	60.68	78.89	89.48	74.09	22.29	90.64	53.77	39.10	78.54	65.28	80.59

Table 5.2: Comparison of reachable accuracies [%] for different training approaches and simulated oracles using RF and SCN for the V3D data set after 30 iteration steps.

Method	Sampl. Method	Oracle	F1-score											**mF1**	**OA**
			L. Veg.	I. Surf.	Car	U. Furn.	Roof	Façade	Shrub	Tree	Gravel	Vert. Surf.	Chim.		
RF															
PL			89.97	**88.17**	**63.76**	**49.18**	95.59	**78.08**	**65.86**	95.36	47.34	**59.63**	80.52	**73.95**	**86.87**
AL	*wE*	$\mathcal{O}_O$	87.04	79.33	49.48	42.15	93.17	74.72	63.22	95.12	46.65	27.40	**85.50**	67.62	81.63
	wE+DiFS	$\mathcal{O}_O$	**91.04**	85.93	59.74	43.64	**95.92**	76.40	64.41	**95.68**	**51.34**	54.80	82.97	72.90	86.58
	wE+DiFS+RIU	$\mathcal{O}_O$	88.38	85.97	55.68	44.07	93.75	75.64	66.46	95.56	49.69	55.53	63.59	70.39	84.84
	wE+DiFS+RIU	$\mathcal{O}_N$	88.06	86.94	56.01	42.88	93.93	75.78	64.43	95.14	46.67	56.17	50.26	68.75	84.82
SCN															
PL			**90.69**	**87.82**	55.17	**52.52**	**96.74**	**81.61**	**63.25**	**96.60**	50.55	**70.97**	63.24	**73.56**	**87.40**
AL	*wE*	$\mathcal{O}_O$	84.91	79.04	51.37	38.98	92.45	75.10	51.51	92.01	43.77	60.90	63.65	66.70	80.25
	wE+DiFS	$\mathcal{O}_O$	88.28	82.06	68.27	40.25	95.01	77.68	56.81	95.66	49.91	70.09	**74.64**	72.61	84.35
	wE+DiFS+RIU	$\mathcal{O}_O$	89.58	85.45	**68.36**	45.50	95.55	75.78	49.87	95.76	54.18	70.87	48.96	70.90	85.44
	wE+DiFS+RIU	$\mathcal{O}_N$	89.29	83.03	63.64	39.06	94.78	73.93	51.50	95.24	**54.59**	67.10	54.31	69.68	84.43

Table 5.3: Comparison of reachable accuracies [%] for different training approaches and simulated oracles using RF and SCN for the H3D data set after 30 iteration steps (RF) and 10 iteration steps (SCN), respectively.

We start our discussion with the V3D data set (cf. Table 5.2), which we have visited frequently in this chapter so far. For the RF classifier, generally speaking, we achieve state-of-the-art accuracy, with rather underrepresented classes (cf. Figure 5.2) and classes with a high inter-class similarity such as *Fence* and *Shrub* (see also Figure 3.3) yielding the worst F1-scores. Those classes are also most demanding for the SCN classifier, overall generating predictions of similar quality as the RF. Compared to pure *weighted entropy* sampling (*wE*) enhancing the selection strategy for both classifiers via *DiFS* significantly improves the respective OA value by approximately 2 pp. From the view of the machine, embodied by the RF, *DiFS* enhanced *wE* can be considered as optimal configuration that leads to a result that is less than 2 pp worse in OA, with only utilizing 1.15 % of labeled AL points compared to the PL run. For the SCN, accuracy scores are slightly worse compared to the RF and the gap between the mF1-score of PL and AL is severe, but still at the same level as for the RF classifier. This gap is mainly due to the drop in accuracy of class *Fence*, which seems to be more difficult to learn from only a few AL samples due to the great intra-class variety and inter-class similarity to class *Shrub*, for instance.

When employing a realistic crowd oracle, we would like to minimize the risk of label ambiguities beforehand by means of *RIU*, i.e., moving from the decision borders for the sake of better interpretability. Although from the view of the machine, this technique contradicts the AL idea, accuracy drops only marginally (less than 1 pp both in OA and mF1-score), but we hope to avoid a confused oracle $\mathcal{O}_S$ through *RIU*. Therefore, we only simulate a noisy oracle $\mathcal{O}_N$ with a 10 % noise rate for the final crowd simulation. Again, overall accuracy levels drop by less than 1 pp. Assuming that our simulated oracle is a valid approximation of a real

Method	Sampl. Method	Oracle	F1-score				**mF1**	**OA**
			U. Furn.	Ground	Building	Tree		
PL			**75.30**	**98.63**	**96.82**	**93.97**	**91.18**	**95.51**
AL	*wE*	$\mathcal{O}_O$	67.70	98.19	96.12	93.31	88.83	94.63
	wE+DiFS	$\mathcal{O}_O$	66.25	98.29	96.03	93.40	88.49	94.65
	wE+DiFS+RIU	$\mathcal{O}_O$	62.19	97.87	94.81	91.90	86.69	93.47
	wE+DiFS+RIU	$\mathcal{O}_N$	59.86	97.82	93.89	91.51	85.77	92.83

Table 5.4: Comparison of reachable accuracies [%] for different training approaches and simulated oracles using RF for the S3D data set after 30 iteration steps.

crowd oracle, we expect to be able to reach final predictions that are only about 3 pp worse in OA compared to the PL baseline, but which are highly efficient since only a small fraction of available training points will be dynamically labeled by the crowd within the fully automized hybrid intelligence system.

Compared to the V3D data set, H3D is even more challenging due to the more detailed class catalog employed to account for the much higher resolution of H3D. Thus, we adapt our RF classifier to this resolution by utilizing features computed for a neighborhood radius $r \in \{0.125, 0.25, 0.5, 0.75, 1, 2, 3, 5\}$ m. Concerning the AL loops incorporating the SCN classifier, they only run for $n_i = 10$ iteration steps with $n^+ = 600$ to ease the computational complexity as motivated in Section 5.1.1 and Figure B3 in the Appendix. As for V3D, the PL variant of both classifiers yields rather similar results and would be among the top submissions in the benchmark challenge[1] (cf. Table 5.3). Due to H3D's abundant point density and the associated occurrence of quasi-duplicates in the representation of points in feature space, the need for increasing the diversity in batches by means of *DiFS* becomes obvious with an improvement of > 4 pp in OA and > 5 pp in mF1 for both classifiers. Again, simulating a real crowd oracle by means of *RIU* (with $d_{RIU} = 0.75$ m as advocated for in Section 5.1.3) and 10 % of random noise reduces the accuracy of the final result in OA by less than 3 pp compared to the PL result, with only 0.12 ‰ (RF) and 0.08 ‰ (SCN) of points labeled.

Regarding the performance of RF vs. SCN, the final accuracies of the simulation are almost identical, but for the SCN this comes with a heavy computational load (cf. Section 5.1.2), so that we identify the RF classifier to be optimal for our hybrid intelligence system. In general, underrepresented classes seem to profit from AL techniques (e.g., V3D: *Powerline*, *Car* as well as *Fence* and *Shrub* in case of RF; H3D: *Car*, *Gravel* and *Chimney*), because the data set generated via AL is more representative and is almost equally distributed with respect to class affiliations of training points (cf. Figure 5.2), especially when our weighting scheme is applied (cf. Section 3.3.2). On the other hand, classes with a very high intra-class variety, in particular visible for classes *Urban Furniture* and *Façade* in H3D, suffer due to sampling only a small subset of points that fails to depict every aspect of this class.

Although having a much more coarse class catalog compared to the two benchmark data sets, S3D is tailored to assess the applicability of AL to large-scale, medium resolved data sets, where a much higher intra-class variability is inherent as a result of capturing a large selection of different representatives of each class. In our case, the whole downtown of Stuttgart is covered, in contrast to only small subsets of built-up areas as in V3D and H3D. Due to the small class catalog, this effect is even emphasized by unavoidable generalization (e.g., considering façades and roofs as class *Building*). Following the above findings, for the S3D, we restrict ourselves to reporting results of the RF classifier, where feature computation is based on neighborhoods of $r \in \{1, 2, 3, 5\}$ m just like for the V3D data set. Rooting from the intra-class variability, top accuracies over all classes are achieved by the PL approach utilizing the completely labeled training set, thus capturing a large variety of different class representatives. Vice versa, class *Urban Furniture*, having by far the highest intra-class variability, suffers the most from a limited training data set as utilized in AL (cf.

[1] at the time of writing this thesis

Table 5.4). Still, the optimal configuration from the view of the machine ($wE + DiFS$) performs less than 1 pp worse in OA compared to the PL approach, but with *DiFS* showing no impact at this saturated stage of the iteration after 30 iteration steps and with already well-sampled classes (nevertheless, *DiFS* leads to faster convergence, which is evident from the accuracy scores at an earlier iteration stage after 10 iteration steps; see Table B2 in the Appendix). As expected, manipulating the selections of the classifier by *RIU* slightly reduces accuracy as well as employing the noisy oracle $\mathcal{O}_N(10\,\%)$. Nevertheless, our AL pipeline, driven by a simulated but realistic crowd oracle, outlines the potential to reduce labeling effort to only 0.23 ‰ of available points, with OA values only about 3 pp worse in OA compared to the PL baseline.

5.2 Optimizing and Evaluating the Crowd Oracle

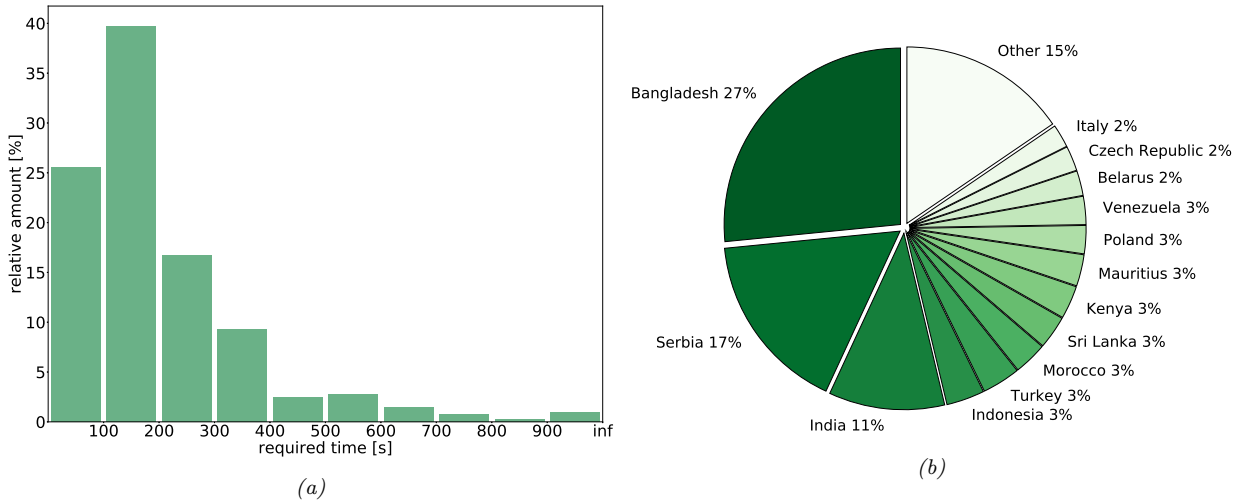
Apart from the machine component in our hybrid intelligence system, we will now turn to the *human processing unit*, i.e., to the crowd that is employed to teach the machine. Although we have postulated that the crowd is capable of dealing with 3D data annotation tasks, this still needs to be verified, which will be the subject of this section. First of all, we will evaluate the general ability of the crowd to label 3D data (cf. Section 5.2.2) and possible measures for quality improvement that do not require an expert to be involved (e.g., for checking crowd labels). Afterwards, we will focus on further techniques to improve crowd labeling, that is, experimenting with the use of 3D meshes as alternative data representation (cf. Section 5.2.3) and stimulating motivation by gamification elements (cf. Section 5.2.4). Finally, the benefit of *RIU* as a simplification of the crowd tasks is yet to be proven (cf. Section 5.2.5). Before going into the details of measures to improve labeling accuracy, we start with an analysis of whether crowdsourced labeling of 3D points is actually economically reasonable (cf. Section 5.2.1).

5.2.1 A Statistical Analysis of Crowdsourced Data Annotation Tasks

Unlike outsourcing complex, high-paying tasks such as programming to the crowd (Vignieri, 2021), the type of crowd jobs we offer are tedious labeling tasks that should take a short period of time to complete. This means, our goal is creating *microtasks* that can be processed in parallel by our *human processing units*. To find out whether respective labeling tasks can be completed in a timely manner and to determine the appropriate payment rate for our crowdworkers, we ran a first experiment where crowdworkers were asked to label a total of randomly picked 10 points each, and we recorded the time required for completion (campaign of *Type C*, cf. Section 3.6.2, for the point cloud modality of H3D's epoch March 2018, with a feasible class catalog as will be identified in Section 5.3.1 and a payment rate of $0.10 per job).

From Figure 5.5, we can observe that the majority of the 400 jobs that we posted were completed in less than 400 s (less than ca. 7 min), resulting in an average annotation time per point of about 40 s. A minimum completion time of about 33 s and a maximum completion time of 1190 s already give a taste of the accuracy to be expected. There are indeed crowdworkers that just try to skip through the job as fast as possible in their pursuit to make fast money, but there are also crowdworkers that really try to interpret the data they are confronted with instead of just picking classes randomly. Completion times that significantly exceed the average can also be due to a poor internet connection (resulting in slow loading progress of point cloud data) or crowdworkers being idle or occupied with other responsibilities.

With respect to determining the payment rate for our workers, when assuming that all crowdworkers can complete our jobs in about 10 min, according to Haralabopoulos et al. (2019), a remuneration of $0.23 would be appropriate. However, this payment rate is defined for *MTurk* where the majority of workers are U.S. residents with a rather high general level of income. Thus, we also need to take into account the countries of origin of our crowdworkers (cf. Figure 5.5*(b)*), which we have empirically determined from the pool of crowdworkers participating in our campaigns. The majority of workers are based in Asian countries, with Bangladesh and India being among the most represented nations. Also, a significant number of workers are located in Eastern European countries such as Serbia. Generally speaking, most crowdworkers live in states



(a) *(b)*

Figure 5.5: Statistics of our crowd campaigns. We recorded the average time to complete a crowd job *(a)* as well as the countries of origin of crowdworkers participating in our campaigns *(b)*. Please note that no completion times > 20 min were recorded, since this is the maximum allowed processing time after which the job gets aborted and assigned to another crowdworker.

with rather low incomes, where crowdsourced work can be a viable way to earn some extra money. Since our crowd is heavily biased towards Asian contributors, we adjust the payment accordingly, but aim to pay at least the minimum wage of their respective homeland. Thus, we consider a per job base payment of \$0.10 (with a bonus of another \$0.05; cf. Section 5.2.2) to be fair.

5.2.2 Evaluating the Performance of the Crowd as Labeling Engine

A crucial step to a completely automized system, at least from the view of the operator, is that despite the involvement of non-expert humans in the loop, no overhead is required to ensure data quality. This means that self-sufficient quality control techniques are needed, which can be drawn from the pool of *quality control on task designing* and *quality improvement after data collection*. As mentioned in Section 3.6.3, we implement those techniques by i) introducing check points into our tasks and ii) harnessing the *Wisdom of the Crowds* through majority voting. Nevertheless, the question remains of *how many* multiple acquisitions from different crowd members are required to be sufficient for our application at hand (i.e., when does the *Wisdom of the Crowds* kick in?).

Thus, we design an experiment where we randomly pick 20 points for each class of the H3D data set (epoch March 2018) that we have recognized to be feasible for crowdsourced annotation (an explanation will be given in Section 5.3.1) and compose a total of 20 crowd jobs, each job including one of the picked points of each class (every point only occurs once in those jobs). Additionally to the selected point, we present all points within a 20 m 2.5D neighborhood to allow interpretation in a *Type C* crowd campaign (cf. Section 3.6.2). Each of those 20 jobs is assigned to 20 different crowdworkers (i.e., $n_{mult} = 20$, cf. Section 3.6.3) granting a payment of \$0.10. After data collection, for each number of crowdworkers $n_{cw} \leq n_{mult}$, we form all possible combinations as outlined in Section 3.6.3 and we compute the averaged OA and class-wise F1-score as depicted in Figure 5.6*(a)*. The higher the number of multiple acquisitions that can be used for majority voting, the better the (averaged) accuracy of labels becomes until reaching an OA of almost 100 % when all 20 acquisitions are used, and also the higher the agreement between the individual workers that took part in the majority vote, expressed by means of *entropy* (averaged over all points and combinations for each n_{cw}). But for economic reasons, our goal is naturally to reduce the number of multiple acquisitions, which,

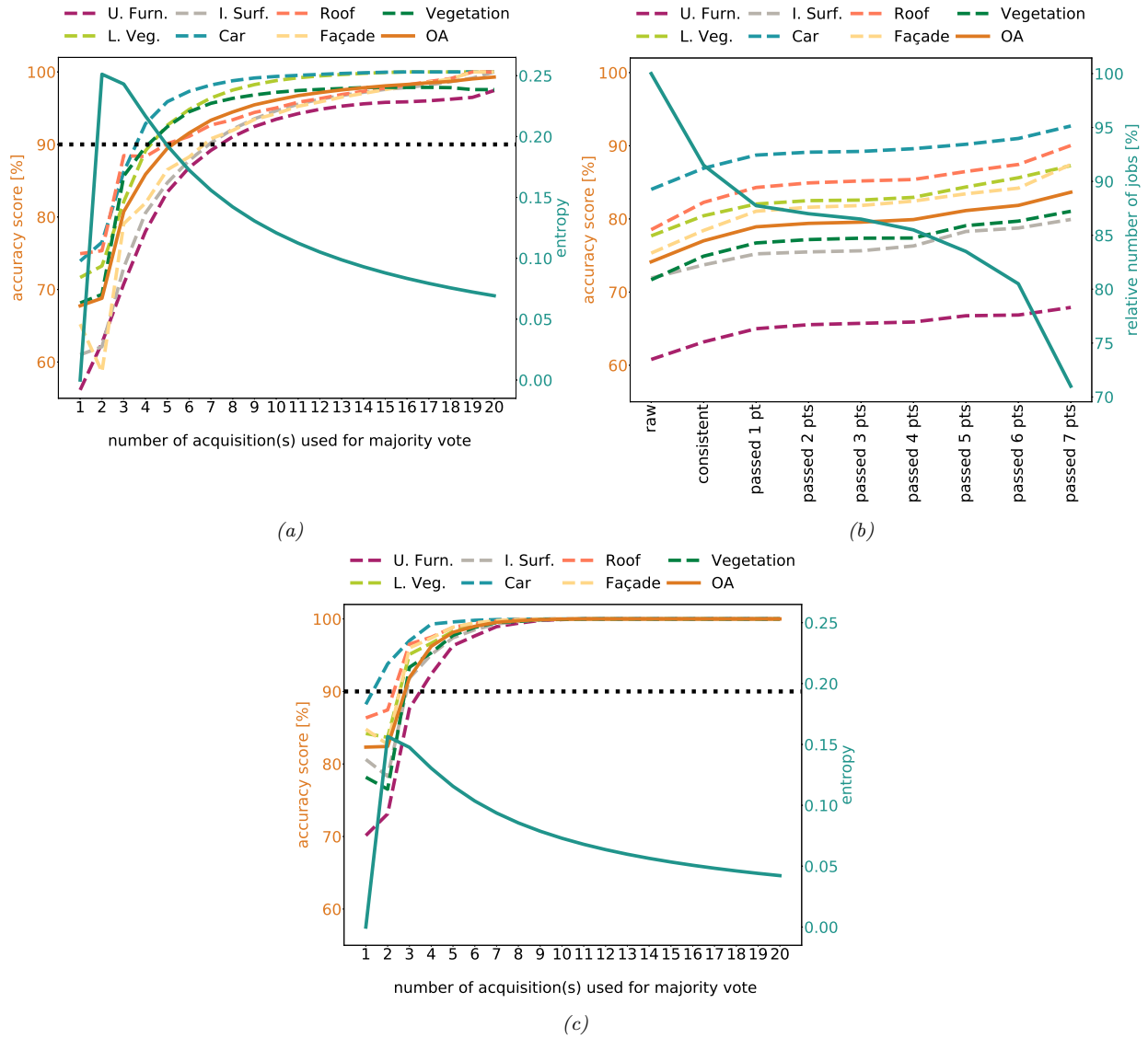


Figure 5.6: Compilation of different measures to improve the accuracy of crowd-generated labels either by majority voting *(a)*, check points *(b)*, and the consequent combination of both techniques *(c)*.

however, comes at the cost of label accuracy. Therefore, it is noteworthy that with $n_{cw} = 5$ votes the OA already exceeds 90 % and with $n_{cw} = 10$ votes it is 95 %.

As demonstrated in Section 5.1.3, our classifiers perform rather robustly against a certain rate of random noise. Precisely, 10 % of such noise almost does not affect the performance of the classifier at all. Thus, we consider an OA of 90 % as our optimal result, which can be reached when averaging $n_{cw} = 5$ (and more) votes. In other words, if no further measures to ensure quality are in place, each point needs to be labeled by $n_{cw} = 5$ independent crowdworkers. Considering a typical batch size of $n^+ = 300$ in an AL run with 10 iteration steps, 10 points per crowd job and a crowd payment of \$0.15, this already accumulates to \$225, which is the baseline payment we aim to further minimize. Additionally, Figure 5.6 *(a)* gives an idea of the complexity of different classes for the crowd, with *Urban Furniture* and *Façade* being the most difficult for interpretation. Both classes contain a big variety of different objects, where *Urban Furniture* practically

serves as class *Other* and *Façade* includes all kinds of façade applications such as balconies, lamps, etc., therefore also containing elements of class *Urban Furniture*. Interestingly, *Impervious Surface* also seems to be of similar complexity, which comes as a surprise since this is a well-defined class. But whenever the colorization of points actually belonging to this class is unclear, e.g., partly mud-covered, overgrown, or simply shaded, crowdworkers hastily decide for *Urban Furniture*. Easily understandable and distinguishable classes such as *Car*, *Low Vegetation* or *Vegetation*, on the other hand, can be labeled by the crowd with rather high quality.

As means to reduce the number of required multiple acquisitions to reach an OA of 90 %, we rely on *quality control on task designing* through check points, i.e., points for which we know the true class affiliation, but which appear just like payload points to crowdworkers. As minimum test to be passed, one of the check points is presented twice to crowdworkers and required to be labeled the same (but not necessarily correctly). This helps to identify random assignment of classes, but does not prevent workers from labeling all points the same. Thus, the next step in extending control is that this point must be labeled correctly and that additional points are also properly labeled (referred to as *passed 1 pt* and *passed [x] pts* respectively in Figure 5.6*(b)*). To evaluate the impact of this measure, we posted a new campaign that included 7 more (check) points compared to the previous campaign, where the submitted results had no effect on passing the job.

After completion of the campaign, however, we used these check points to assign the results to different quality levels to generate Figure 5.6*(b)*. As expected, the stricter the check mechanisms are (i.e., the more checks must be passed), the better the accuracy of labels becomes. In this case, accuracy is computed over all crowd labels, i.e., without majority decision, from the pool of crowd labels and the corresponding GT label for each point. This means that a selected 3D point contributes up to 20 times to the final accuracy value reported in Figure 5.6*(b)*, depending on whether a respective job containing this point passed the checks. On the other hand, the stricter our checks are, the more jobs we are forced to discard without granting workers any payment (up to 30 % with 7 check points). In order not to deter crowdworkers by a high rejection rate of our jobs, we also try to minimize the number of check points and identified passing 3 check points (including the consistency check) as a good compromise between accuracy and rejection rate. By relying on check points, we reject about 13 % of low-quality acquisitions while improving the accuracy level by about 5 pp (cf. Figure 5.6*(b)*).

After identifying 3 check points as reasonable means to control accuracy on data acquisition, we repeated the first experiment where each job is posted to $n_{mult} = 20$ different crowdworkers, but this time the result is only accepted if crowdworkers meet the minimum requirement of passing those checks. As further incentive, we grant a bonus of \$0.05 on the advertised payment of \$0.10 in case of successful check point labeling, i.e., the effective payment is \$0.15. Using only those assumed to be high-quality results, we repeat the process of computing the average accuracy values of all possible combinations as discussed in Section 3.6.3. As anticipated, this yields a much faster convergence of our accuracy curves in Figure 5.6*(c)* compared to Figure 5.6*(a)* and also a higher level of agreement depicted by the *entropy* curve. When using $n_{cw} = 9$ (and more) votes, an accuracy close to 100 % can be reached. But most crucially, the 90 % OA level is already reached after $n_{cw} = 3$ votes, not $n_{cw} = 5$ as before. This means a reduction of the assumed campaign cost from $\$15 = 20\,\text{jobs} \cdot 5\,\text{workers} \cdot \$0.15$ to almost half ($\$9 = 20\,\text{jobs} \cdot 3\,\text{workers} \cdot \$0.15$) and a reduction from \$225 to \$135 for the aforementioned AL iteration. Consequently, for all following campaigns, we will assign each point identified by the machine as informative to a total of $n_{cw} = 3$ different crowdworkers, each having passed 3 check points including the consistency check.

5.2.3 The Impact of Different Data Modalities on the Performance of the Crowd

While GT inference measures such as majority voting (cf. Section 5.2.2) are an efficient means to improve label accuracy by integrating the results of multiple crowdworkers, these techniques are limited by the accuracy of the (filtered) results generated by individual crowdworkers. Thus, in order to improve data quality in the first

place, as motivated in Section 3.6.4, we utilize 3D textured meshes as alternative data modality that might be easier for non-experts to interpret. To fairly compare which modality is better suited for crowdsourced data collection, we repeat the third experiment conducted in the previous section, i.e., posting 20 jobs with one sample of each of the 7 classes (and 3 additional check points including the consistency check) to a total of 20 different crowdworkers granting a \$0.10 + \$0.05 payment. But in this case, the vicinity of the randomly selected points is visualized by a cylindrical 2.5D neighborhood cropped from the 3D textured mesh ($r = 20\,\text{m}$) in our *Type C* labeling tool (cf. Section 3.6.2). To generate a meaningful comparison of data of the same level of quality, we would like to emphasize again that in both campaigns, results are filtered based on check points. Otherwise, we might be confronted with one campaign including results from a tremendous number of malicious or careless crowdworkers, while this number might be negligible for the other campaign, thus leading to non-comparable results.

To evaluate the respective accuracies of both campaigns, we compute confusion matrices from the pool of crowd label-GT pairs and consider each crowd label individually (i.e., no majority voting). Thus, our results are statistically supported by 400 points per class, for a total of 2800 points (20 jobs · 7 points · 20 workers). Figure 5.7 demonstrates that by purely switching the data modality (i.e., replacing a point cloud oracle $\mathcal{O}_{CP}$ with a mesh oracle $\mathcal{O}_{CM}$), the overall labeling accuracy can be improved by 4 pp in our case of randomly selected points. While all classes profit, especially the F1-scores of *Urban Furniture* and *Vegetation* are improved. For both classes, fewer points are confused with other classes, and particularly the confusion between those two classes is significantly reduced. This might be due to the much clearer depiction of the 3D environment by applying realistic textures, helping to distinguish between man-made (urban furniture) and natural objects (vegetation).

However, higher rates of confusion remain when points near class borders are selected (e.g., *Low Vegetation* vs. *Impervious Surface* and *Roof* vs. *Façade*). Although error rates have been reduced through relying on a crowd oracle utilizing mesh data $\mathcal{O}_{CM}$, higher error rates still appear for samples from classes that are hard to separate from other classes such as *Urban Furniture* quasi serving as class *Other*. While the latter is an inherent problem, the issue of labeling at class borders could be alleviated by the *RIU* technique (cf. Section 3.3.4). Nevertheless, those experiments emphasize that a high-quality 3D mesh generated from concurrently captured imagery and LiDAR data is to be preferred when non-experts are tasked with interpreting 3D data. This is the reason why we also employ mesh data whenever possible (i.e., in case of the H3D data set) in context of our AL-based learning schemes in Section 5.3.2.

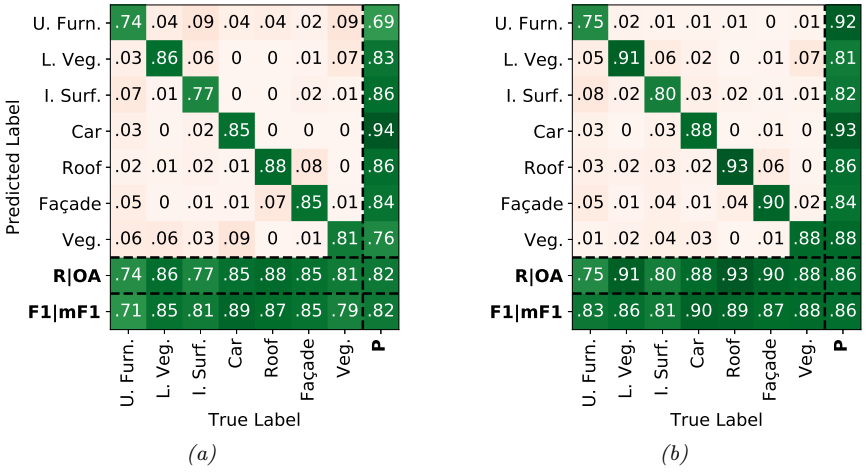


Figure 5.7: Confusion matrices computed from crowd labels generated utilizing either the point cloud *(a)* or the mesh modality *(b)* to illustrate the vicinity of selected points.

5.2.4 Can the Crowd be Tricked by Gamification?

Another way to improve (raw) label accuracy and crowdworkers' willingness to perform work is to attempt to intrinsically motivate crowdworkers by means of gamification (cf. Section 3.6.5). To evaluate whether such elements have an impact in context of paid crowdsourcing, we launched various crowd campaigns, each containing one of the four gamification elements discussed in Section 3.6.5, and one incorporating all of them. As baseline solution, we conducted the same campaign without any gamification elements. Each crowdworker is confronted with the same task, that is, labeling 20 pre-selected faces from the H3D mesh data (epoch March 2018), and has the option to repeat the task. All campaigns discussed in this section were processed by the same *MW* group (*International Workers*), launched at the same time of day to guarantee no significant change of nationalities in the participating group, and were paid equally with \$0.10. Each campaign consists of contributions from 90 crowdworkers. Please note that no check points are included in those tasks, since alongside improving quality, we aim to encourage crowdworkers to do more work with gamification elements, and adding annoying check points might quickly make our gamified tools lose their appeal. The downside of this approach is that the results obtained could be dominated by crowdworkers who perform exceptionally well or poorly, so variations are inevitable to some degree. As a remedy, campaigns are posted multiple times to check for reproducibility and thus the resilience of the results.

In order to get a feeling for whether such gamification elements show any effect, Table 5.5 depicts the results of one trial for each campaign. In terms of the OA values, results are relatively stable - at least considering that no quality control measures are applied. This means no improvement in labeling accuracy is to be expected through gamification, which is a sobering finding. On the other hand, this can be interpreted to mean that crowdworkers generally try to perform as well as possible, even without additional game incentives. Possibly, avoiding reputational degradation due to poor performance is a greater motivation than such gamification elements. Thus, the accuracy is not so much limited by the will of crowdworkers, but by their ability to perform rather complex 3D labeling tasks.

On the other hand, as for the amount of completed work, gamification brings an advantage. Recall that in all campaigns, crowdworkers are only required to label a total of 20 faces (i.e., to complete one full job). Even in the baseline campaign without gamification elements, crowdworkers output more labels - probably hoping for a bonus payment. In this context, assigning a score to a crowdworker and comparing it with others seems to encourage crowdworkers to contribute more (cf. Table 5.5). The same applies to making the labeling experience more fun by means of audio-visual effects boosting the stamina of workers before growing tired of labeling. Although progress bars might be good indicators of keeping track of completed work, they do not seem to stimulate motivation. Strikingly, while each gaming element separately increases the amount of completed work only slightly, the combination of all elements seems to trigger a cumulation of individual effects and leads to an average number of faces that is more than double that of the baseline campaign. Even though explaining such phenomena might be more suited for experts in working psychology, we assume that this is mainly due to pairing scoring of results (i.e., triggering anticipation of workers for a reward) with

campaign		∅ number of labeled faces per crowdworker	OA [%]
no gamification		29.6	74.55
gamification	progress bar	30.9	65.65
	score	37.6	72.37
	high score list	38.7	74.97
	audio-visuals	33.1	68.72
	all elements	73.3	72.91

Table 5.5: Effect of different gaming elements on the labeling results delivered by crowdworkers.

ID	worker group	method	∅ number of labeled faces per crowdworker	OA [%]	time to complete [h]
M1	all workers	without gamification	26.6	74.1	3.4
M2			26.6	75.1	1.5
M3			25.1	74.1	2.1
MG1		with gamification	54.4	66.8	2.8
MG2			45.5	71.3	1.6
MG3			30.8	73.5	1.9
O1	one-time workers	without gamification	33.5	65.6	6.7
O2			30.0	73.6	24.6
O3			30.2	65.1	64.1
OG1		with gamification	58.8	62.6	13.4
OG2			73.1	68.9	18.1
OG3			63.3	67.2	41.9

Table 5.6: Statistics for different crowd campaigns. We distinguish between user groups where an individual crowdworker can participate in multiple campaigns or just in one and test the effect of gamification elements on both groups.

immediate feedback through audio-visual effects (i.e., giving them a feeling of satisfaction/spurring ambition in case of correct/false answers). At the same time, the progress bar shows workers how close they already are to their goal of finishing the task and getting their remuneration.

However, due to the lack of quality assurance measures in those campaigns, there is still a chance that these results are of purely random nature, caused by some extremely motivated workers. Thus, we repeated both the campaign without any gamification elements and the campaign with all gaming elements three times. Table 5.6 summarizes the results of those campaigns, where crowdworkers were allowed to participate in as many campaigns as they pleased (cf. M1-M3 vs. MG1-MG3). As for the multiple trials of the non-gamified baseline (denoted as M1-M3 in Table 5.6), the results seem to be robust, with OA values rather similar to the initial trial reported in Table 5.5 and a slightly lower average number of faces per crowdworker. This verifies our findings with respect to the baseline campaign.

Regarding the gamified counterparts (denoted as MG1-MG3 in Table 5.6), OA values show some variance, which is to be expected due to lack of control mechanisms. Differences can be caused, for instance, by individual crowdworkers being overwhelmed with the 3D data representation, not understanding what is asked of them, or intentionally performing poorly. What is striking is a clear trend of a decreasing average number of labeled faces, starting with our initial campaign (cf. Table 5.5) and continuing throughout our reproducibility tests (cf. MG1-MG3 in Table 5.6). But there seems to be a slight trend of increasing OA for those gamified trials.

To explain both phenomena, Figure 5.8 plots the relative number of workers participating for the first time in one of our campaigns, so-called one-time workers, to the average number of labeled faces and OA values. As expected, the number of one-time users decreases because the more campaigns we launch on *MW*, the more likely we are to face a crowdworker who has also participated in one of our previous campaigns. This decreasing number of one-time workers is i) positively correlated with the average number of labeled faces per worker and ii) negatively correlated with the achieved OA values. The latter is obvious - the more frequently a crowdworker performs a specific task (even if it is a complex and unfamiliar one), the better his performance gets. The drop in the average number of labeled faces per crowdworker, on the other hand, means that our game elements seem to lose their appeal when encountered frequently. But this is also the case with most commercial games, that people stop playing as soon as there is no new content. In our case of labeling, however, this effect comes more quickly since basically every task is the same (although different

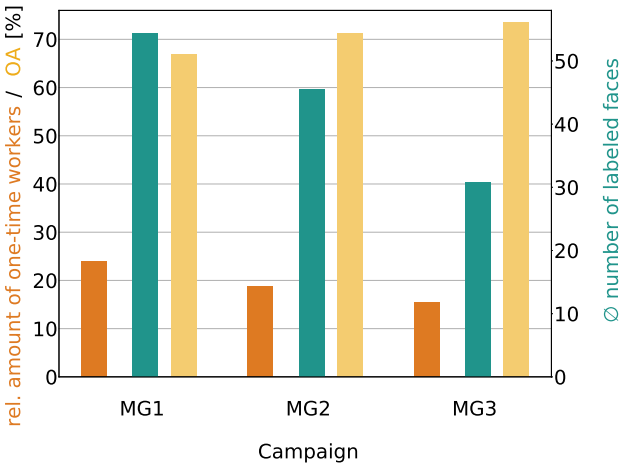


Figure 5.8: Comparison of the relationship between the relative amount of one-time workers in campaigns MG1-MG3 and campaign results evaluated for 3 trials of the same gamified campaign (see also Table 5.6). As campaign indicators, we consider i) the achieved OA as well as ii) the average number of labeled faces per worker, showing a positive and a negative correlation to the number of one-time workers, respectively. Please note that campaigns MG1-MG3 are preceded by our initial, fully gamified campaign (cf. Table 5.5), so that crowdworkers can be faced again in MG1, resulting in presence of non-one-time users.

faces are randomly chosen in each trial), so users grow tired of it after their first round. This is also identified by Thaler et al. (2012) as one of the major challenges in gamifying categorization tasks.

To counter this effect, we relaunch both the non-gamified baseline (cf. O1-O3 in Table 5.6) and the gamified campaign (cf. OG1-OG3 in Table 5.6) in three trials, but now crowdworkers are allowed to participate in only one campaign in this new series of campaigns, i.e., we are confronted only with one-time workers. We can observe that the obtained OA values are more similar to the minimal values achieved in the previous campaigns. This is due to the fact that none of the crowdworkers participating in those campaigns has any prior knowledge of the task and hence lacks experience. With respect to the number of labeled faces, results of the non-gamified campaigns are stable and similar to our initial campaign (cf. Table 5.5). As hoped for, the number of processed faces over all trials for our gamified campaigns is on a level that is twice as high as for the baseline campaigns. This underlines that our game elements are indeed suitable for stimulating motivation of workers. A downside, however, comes in the time required to complete such campaigns. Since we reject workers who have already worked in a campaign, we effectively shrink the pool of available and active workers. While all previous campaigns could be completed in a maximum of 3.4 h, filtering crowdworkers prolongs the campaign durations to up to 64.1 h. This can be an issue whenever we are confronted with time-critical labeling tasks or, considering our hybrid intelligence system, when it is costly having the machine component stay idle (e.g., in case of large-scale supercomputers, often used for ML applications).

Thus, to summarize, by means of gamification we receive more labeled faces, but on an accuracy level identical to that without gamification. However, from an ethical point of view, enticing crowdworkers with already rather low payment rates into free labor is questionable, which is why we do not further pursue the concept of gamification in the following more extensive experiments.

5.2.5 The Impact of *RIU* to Ease Labeling

Although the aforementioned techniques (especially the quality control measures in Section 5.2.2 and relying on alternate data representation tested in Section 5.2.3) can help improve accuracy, they will not succeed whenever there is an inherent label ambiguity of the selected points. In AL, we must expect that this applies

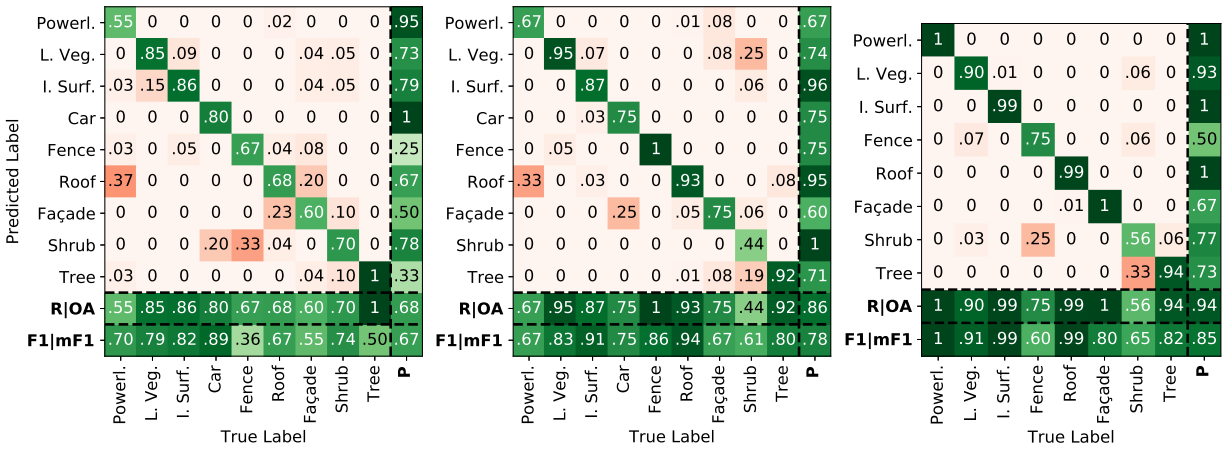
to the majority of selected instances since we tend to draw samples from the vicinity of decision borders in feature space, which often correspond with class borders in object space (cf. Section 3.2). As discussed in Section 3.3.4, the *RIU* sampling add-on should be capable of resolving those ambiguities while only marginally affecting the performance of the classifier, driven by the generated labels (cf. Section 5.1.3). Thus, the only question remaining to be examined is the anticipated increase in labeling accuracy due to *RIU*. To this end, the straightforward idea is to compare overall labeling accuracies over complete AL loops for different d_{RIU} values (the maximum distance allowed to the classifier-selected seed point). But this procedure does not allow to compare the results fairly, because not only the *RIU*-selected points, but also the seed points queried by the classifier would differ, as AL loops would diverge quickly with relying on different point samples for training.

Thus, to fix seed points, for this examination, we should focus on one dedicated iteration step in such an AL loop for which different d_{RIU} values can be applied in the querying step to receive respective point batches for crowd evaluation. Optimally, those batches (or at least the set of seed points) can be described by an equal distribution of classes, which, however, rarely occurs within a single iteration step of an AL loop. Nevertheless, relying on a non-equally distributed, machine-generated batch is still a better option compared to manually selecting seed points. Those points would be highly biased towards human operator understanding and thus not representative for the samples selected in our AL runs, and would also undermine the significance of the results obtained. Thus, our selection of seed points is accomplished by RF-driven *wE+DiFS* sampling (RF parametrization as outlined in Section 5.1.1) for the V3D data set.

Consequently, a representative batch of a specific iteration step containing samples for every class but showing an imbalance between classes was selected. In this batch with $n^+ = 300$ samples, most underrepresented classes are *Car* (6 samples) and *Fence* (7 samples), while class *Roof* is the most represented class (85 samples). As an indicator of class balance, we derived an *entropy* value of the relative class occurrence histogram of 2.82 compared to an optimal value of $-\log_2(1/9\,\text{classes}) = 3.17$ for a true equal distribution. Building upon the classifier-selected seed points, we apply different values for $d_{RIU} \in [0\,\text{m}, 1.5\,\text{m}, 4\,\text{m}]$ and distribute the $n^+ = 300$ points equally into 30 jobs with 10 payload points each but also equipped with 3 check points and the consistency check (cf. Section 3.6.3). The final label for each point is inferred by majority voting from $n_{cw} = 3$ votes and compared to GT labels to generate the confusion matrices, as depicted in Figure 5.9. This allows for a fair comparison of the effect of *RIU* on label accuracies under the same conditions as in our aspired AL runs.

As anticipated, *RIU* is indeed able to increase the achievable label accuracy of crowdworkers by resolving label ambiguities and thus minimizing the risk of a confused oracle $\mathcal{O}_S$ being encountered. Using a maximum distance to the machine-selected seed point of $d_{RIU} = 1.5\,\text{m}$ already boosts label accuracy by 18 pp in OA, or by 26 pp for $d_{RIU} = 4\,\text{m}$. Although $d_{RIU} = 4\,\text{m}$ seems tempting with regard to crowd labeling accuracy, Figure 5.3*(b)* demonstrates that this distance is far too excessive to actually generate labels suitable for training an effective classifier. This is also confirmed by the lack of class *Car* in the confusion matrix, since *Car* points are completely skipped through *RIU* with $d_{RIU} = 4\,\text{m}$. Nevertheless, such a strong simplification is well suited for uncovering most demanding label queries to the crowd (cf. Figure 5.9*(c)*). Those difficulties are due to a class catalog that may be challenging for crowdworkers to comprehend, which becomes obvious for classes *Fence* vs. *Shrub* and *Shrub* vs. *Tree*. While it is often demanding even for an expert to distinguish between these classes, this issue is even emphasized when crowdworkers from Asian countries (cf. Figure 5.5*(b)*) are confronted with vegetation types from Middle European countries that are unfamiliar to them.

When comparing the confusion matrix of $d_{RIU} = 1.5\,\text{m}$ to the one without applying the *RIU* sampling add-on (i.e., $d_{RIU} = 0\,\text{m}$), we can observe that *RIU* is capable of resolving typical class ambiguities between adjacent classes in object space (e.g., *Low Vegetation* vs. *Impervious Surface* and *Roof* vs. *Façade*). But still, vegetation related classes and the distinction between classes *Powerline* and *Roof* remain difficult for crowdworkers. The latter could be caused by the suboptimal colorization of the V3D data set by an orthogonal projection of an orthophoto, yielding "roof-colored" *Powerline* points that are hard to recognize. Please note that the reason for high and fluctuating rates of confusion, especially for classes *Car* and *Fence*, is mainly



(a) $d_{RIU} = 0\,\text{m}$ *(b)* $d_{RIU} = 1.5\,\text{m}$ *(c)* $d_{RIU} = 4\,\text{m}$

Figure 5.9: Obtained confusion matrices of crowd-generated labels for different distance values d_{RIU} used by the *RIU* sampling add-on to select alternative points instead of those suggested by the machine. Results are evaluated for an exemplary and representative iteration step for the V3D data set.

due to the aforementioned underrepresentation of these classes in the respective batch to be labeled. While *RIU* shows a clear potential to boost the accuracy of crowd-generated labels for the presented toy example, only employing it in our crowd-based AL loops, reported in Section 5.3, will show the true effectiveness in a large-scale test.

5.3 Formulating the Hybrid Intelligence System

After evaluating the performance of the machine part (cf. Section 5.1) and the human aspect (cf. Section 5.2) separately, the findings of those experiments pave the way for employing the desired hybrid intelligence system. Thus, we are now able to run AL loops with the oracle being constituted by the crowd, completely relieving an expert annotator from such duties. This hybrid intelligence system is utilized for conducting in-domain AL loops on various test sites (cf. Section 5.3.1). But we also aim to transfer knowledge obtained from crowdworkers between different data sets through ATL (cf. Section 5.3.3), as this technique shows potential to reduce labeling effort even more than pure AL. Whenever possible, we utilize measures specifically designed for the crowd-powered AL or ATL runs to ease the labeling process for crowdworkers. This involves both addressing ambiguous samples through the *RIU* technique (cf. Section 3.3.4) and employing alternative data modalities (cf. Section 5.3.2).

5.3.1 Employing the Crowd-Driven AL Loop for Efficient Segmentation of ALS Point Clouds

Recalling the overall aim of enabling ML by an efficient and fast generation of training data through paid crowdsourcing, this section is dedicated to evaluating the performance of a hybrid intelligence system formed by combination of an ML algorithm and the crowd, merged by means of AL. Thus, this section can be considered as reiteration of parts of the experiments conducted in Section 5.1, differing, however, by the fact that we no longer rely on simulated oracles, but on real crowd oracles $\mathcal{O}_C$. Again, we test the presented framework for all three test sites with different characteristics (cf. Chapter 4), especially focusing on the individual and joint performance of the crowd in interplay with the machine learning from *the human processing units*. Please note that results reported in this section exclusively address crowd oracles that were provided with point cloud data, i.e., $\mathcal{O}_{CP}$. Also, all the experiments discussed in the following are run

utilizing the CATEGORISE framework, allowing a fully automated execution - at least from the operator's point of view.

Initializing the Loop

Due to our overall goal of excluding an expert operator from labeling tasks and as outlined in Algorithm 1, the crowd is responsible for generating the initialization data set necessary to kick off the AL run, i.e., the interactive communication between crowd and machine in the first place. Thus, by utilizing our web tool of *Type A* (cf. Section 3.6.2), for each data set (namely V3D, H3D epoch March 2018 and S3D), the crowd is asked to identify one point for each class. Precisely, 100 crowdworkers ($n_j = 100$) are tasked at a payment rate of \$0.10. The respective confusion matrices can be found in Figure 5.10*(a)* for the V3D data set and in the Appendix Figure B6*(a)* and B7*(a)* for the H3D and S3D data sets, respectively. Unsurprisingly, a lot of crowdworkers deliver insufficient labels, causing a rather low OA of about 53 % (V3D), which is due to no quality control mechanisms being in place. Although these unchecked initial results should be taken with a grain of salt, a general trend with respect to achievable accuracy for each data set still becomes obvious. Naturally, for S3D, the crowd performs better as we are dealing with a significantly simpler classification task than for the other two test sites (less and more general classes). Interestingly, accuracies of H3D exceed those of V3D by far, although being a labeling task of similar complexity. This is likely because of the more realistic and thus more familiar depiction of the point cloud data in case of H3D due to the much higher point density (cf. Table 4.1).

To refine generated labels, the next step is to filter the results obtained in the first place by a second group of crowdworkers (cf. Algorithm 1). This is accomplished by means of our *Type B* labeling tool (cf. Section 3.6.2). Here we ask a total of $n_{cw} = 3$ crowdworkers to check each point at a payment of \$0.10+\$0.05 (base payment + bonus) in the sense of majority voting (also, 3 check points are utilized, with one being shown twice, to receive only high-quality results; cf. Section 5.2.2). Since crowdworkers only need to provide *Correct*/*False* answers, this can be considered a two-class categorization problem, whose accuracy can thus be evaluated by the same metrics as our multi-class experiments. Table 5.7 gives an overview of the results from these campaigns after aggregation of answers from $n_{cw} = 3$ crowdworkers by means of majority voting. Since we consider these campaigns as *filters*, recall values are of special interest for detecting both false and

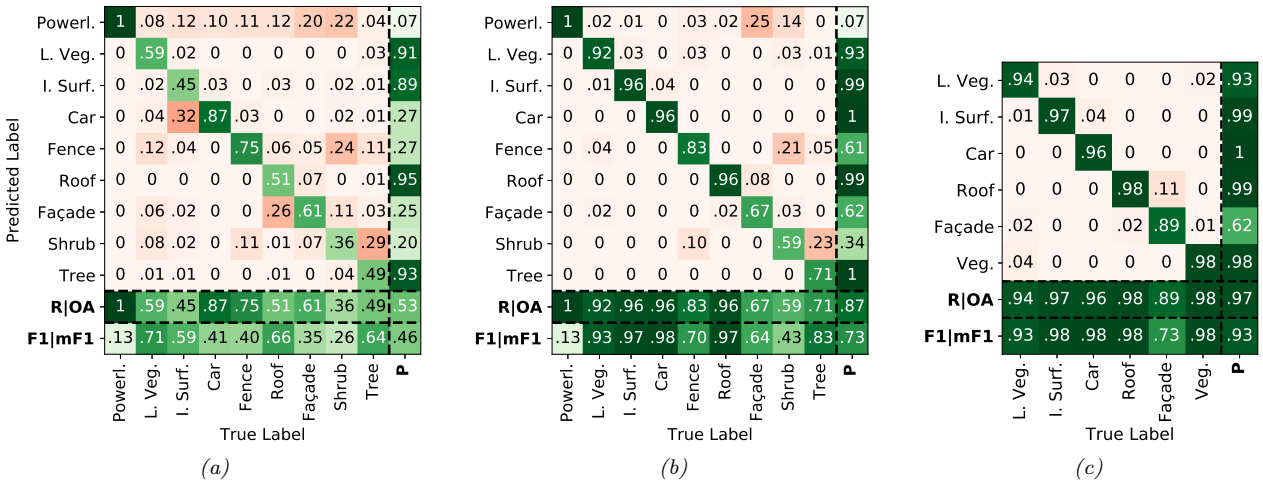


Figure 5.10: Labeling accuracy of the crowd in context of the initialization of an AL loop on the V3D data set. The accuracy of the raw and unchecked crowd labels *(a)* are verified by a second group of crowdworkers in a crowd campaign of *Type B* to obtain refined results *(b)*. To further ease the labeling complexity, we restrict the class catalog and merge classes accordingly *(c)*.

Data set	Recall		OA	rel. amount of remaining samples
	False	*Correct*		
V3D	87.94	98.05	91.66	57.60
H3D	60.31	97.31	84.44	78.80
S3D	85.71	98.96	95.25	75.25

Table 5.7: Statistics of our *Type B* crowd campaigns for improving the initially assigned labels by crowdworkers. The relative amount of remaining samples refers to all points that are marked by the crowd as labeled correctly and thus remain for the initialization data set. All measures are given in [%].

correct labels. While almost no correctly labeled points are marked as incorrect by the crowd, crowdworkers struggle to identify false labels. In other words, crowdworkers seem to hesitate to flag points as incorrect if they are not completely sure about the correct answer. In this regard, the crowd performs worst on the H3D data set, which is actually supposed to give the easiest representation of data. Partly this is due to a higher initial level of accuracy compared to the V3D data set (cf. Figure 5.10 vs. Figure B6 in the Appendix), but mainly the poorer performance is caused by more obvious errors in the other data sets.

V3D and S3D rely on a suboptimal colorization by orthogonal projection of a temporally disjoint orthophoto. Among other issues such as roof-like colorization of façades, this is especially problematic for dynamic objects such as cars (cf. Figure B8 in the Appendix). Often, points geometrically depicting streets, but radiometrically representing cars (due to mapping of car color values to streets), are mistaken for cars in the *Type A* campaigns. With a closer look at the point cloud and with the simplified 3D point cloud navigation requirements in our *Type B* campaigns, such errors can be easily detected. Thus, data sets with non-optimal colorization perform better in this evaluation of *Type B* campaign results (cf. Table 5.7) since the errors are more obvious. After checking results, all points that are marked as *False* are eliminated, which is why a high recall value for *Correct* is particularly desirable (i.e., dropping true labels should be minimized). On average about 30 % of initially collected labels are discarded in the process (cf. Table 5.7) and *filtered* confusion matrices can be built for all data sets (cf. Figure 5.10*(b)* and Figure B6*(b)* and B7*(b)* in the Appendix). As can be seen, we can improve OA of our crowdsourced labels by about 24 pp on average, underlining the impact of such quality control tools, but this improvement in accuracy comes at the expense of the size of the initial training set.

Nevertheless, some errors remain, especially for the V3D and H3D data sets, containing rich class catalogs. This means that some classes are hard to comprehend for our crowdworkers, being in accordance to the findings of Bayas et al. (2016), stressing a limited feasible class catalog for crowdsourced data acquisition. However, these errors are more due to ambiguous classes rather than hard label errors, e.g., in case of V3D, *Shrub* vs. *Tree* and *Fence* vs. *Shrub*, but also due to insufficiently and sparsely depicted objects such as powerlines. Since distinguishing between such classes is hard to communicate to crowdworkers and even experts might argue about it, for V3D, we decide to merge classes *Fence*, *Shrub* and *Tree* into class *Vegetation* and merge class *Powerline* with class *Roof*. Similarly, for H3D, classes *Shrub* and *Tree* are merged into *Vegetation*, *Chimney* is added to *Roof* and *Soil/Gravel* to *Low Vegetation* (class *Vertical Surface* was merged directly with *Façade* in the first place, following a similar arguing). Naturally, this step further improves accuracies, where remaining confusion for H3D mainly happens between quasi-class *Other* (i.e., *Urban Furniture*) and all other classes (cf. Figure B6*(c)* in the Appendix) and *Roof* vs. *Façade* in case of V3D (cf. Figure 5.10*(c)*). The latter is also due to the aforementioned merging of classes *Powerline* and *Roof*, as crowdworkers originally sometimes tended to label sparsely discretized linear point agglomerates as *Powerline* instead of *Façade* (cf. Figure B9 in the Appendix). Nevertheless, training data sets with an average OA over all our test sites of 95 % can be utilized for initializing the AL loop.

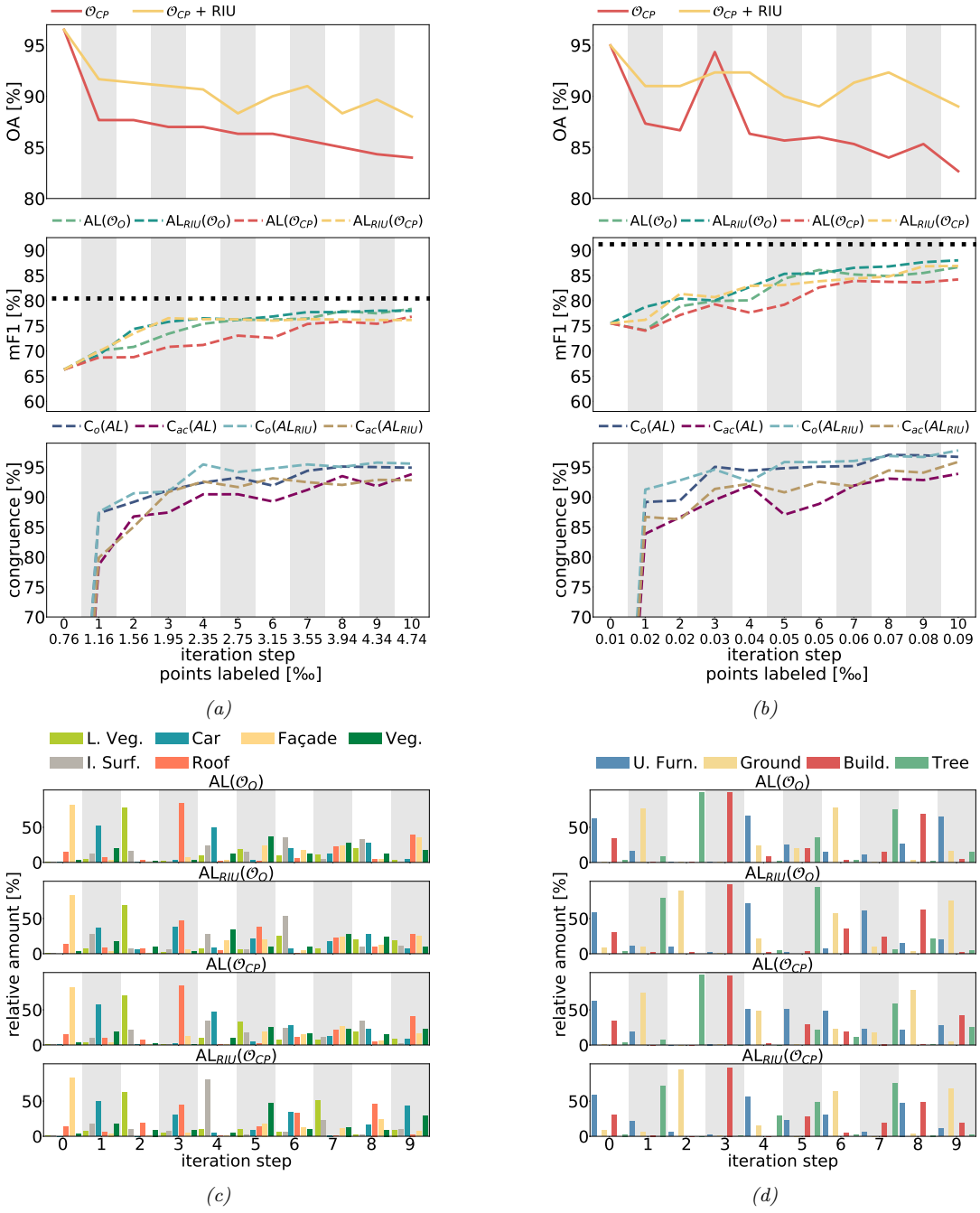
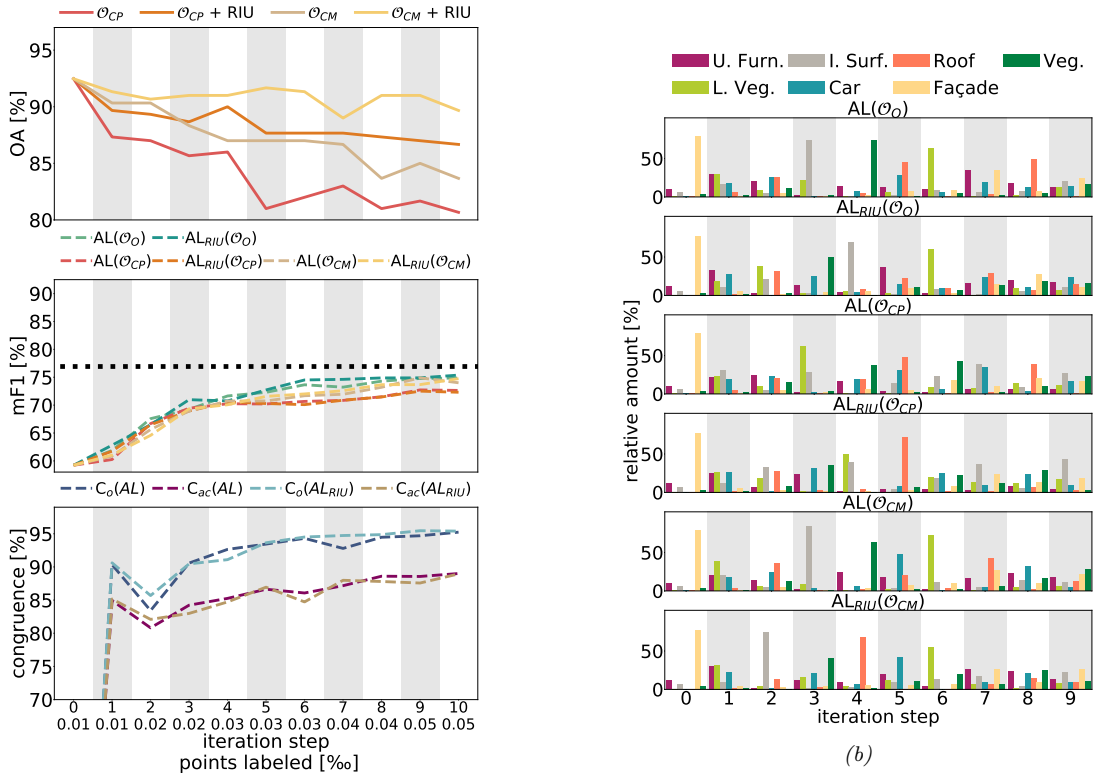


Figure 5.11: Overview of the achieved performance of our hybrid intelligence system for both the V3D (*first column*) and S3D data set (*second column*). In *(a)* and *(b)* we report the achieved labeling accuracy of the crowd (*first row*) as well as the accuracy of the machine learning from the crowd over all iteration steps (*second row*). *Dotted black lines* represent the result of PL. Additionally, our congruence measures C_o and C_{ac} are evaluated for the runs relying on real crowd oracles $\mathcal{O}_{CP}$ to test the feasibility of our stopping criterion, i.e., $AL(\mathcal{O}_{CP})$ corresponds to $C_o(AL)$ & $C_{ac}(AL)$ and $AL_{RIU}(\mathcal{O}_{CP})$ to $C_o(AL_{RIU})$ & $C_{ac}(AL_{RIU})$ (*third row*). Depictions in *(c)* and *(d)* represent the class-wise relative amount of points sampled in each iteration step, evaluated based on the GT labels.



(a) *(b)*

Figure 5.12: Overview of the achieved performance of our hybrid intelligence system for the H3D data set. In *(a)* we report the achieved labeling accuracy of the crowd (*first row*) as well as the accuracy of the machine learning from the crowd over all iteration steps (*second row*). The *dotted black line* represents the result of PL. Additionally, our congruence measures C_o and C_{ac} are evaluated for the runs relying on real crowd oracles $\mathcal{O}_{CM}$ to test the feasibility of our stopping criterion, i.e., $AL(\mathcal{O}_{CM})$ corresponds to $C_o(AL)$ & $C_{ac}(AL)$ and $AL_{RIU}(\mathcal{O}_{CM})$ to $C_o(AL_{RIU})$ & $C_{ac}(AL_{RIU})$ (*third row*). Depictions in *(b)* represent the class-wise relative amount of points sampled in each iteration step, evaluated based on the GT labels. Results regarding the crowd oracle provided with the mesh representation for labeling ($\mathcal{O}_{CM}$) will be revisited in Section 5.3.2.

Parametrization of the Hybrid Intelligence System

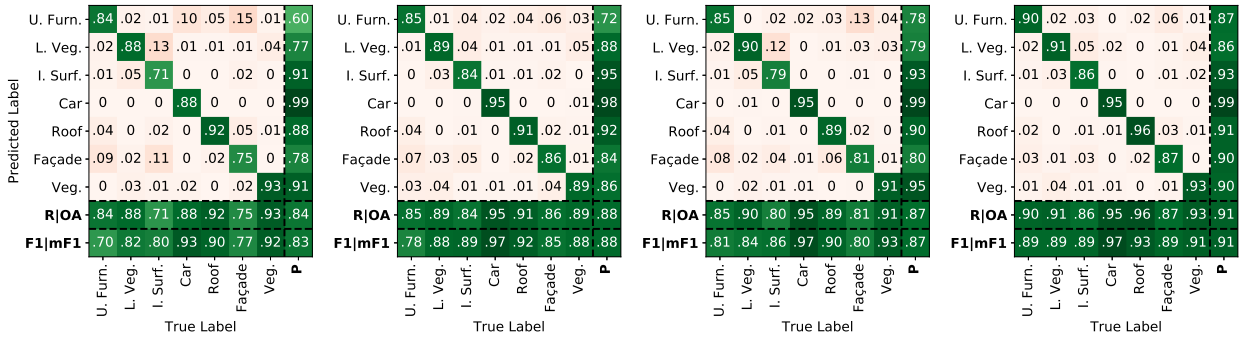
Based on these initial training sets, we launch respective AL loops for all our test sites applying the *weighted entropy* sampling strategy with *DiFS* sampling add-on and a batch size $n^+ = 300$. We employ an RF classifier parametrized by an ensemble of $n_e = 100$ Decision Trees with a maximum depth of $d^{max} = 18$ and a minimum number of samples at a node to justify a new split of $n^{min}_{samples} = 7$, following the findings in Section 5.1. In each case, we conduct a total of $n_i = 10$ iteration steps and allocate 10 payload points per crowd job (of *Type C*), which are posted to $n_{cw} = 3$ different crowdworkers for the purpose of majority voting at a payment rate of \$0.10 + \$0.05. Furthermore, the check point configuration identified to be optimal in Section 5.2.2 (3 check points, with one being shown twice) is in place. To compare the runs relying on a crowd oracle $\mathcal{O}_{CP}$ to the theoretically maximum achievable performance, in each case we also simulate corresponding AL loops by utilizing GT oracles $\mathcal{O}_O$ and contrast all runs to the respective PL solutions using the completely labeled training set. Regarding this, Figure 5.11 and 5.12 give an impression of the learning process of the hybrid intelligence system over all iteration steps. Table 5.8 summarizes the final accuracies achieved by the ML model for the semantic segmentation task.

Method	Oracle	F1-score [%]							[%]	
		U. Furn.	Car	L. Veg.	I. Surf.	Roof	Façade	Veg.	**mF1**	**OA**
V3D										
PL	-	-	65.64	82.25	91.28	94.81	62.30	86.24	80.42	88.11
AL	$\mathcal{O}_O$	-	66.00	79.35	90.70	**93.14**	**57.86**	**83.06**	**78.35**	**85.89**
	$\mathcal{O}_O$ + *RIU*	-	67.80	80.67	91.13	91.12	55.22	81.83	77.96	85.29
	$\mathcal{O}_{CP}$	-	66.05	80.11	90.71	91.25	49.87	82.91	76.82	85.03
	$\mathcal{O}_{CP}$ + *RIU*	-	**68.26**	**81.10**	**91.24**	88.76	45.39	82.14	76.15	84.19
H3D										
PL	-	41.14	51.63	90.68	85.26	92.96	83.77	93.05	76.92	88.16
AL	$\mathcal{O}_O$	33.93	56.34	**90.31**	82.70	88.33	79.73	**92.66**	74.86	**86.65**
	$\mathcal{O}_O$ + *RIU*	**36.97**	55.42	89.91	**83.84**	**90.05**	79.61	91.91	**75.39**	86.60
	$\mathcal{O}_{CP}$	32.32	53.95	89.92	76.26	88.88	75.43	91.70	72.64	83.38
	$\mathcal{O}_{CP}$ + *RIU*	31.00	53.04	88.54	79.69	86.82	76.43	90.67	72.31	83.55
	$\mathcal{O}_{CM}$	33.37	**57.40**	88.34	78.14	88.89	**79.83**	92.07	74.01	85.15
	$\mathcal{O}_{CM}$ + *RIU*	34.56	56.20	89.59	81.81	89.18	79.35	92.56	74.75	86.26
S3D										
PL	-	75.30		98.63		96.82		93.97	91.18	95.51
AL	$\mathcal{O}_O$	62.06		98.18		94.66		91.69	86.65	93.56
	$\mathcal{O}_O$ + *RIU*	**66.03**		**98.32**		95.22		92.40	87.99	**94.09**
	$\mathcal{O}_{CP}$	54.68		97.62		92.34		92.15	84.20	92.18
	$\mathcal{O}_{CP}$ + *RIU*	61.48		97.67		**95.53**		**92.84**	86.88	93.86

Table 5.8: Comparison of accuracies reached for the V3D, H3D and S3D data set for PL and various AL approaches after 10 iteration steps using different oracle types and sampling functions. Results concerned with the crowd oracle utilizing the mesh modality $\mathcal{O}_{CM}$ will be revisited in Section 5.3.2.

The Performance of the Crowd

With respect to the labeling accuracy of the crowd, we can observe that the OA values yielded in the respective iteration steps are significantly worse compared to those of the initial training set (i.e., iteration step 0). Please note that OA is used instead of mF1, as the latter would only poorly represent the true accuracy, since in individual iteration steps only very few samples for specific classes could be sampled, thus potentially severely affecting the mF1 measure (cf. Figure 5.11*(c)&(d)* & Figure 5.12*(b)*). But in contrast to the initial labeling step, where crowdworkers themselves select the points to be labeled, in the AL iteration steps they are determined by the machine. Since the ML model aims to resolve ambiguities between classes, it will tend to choose points it is currently most uncertain about near the decision borders, which often correspond to class borders as well (cf. Section 3.2). Thus, the reduced accuracy in AL iteration steps is due to dealing with points that are more complex for crowdworkers to label. Apart from this observation, all accuracy curves follow a decreasing trend. But at the same time, the more advanced the iteration, the better the performance of our model, as border cases of previous iteration steps can now be solved successfully. However, this also means that cases where the machine is uncertain become gradually more demanding, i.e., it focuses on more and more special edge cases which are difficult for interpretation not only for the machine but also for the crowd.



(a) $\mathcal{O}_{CP}$ *(b)* $\mathcal{O}_{CP}$ + *RIU* *(c)* $\mathcal{O}_{CM}$ *(d)* $\mathcal{O}_{CM}$ + *RIU*

Figure 5.13: Obtained confusion matrices over all iteration steps for different set-ups of the crowd oracle for the H3D data set. We experiment both with the provided data modality (point cloud or mesh in case of $\mathcal{O}_{CP}$ and $\mathcal{O}_{CM}$, respectively) as well as with the *RIU* technique (cf. Section 3.3.4).

As hoped for and already exemplified in Section 5.2.5, the *RIU* technique (with $d_{RIU} = 1.5\,\text{m}$) effectively helps to ease labeling for crowdworkers by focusing on points that are related to, but easier to interpret than the point originally selected by the machine. A more detailed analysis of the crowd performance both with and without support from *RIU* is given in Figure 5.13 for the H3D data set and Figure B10 in the Appendix for the V3D and S3D data set. Generally speaking, our expectations rooting from Section 5.2.5 have been proven correct, as *RIU* is indeed capable of improving the OA of a training set generated within the AL iteration by about 4 pp, which is caused by presenting points to the crowd that are easier to label. This technique allows to resolve label ambiguities between adjacent classes, such as *Low Vegetation* vs. *Impervious Surface*, *Roof* vs. *Façade*, *Impervious Surface* vs. *Façade* in case of the H3D data set, as can be seen from Figure 5.13 (a similar pattern is apparent with regard to the V3D data set, cf. Figure B10 in the Appendix). Although for H3D the rate of confusion of class *Urban Furniture* with the other classes can be reduced, the highest frequency of confusion can still be observed with respect to this class due to the high intra-class variability. This issue cannot be solved by simply increasing the distance to the class border when the class affiliation of the object under consideration is questionable as a whole. An exception in this regard is the S3D data set, where the remaining classes are very distinctly defined in this concise class catalog (cf. Figure B10 in the Appendix).

Apart from the performance of the human component of the system, also the corresponding time effort should be considered, especially as it ultimately determines the required time of a complete run of the hybrid intelligence system by means of the CATEGORISE framework. This is due to a negligible processing time of the machine part, i.e., the classifier, provided that we rely on the RF model (cf. Section 5.1.2). The time required for the labeling step of a crowd-driven training cycle/iteration step of our system can be specified as less than 11 hours. Thus, a whole AL run including initialization is completed in about 5 days (10 iteration steps · approx. 11 h + approx. 16 h for initialization = 126 h).

The Performance of the Machine

Considering the machine part of the intelligence system, i.e., the ML model, its performance is depicted in Figure 5.11*(a)&(b)* and Figure 5.12*(a)* for our different data sets. As for the runs with a simulated omniscient oracle $\mathcal{O}_O$ (as baseline solution), AL runs of all data sets show the typical convergence behavior and approximate the PL baseline solution. When adding the *RIU* technique (i.e., $AL_{RIU}(\mathcal{O}_O)$), respective runs tend to perform even better. This is because increasing distance from the class boundary can correspond to selecting a more generalized batch, rather than only focusing on most informative and thus most demanding instances. Although we need to resolve exactly such occurrences, in early iteration steps performance can

often be boosted when relying on more typical samples for different classes. In addition, in the first and second iteration steps, the selection of points in $AL_{RIU}(\mathcal{O}_O)$ is slightly more related to an equal class distribution than in $AL(\mathcal{O}_O)$ (cf. Figure 5.11*(c)*), possibly being beneficial for the ML model. While mF1-scores of runs both with and without *RIU* differ only marginally (especially towards the end of the iteration), the gap increases when replacing the simulated oracle $\mathcal{O}_O$ with the real crowd oracle $\mathcal{O}_{CP}$ in case of the V3D and S3D data set, due to the higher error level of labeled data in the variants without *RIU*, i.e. $AL(\mathcal{O}_{CP})$ (cf. Figure 5.11*(a)&(b)*). In this respect, the *RIU* technique not only increases the labeling accuracy of the crowd, but also the performance of the RF classifier learning from the crowd, and is thus capable of positively impacting the course of the iteration.

For the H3D data set, this does not seem to be true, as *RIU* neither improves nor diminishes the results. This is striking, since we would expect a smaller error rate in labeling to be associated with an improvement of the ML model learning from it. Apparently, the number of correctly labeled points of class *Urban Furniture* is decisive for the performance of the classification model, which is 241 for $AL(\mathcal{O}_{CP})$ and 244 for $AL_{RIU}(\mathcal{O}_{CP})$, thus a very similar amount. The more successful baseline runs with omniscient oracles $\mathcal{O}_O$, however, are provided with 484 and 473 samples for $AL(\mathcal{O}_O)$ and $AL_{RIU}(\mathcal{O}_O)$, respectively. Since the AL framework is typically able to select the samples that improve its performance, which should come from class *Urban Furniture* in our case, but chooses other classes instead, this means that the model is rather sure about this class - or at least is more uncertain about other classes. This uncertainty in other classes is a consequence of confused crowdworkers that the machine learns from. Since this crowd oracle frequently confuses *Urban Furniture* and *Façade* (due to façade furniture) with other classes (cf. Figure 5.13), our classifier becomes uncertain even though these other classes are actually well defined and have a solid base of correctly labeled points, but this clear definition is blurred by the aforementioned confusions. Thus, the ML model falls back to sampling from those other classes. This selection behavior of the machine can be taken from Figure 5.12*(b)*. For $AL(\mathcal{O}_{CP})$ and $AL_{RIU}(\mathcal{O}_{CP})$, no significant number of *Urban Furniture* samples is queried from the fifth iteration step on, in contrast to the more successful runs with an omniscient oracle (i.e., $AL(\mathcal{O}_O)$ and $AL_{RIU}(\mathcal{O}_O)$). This means that such noisy confusion by the crowd, which occurs in highly complex and convoluted village scenes where crowdworkers are overwhelmed with the categorization task (and where querying an alternative point via *RIU* has little effect due to the complexity of the complete area), prevents *RIU* from reaching its full potential for the classifier, especially when we are dealing with a suboptimal point cloud data representation compared to a realistically textured mesh modality. Therefore, we will revisit these results in the next section, dedicated to replacing the point cloud with a textured 3D mesh.

Apart from clarifying this specific effect, from the selection of samples in each iteration step in Figure 5.11*(c)&(d)* and Figure 5.12*(b)*, especially from the corresponding loops with omniscient oracles $\mathcal{O}_O$, we learn that *RIU* can have a real effect on the course of the iteration. Initially, RF models of each run learn from the same initialization data set, so early selections differ only slightly, but deviate gradually more with the number of iteration steps due to the altered sample selection. Furthermore, an overall trend of sampling batches of points more related to an equal class distribution becomes obvious for the V3D and H3D data set. Please note that this is not necessarily a desired property. For instance, if the model is generally confused about a specific class with high intra-class variability, it would be desirable to focus sampling on only this specific class, where selected samples should cover the whole bandwidth of this class in feature space, i.e., samples should be drawn from different *DiFS* clusters (cf. Section 3.3.3). This is also the reason why samples in batches of the large-scale S3D data set are not approximately equally distributed (at least not in the first 10 iteration steps), as we are confronted with a plethora of different representatives for each of the rather generalized classes in a data set spanning such a vast area. In the long run, however, it is for sure beneficial to provide the classifier with a training data set that evenly covers all classes, as our *weighted* sampling strategies intend to accomplish (cf. Section 3.3.2).

However, the question remains how these crowd-powered runs fare in comparison to our baseline solutions of PL, which are depicted in Figure 5.11*(a)&(b)* & Figure 5.12*(a)*, respectively, and are also contrasted numerically to our AL runs after $n_i = 10$ iteration steps in Table 5.8. With respect to achievable OAs,

the method we advocate for, AL with the *RIU* technique, relying on a real crowd oracle ($AL_{RIU}(\mathcal{O}_{CP})$) completely excluding an expert annotator, is only 3.92 pp (V3D), 4.61 pp (H3D) and 1.65 pp (S3D) below the performance of the respective PL baselines. As discussed in Section 5.1.4, classes with great intra-class variety such as *Urban Furniture* and *Façade* (including façade furniture) suffer the most from focusing only on a small AL training data set, but at the same time underrepresented classes (such as class *Car*) profit from AL sampling. It is worth emphasizing that these results are achieved by only labeling a small fraction of available training points of 4.7 ‰ (V3D), 0.1 ‰ (H3D) and 0.1 ‰ (S3D) (the absolute amount of labeled training points for each data set is 300 points · 10 it. steps + initialization points). Thus, these results come at a labeling cost of \$190 (100 jobs · \$0.10 + 100 jobs · 3 rep. · \$0.15 + [n^+/10 pts per job] · 3 rep. · 10 it. steps · \$0.15) plus a 10 % *MW* fee.

Terminating the Loop

While the accuracy values discussed before refer to the final classification performance after a fixed number of $n_i = 10$ iteration steps, we anticipate that a similar level of accuracy can be achieved with less iteration steps and thus less label effort. In other words, we aim to achieve a model that performs well with a minimum number of required iteration steps, i.e., for efficiency reasons, an accuracy curve reaching a stable high level fast is preferred to one performing slightly better after a larger number of iteration steps (assuming the same batch size n^+). We strive to identify the state of iteration where more label effort would only marginally improve model performance by means of our congruence values (cf. Section 3.5), evaluated for our crowd-based runs. We compute both the overall congruence C_o and the class-wise congruence C_{ac} between the current and the previous iteration step, i.e., $d_{stop} = 1$ (cf. Section 3.5). We succeed in describing the progress of the iteration if we are capable of computing curves that behave as similar as possible to the accuracy graphs, but without relying on GT data not available in real-world applications.

In case of V3D (cf. Figure 5.11*(a)*), this seems to be true. After a congruence value of 0 in the first iteration step (due to the lack of a model prediction from a previous step), congruence values correspond well to accuracy curves. Whenever there is an upward trend in accuracy, the congruence values also increase (e.g., consider congruence measures of $AL(\mathcal{O}_{CP})$; *dark blue* and *violet* curve in Figure 5.11*(a)*). Vice versa, flattening of accuracy curves also corresponds to flattening of congruence graphs towards the end of the iteration. In this regard $AL(\mathcal{O}_{CP})$ and $AL_{RIU}(\mathcal{O}_{CP})$ are suitable examples. While $AL(\mathcal{O}_{CP})$ shows a linearly increasing behavior, the latter flattens from the third iteration step on. In the congruence curves, we observe the same effects as well but with an inherent delay of one iteration step due to the required comparison with the prediction of the previous step. When the standard deviation of C_{ac} congruence values is computed, e.g., over the last $n_{stop} = 5$ iteration steps, we achieve a value of 1.4 % for $AL(\mathcal{O}_{CP})$ vs. 0.5 % for $AL_{RIU}(\mathcal{O}_{CP})$ at the tenth iteration step. Thus, depending on the stopping threshold, and as desired, $AL_{RIU}(\mathcal{O}_{CP})$ would be terminated earlier than $AL(\mathcal{O}_{CP})$. But of course, the termination of the loop also depends on the number of iteration steps n_{stop} which support the calculation of the standard deviation. This value was chosen gently, i.e., in such a way that stopping too early is avoided, to ensure that a stable plateau has been identified.

For the H3D data set, again congruence curves of $AL(\mathcal{O}_{CM})$ and $AL_{RIU}(\mathcal{O}_{CM})$ follow the steady linear increase of mF1 curves and resemble congruence values of $AL(\mathcal{O}_{CP})$ in V3D (please note that congruence curves are computed for an $\mathcal{O}_{CM}$ oracle, but which is not relevant with respect to congruence values; cf. Section 5.3.2 for a discussion of the impact of $\mathcal{O}_{CM}$ in an AL loop). But in contrast to V3D, a clear drop in congruence is observable for H3D at the second iteration step. This is not necessarily a bad omen for the progress of the iteration, as it only indicates a significant change in predicted labels at an unstable stage of the training. This is due to adding new samples that have significantly altered the current belief about decision borders. Figure 5.12*(b)* underlines this hypothesis. In the first iteration step, mainly samples of class *Façade* are queried, which supposedly leads to an improvement in the recognition of representatives of this class and probably also implicitly improves accuracies for adjacent classes, which might now be better distinguishable. However, in the second iteration step, for the first time, a great variety of classes within the batch are sampled. Therefore, the RF model is able to improve its overall classification capabilities with

respect to a greater bandwidth of classes, thus predicting significantly different. Remember that this effect becomes visible in congruence curves with a delay of one iteration step, explaining the strong discrepancy between the two successive predictions.

A similar behavior showing the fidelity of the congruence curves with respect to the accuracy curves (and thus suitability for defining a stopping criterion) can be spotted for the S3D data set (cf. Figure 5.11*(b)*). Please note that the characteristic up-and-down bending of the $AL(\mathcal{O}_{CP})$ mF1 curve between iteration steps 2 and 4 is also translated to the corresponding $C_{ac}(AL)$ curve (again with a delay of one iteration step), where the first upward trend is triggered by the inclusion of a considerable number of high intra-class variability *Building* samples in the third iteration step (cf. Figure 5.11*(d)*).

5.3.2 Evaluating the Effect of Different Data Modalities in AL Loops

Motivated by the already observed improvement in crowd labeling accuracy when utilizing a different data modality (cf. Section 5.2.3), we now expand the experiments from the previous section by incorporating the meshed data representation in an actual AL loop. We hope to observe i) an improvement of the labeling accuracy for the categorization of machine-selected points within the AL loop (in contrast to randomly chosen samples as in Section 5.2.3), and ii) that this enhanced labeling accuracy has a positive effect on the course of the AL loop and the achieved classification performance of our RF model. Due to the lack of open data sets including both concurrently acquired and labeled point cloud and mesh data, we restrict our experiments to the H3D data set. Precisely, we repeat respective experiments of the previous section, i.e., *weighted entropy*-based runs with *DiFS* sampling add-on and the same initialization data sets. But now points identified by the machine are presented to crowdworkers by visualizing the cylindrical neighborhood by means of a subset from the 3D textured mesh instead of the colorized point cloud (cf. Section 3.6.4).

To verify that the meshed data modality again helps to improve the achievable labeling accuracy of the crowd, we revisit Figure 5.13, depicting the respective confusion matrices over the complete AL runs. When comparing the achieved OA values, we can observe that in the mesh-based runs the crowd performs about 3 pp better compared to their point cloud-based counterparts. In a direct comparison between the crowd oracle being provided with the point cloud $\mathcal{O}_{CP}$ and the one presented with the mesh $\mathcal{O}_{CM}$, mainly confusion of unsystematic nature can be reduced, e.g., confusion of other classes with *Urban Furniture* due to the much sharper mesh depiction of the scene by the continuous surface representation, textured with more familiar imagery data. Thus, the reduction in confusion for this class can also be understood in such a manner that now crowdworkers can more confidently assign labels and do not have to resort to misusing *Urban Furniture* (i.e., *Other*) as class "*I don't know*" in resignation, also advocating for the use of textured meshes whenever available.

However, the more systematic confusion of *Low Vegetation* vs. *Impervious Surface*, *Roof* vs. *Façade* and *Impervious Surface* vs. *Façade* triggered by the machine selection of points situated near the decision borders, and thus often near the class borders (cf. Section 3.2), cannot be resolved. But exactly such confusion can be reduced with the help of the *RIU* technique by effectively increasing the distance of the selected point to the class border in object space (cf. Section 3.3.4). Thus, the optimal configuration, i.e., providing the crowd with a textured 3D mesh and easing labeling by means of *RIU*, leads to an improvement in OA of about 7 pp compared to the baseline solution of employing the point cloud without *RIU*. The clear advantage of this mesh modality for visualizing 3D content is also evident from Figure 5.12*(a)*. Oracles provided with the mesh achieve higher accuracy values over all iteration steps compared to their point cloud-based counterparts, with $\mathcal{O}_{CM}$+*RIU* showing an almost constant level of accuracy at about 90 % over all iteration steps, regardless whether typical examples as in early iteration steps or more special cases as in the later course of the iteration are queried for labeling. Thus, together with the findings in Section 5.2.3, we argue for the employment of 3D textured meshes whenever working with crowdworkers or, more generally, with geospatial non-experts.

In terms of the ML model performance, usage of the mesh modality leads to a significant improvement of mF1-scores from the fifth iteration step on (cf. Figure 5.12*(a)*). Recall that this is the point from which sampling of *Urban Furniture* instances was basically stopped in runs relying on a point cloud representation ($\mathcal{O}_{CP}$), due to the high level of uncertainty of the other classes resulting from the frequent usage of label *Urban Furniture* for representatives of all classes (cf. Section 5.3.1). The AL runs based on a mesh-using crowd (i.e., $AL(\mathcal{O}_{CM})$ and $AL_{RIU}(\mathcal{O}_{CM})$) ensure querying a sufficient number of representatives of *Urban Furniture*, as the other classes are already properly sampled and the classifier is sufficiently confident when predicting these classes, due to less frequent mislabeling of respective training samples as *Urban Furniture* by the crowd now (cf. Figure 5.13*(d)*). Interestingly, the oracle refraining from *RIU*, i.e. $\mathcal{O}_{CM}$, performs similar to $\mathcal{O}_{CP}$+*RIU* (cf. Figure 5.13*(b)&(c)*), while yielding a superior classification accuracy for the RF learning from those labels (cf. Table 5.8). This is caused by a more desired error behavior, as apart from the systematic confusion present for $\mathcal{O}_{CM}$ (that could be solved by *RIU*), main confusion occurs between *Urban Furniture* and *Façade* or, more precisely, façade furniture (cf. Figure 5.13*(c)*). In other words, these two classes contaminate each other, but not the rest of the classes, so that the AL algorithm can focus on resolving ambiguities related to only those two classes without exploring new concepts for a multitude of classes. Thus, especially for complex scenes such as in the H3D data set, not only the crowd profits from an easier understanding by means of the mesh data modality, but also the machine, learning from the crowd. This becomes obvious also from the performance of our RF classifier presented in Table 5.8. AL runs providing the crowd with 3D textured meshes (i.e., $\mathcal{O}_{CM}$) perform significantly better than their point cloud-driven counterparts, e.g., by 2.7 pp in OA in case of $AL_{RIU}(\mathcal{O}_{CM})$ compared to the corresponding point cloud-based run, even reducing the gap to the PL baseline to a margin of less than 2 pp.

5.3.3 Assessing the Long-Term Flexibility of ATL

In the previous two sections (Section 5.3.1 & 5.3.2), we have demonstrated that an arbitrary point cloud can be semantically enriched by means of our crowd-based AL loops with minimum labeling effort and hence cost. Thus, by design, our methodology offers a rather high level of flexibility and can be set up quickly for a new data set. However, to even further minimize costs, we strive to reuse information that is already available. In other words, training data from previous runs should be reused whenever it carries useful information for the current data set at hand in order to reduce the number of labels that need to be set anew. Since such approaches can only be considered effective if they actually lead to such a reduction of training data, we first simulate the expected behavior of DA with an artificial crowd oracle that is modeled based on our observations from the previous sections and that will turn out to be a valid approximation before verifying results with a real crowd oracle.

The presented ATL methodology (cf. Section 3.4) will be exemplified for multi-temporal point cloud analysis on the example of different H3D epochs (cf. Section 4.2.2). But we are not only confronted with possible domain gaps resulting from temporal differences, but also from the coverage of different areas (precisely, H3D epoch March 2018 covers only half the area of the other epochs). Within this experimental section, we consider the following DA scenarios:

- **Scenario I:** DA of high-resolution epoch March 2018 to also high-resolution epoch November 2018, i.e., DA is to be employed to transfer our classifier to an extended area, thus having similar geometric properties, but deviating acquisition time-dependent radiometric characteristics due to phenological and atmospheric changes.

- **Scenario II:** DA of high-resolution epoch March 2018 to significantly lower-resolution epoch March 2016, the latter colored by RGB values from epoch November 2018 (epoch March 2018 could be used as well, but would simplify the DA problem to a non-realistic level), i.e., DA between two data sets having deviating geometric and deviating radiometric properties, where point clouds stem from completely different sensors (cf. Section 4.2.2).

- **Scenario III:** Same as Scenario II, but colorization of epoch March 2016 by relying on the RGB tuples as provided in the H3D benchmark, which have been derived from an orthogonal projection of a temporally disjoint orthophoto onto the point cloud. Thus, Scenario III is designed to assess the impact of different colorization methods on DA.

Building Baseline Solutions

To illustrate the problem of lack of generalizability that we are typically confronted with when a pre-trained classifier is to be employed for deriving predictions on a new domain, we contrast a conventional PL result of the target domain D_T of each scenario with a so-called Passive Transfer Learning (PTL) result. For the latter, a model derived from the training data of the source domain D_S is utilized directly for inference on the test data set of the target domain D_T. Please note that in the experiments presented here, we make use of a reduced class catalog as found to be feasible in Section 5.3.1. While for the more related point cloud data sets of Scenario I and Scenario II accuracy suffers an mF1/OA loss of 8.27/4.46 pp and 12.30/5.17 pp (cf. Figure 5.14), we even record a decrease in accuracy of 26.59/20.13 pp for Scenario III, for which we expected the largest domain gap. These results can also be considered as baseline solutions for ATL. As a minimum requirement, we should be capable of exceeding the result of PTL where no measures were utilized at all to overcome the domain gap. An optimal result, on the other hand, would be achieving an accuracy similar to that of PL on the target domain D_T. However, we expect this goal to be hard to reach since, in contrast to our previous experiments, we are now confronted with two challenges at the same time: Resolving a possible domain gap while minimizing the number of necessary labels to only a fraction of what would have been required in a conventional PL approach.

As a second type of baseline for ATL, we also evaluate the effectiveness of conventional AL loops launched directly on D_T, which already significantly minimize labeling effort compared to PL. Therefore, for the two data sets that have not been involved in our previous experiments, namely H3D epoch November 2018 and epoch March 2016, we perform all necessary steps to derive initialization data sets for the AL loops, i.e., as before, in a first campaign a group of crowdworkers marks one arbitrary point for each class, and in a second campaign the crowdworkers check those labels (cf. Section 3.6.2). As expected, the accuracy of annotations generated for the more realistic scene depiction in epoch November 2018 exceeds that of epoch March 2016, which apart from being the sparser point cloud, also suffers from the suboptimal colorization technique. Nevertheless, for both data sets, training pools with an accuracy close to that of the initialization training sets in our previous experiments (cf. Section 5.3.1) of about 90 % in OA can be reached (a more detailed evaluation of the initialization process can be found in the Appendix in Figure B11 & B12).

Based on these initialization data sets, we can run respective AL runs on D_T for all our scenarios (we use the same initialization set for both Scenario II and Scenario III). Those AL loops are parametrized just like in the previous section, i.e., we employ RF classifiers having an ensemble of $n_e = 100$ Decision Trees with a maximum depth of $d^{max} = 18$ and a minimum number of samples at a node to justify a new split of $n^{min}_{samples} = 7$ with features listed in Table 3.1, computed for radii $r \in \{0.125, 0.25, 0.5, 0.75, 1, 2, 3, 5\}$ m for Scenario I and $r \in \{1, 2, 3, 5\}$ m in Scenario II and Scenario III due to the sparsity of H3D epoch March 2016. These classifiers are then incorporated in AL loops with *weighted entropy* sampling in batch-mode AL with a batch size of $n^+ = 300$ and *DiFS* add-on. For powering such loops, we utilize both an omniscient oracle $\mathcal{O}_O$ as well as a simulated noisy crowd oracle $\mathcal{O}_N$ for which we assume an error rate of $\sigma = 10\,\%$. No systematic errors were modeled for our simulated oracle since we allow sampling points by means of *RIU* with $d_{RIU} = 0.75$ m that are within this margin from the seed point situated on or near class borders. Considering the resolution of the H3D epochs, we assume this to be a sufficient simplification (cf. Section 5.1.3).

For all our target domains in Scenario I - III depicted in Figure 5.14, those AL runs approximate the PL result well, as expected, and are characterized by a rather steep gain in performance in early iteration steps (up to about iteration step 5). Please note that AL runs with the simulated crowd oracles ($AL_{RIU}(\mathcal{O}_N)$) tend to perform at least as well in OA as the omniscient oracle counterparts ($AL(\mathcal{O}_O)$), again underlining

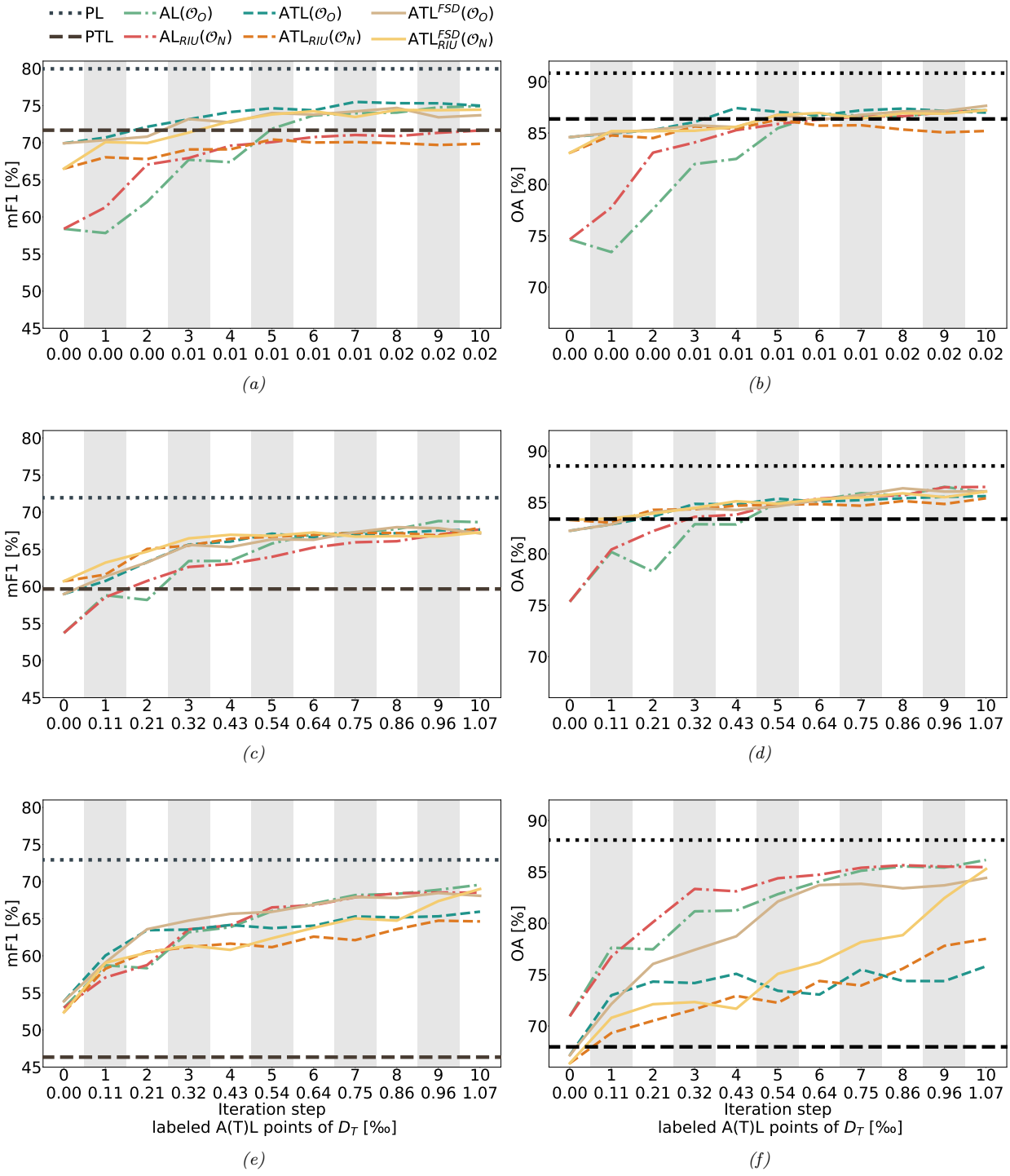


Figure 5.14: Compilation of the classification performance of the simulated runs for Scenario I - III (*top to bottom*), evaluated in terms of mF1-score (*left column*) and OA (*right column*).

RIU's enhanced generalization capabilities (cf. Section 5.3.1), which are especially helpful in early iteration steps (cf. Figure 5.14). In case of the $AL_{RIU}(\mathcal{O}_N)$ run for epoch November 2018 as baseline for Scenario I, the slightly decreased mF1-score (cf. Figure 5.14*(a)*) can be traced back to an increased confusion between classes *Urban Furniture* and *Car*, where noisy labels are especially harmful due to the similarity of features (e.g., trash bins or containers vs. cars). This effect occurs only now, as H3D's epoch November 2018 covers a larger area than epoch March 2018, and thus suffers from an even larger intra-class variability of class *Urban Furniture*.

Simulating ATL

Although these results can be considered as further demonstrating the effectiveness of the proposed hybrid intelligence system, we are now more interested in leveraging existing knowledge by applying ATL. For this purpose, for each of our scenarios, not only conventional AL but also ATL runs are computed being parametrized just like their AL counterparts. However, as initialization data sets, the resulting training set of a complete AL loop from the source domain (H3D epoch March 2018) with $n_i = 10$ iteration steps is used, based on one hand on a simulated crowd oracle $\mathcal{O}_N$ with *RIU* sampling and on the other hand on an omniscient oracle $\mathcal{O}_O$ (also generated by means of *weighted entropy+DiFS*). We also perform enhanced loops by removing the samples most inconsistent with the target data set to ease adaption to the new domain, referred to as ATL runs with *filtered source domain* ATL^{FSD} in Figure 5.14. In this regard, q is set to 1.1, meaning that in each iteration step, sampling of 300 points from D_T is paired with eliminating of 330 most disagreeing samples from D_S (cf. Section 3.4).

With respect to Scenario I, as hoped, all ATL runs initialize on a much higher level of accuracy (about 10 pp both in mF1 and OA; cf. Figure 5.14*(a)&(b)*). When comparing ATL runs where removal of contradictive samples is allowed (ATL^{FSD}) with those where it is not (ATL), for omniscient oracles $\mathcal{O}_O$ this enhanced method seems unnecessary and even slightly diminishes accuracy. However, when asking simulated noisy crowd oracles $\mathcal{O}_N$ for labels, the effectiveness of this strategy becomes obvious, as $ATL^{FSD}_{RIU}(\mathcal{O}_N)$ significantly outperforms $ATL_{RIU}(\mathcal{O}_N)$ which fares even worse than the $AL_{RIU}(\mathcal{O}_N)$ baseline. Again, the reason for this suboptimal behavior is confusion between classes *Urban Furniture* and *Car*, where the inherent overlap in feature space is even further amplified by mixing samples from two different domains. To summarize, a scenario where we capture the same (or an extended) scene with the same sensor set-up, but under significantly different atmospheric conditions (cf. Section 4.2.2) and thus radiometric properties, is still well suited for transferring previously learned knowledge.

Since our motivation for DA is mainly of economic nature, i.e., minimizing labeling and thus costs, we will also evaluate results in this regard. First of all, the costs for the generation of an initialization data set, which amount to $100\,\text{jobs} \cdot \$0.10 + 100\,\text{jobs} \cdot 3\,\text{rep.} \cdot \$0.15 = \$55$ for a conventional AL loop on the target domain D_T, can be completely avoided by ATL. With respect to further savings during the iteration, $ATL^{FSD}_{RIU}(\mathcal{O}_N)$ is about three iteration steps ahead of $AL_{RIU}(\mathcal{O}_N)$ (e.g., see the OA for $AL_{RIU}(\mathcal{O}_N)$ in iteration step 4 vs. $ATL^{FSD}_{RIU}(\mathcal{O}_N)$ in iteration step 1), providing the possibility to save the expenses incurred for these steps ($3\,\text{it.steps} \cdot 30\,\text{jobs} \cdot 3\,\text{rep.} \cdot \$0.15 = \$40.5$). Thus, the more limited the labeling budget, the more beneficial ATL becomes.

From the results of Scenario II, where we are confronted with two domains of rather different geometric properties, we can observe an overall trend rather similar to Scenario I, but due to an increased domain gap, the margin between AL and ATL gets consequently smaller, i.e., reusing knowledge from the source domain is less helpful for the target domain D_T. Nevertheless, transferring knowledge still helps to reduce costs of enriching the target domain with semantic information. $ATL^{FSD}_{RIU}(\mathcal{O}_N)$ reaches a close to final accuracy level already in the fourth iteration step, while $AL_{RIU}(\mathcal{O}_N)$ requires nine iteration steps to yield the same accuracy (cf. Figure 5.14*(c)*). Thus, from Scenario II we can learn that even significantly different geometries between (partly overlapping) domains do not hinder our classification model from successfully exploiting knowledge from a source domain. In other words, this means that our geometric features utilized by the RF,

especially those based on the structural tensor, generalize well although not explicitly designed to be domain invariant. Radiometric features (especially *Reflectance/Intensity* of laser returns and color features), on the other hand, are expected to be sensitive to domain changes, and are also to be examined.

Hence, for Scenario III, we now increase the domain gap even more by not only attempting a transfer to a different geometry, but also to a completely different and coarser radiometry obtained from projection of orthophoto colors onto the point cloud. This means that in the target domain (epoch March 2016) we are more confronted with a 2D colorization, in contrast to a true 3D colorization as in the source domain (epoch March 2018). In comparison to Scenario I and Scenario II, pure AL runs perform best in Scenario III, demonstrating that the domain gap is too severe for source domain samples having a positive impact on the classification of the target domain, making them more of a burden than a help. However, this is exactly the use case for which our measure for removing deprecated samples of D_S is designed. While this technique had only a limited positive impact on the performance of respective runs in Scenario I and Scenario II, ATL^{FSD} becomes indeed crucial in Scenario III to obtain a final accuracy close to that of the AL runs (cf. Figure 5.14*(e)&(f)*) in contrast to their pure ATL counterparts. Although ATL^{FSD} loops are not suited for minimizing the number of required iteration steps (as the AL accuracy is only reached in iteration step 10), they make the relatively expensive generation of an initialization data set for D_T obsolete, so that significant costs can still be saved.

We would like to emphasize that despite the differing severity of domain gaps in our scenarios, setting the deletion/inclusion ratio q to 1.1 successfully deals with these different gaps in all cases, underlining the robustness of this parameter. This means that when applying the ATL^{FSD} technique, an operator can include source domain samples either way, and it is guaranteed that the course of the iteration is only positively impacted in terms of required labeling costs. Even in Scenario III, we achieve the same level of accuracy as a conventional AL run, but without having to provide a dedicated initialization training set.

To shed light on this very interaction between deletion and inclusion as incorporated in ATL^{FSD} runs, we take Scenario III as an example. Interestingly, the learning progress of $ATL^{FSD}(\mathcal{O}_O)$ is significantly more desirable than that of $ATL^{FSD}_{RIU}(\mathcal{O}_N)$, since it follows a converging curve, in contrast to the course of $ATL^{FSD}_{RIU}(\mathcal{O}_N)$, making an early stopping impossible. Precisely, in $ATL^{FSD}_{RIU}(\mathcal{O}_N)$, instances of class *Low Vegetation* are frequently mispredicted as class *Impervious Surface* (e.g., 40 % in iteration step 5). This can be traced back when evaluating the class affiliations of both sampled points from D_T and deleted points from D_S throughout the iteration, as depicted in Figure 5.15. For $ATL^{FSD}(\mathcal{O}_O)$, 20 % of deleted points in iteration steps 2-4 belong to class *Impervious Surface*, indicating that this class is subject to a domain shift, which comes however striking as one would assume that this class would be well generalizable due to its clear, data invariant appearance. Indeed, the actual cause lies in the class *Façade*, which is only scarcely represented in H3D epoch March 2016, in contrast to the other epochs (cf. Figure 4.2*(c)* vs. *(a)*). Since in AL we always sample points that are most challenging for the ML model to interpret and that are typically situated on class borders (cf. Section 3.2), such as the border between classes *Impervious Surface* and *Façade*, exactly such points are sampled in the source domain epoch March 2018. However, in the target domain epoch March 2016, typically façade points are only scarcely located in the vicinity of points from class *Impervious Surface*. Therefore, the geometric feature description can be considered outdated, and $ATL^{FSD}(\mathcal{O}_O)$ has rightfully eliminated these samples. In contrast, $ATL^{FSD}_{RIU}(\mathcal{O}_N)$ focuses on removal of class *Urban Furniture* in early iteration steps as it is somehow distracted with this class when not only having to deal with this quasi-class *Other* but especially when label noise is present "smearing" the borders of this class even more.

In general, from the class affiliations of sampled and deleted points in Figure 5.15, we can observe that sampling in an iteration step often focuses on specific classes (especially for $ATL^{FSD}(\mathcal{O}_O)$, also explaining its faster convergence; cf. Figure 5.14*(e)&(f)*), as already discussed for AL runs in Section 5.3.1. When deleting points, on the other hand, the respective distributions in each iteration step are more like equal distributions, meaning that there is no specific class that has completely changed its appearance (in feature space). The fact that samples disagreeing with the target domain are spread among all classes is to be expected when transferring from a high-resolution data set (epoch March 2018) to a much coarser point cloud (epoch March

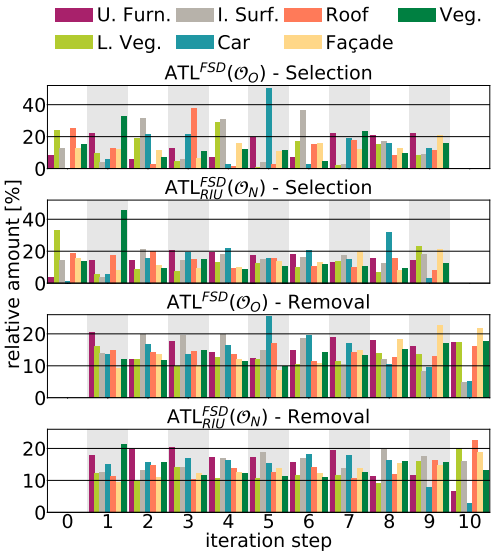


Figure 5.15: Comparison of class affiliations of sampled points of the target domain D_T with those removed from the source domain D_S for Scenario III (at iteration step 0, no points can be deleted since no points of D_T are present yet to derive disagreement rates, thus the offset by 1; cf. Section 3.4).

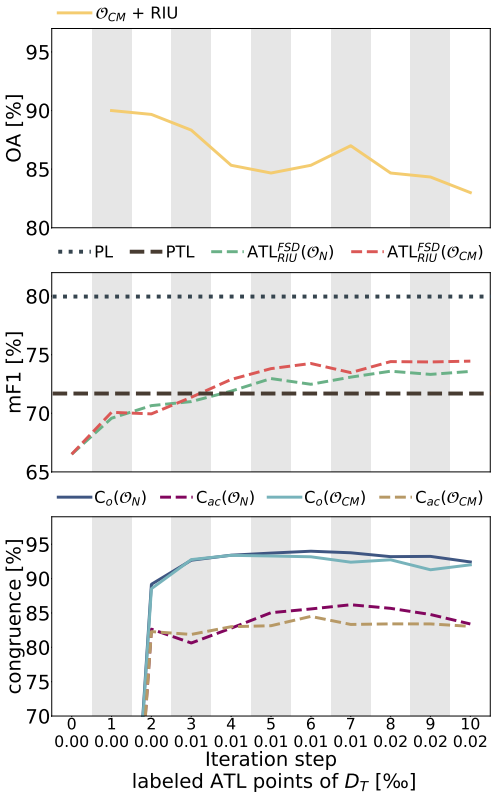


Figure 5.16: Overview of the performance of the hybrid intelligence system for the ATL task outlined in Scenario I. We contrast the labeling accuracy of the crowd (*top*) with the performance of the machine (*middle*) and evaluate congruence values for ATL runs relying on both a simulated noisy oracle $\mathcal{O}_N$ and a real crowd oracle $\mathcal{O}_{CM}$ required for automatic stopping of the ATL iteration (*bottom*).

2016). An interesting observation can be found in iteration step 5 for $ATL^{FSD}(\mathcal{O}_O)$, where the ML model is mainly concerned with updating its understanding of class *Car* in this coarser scenery, in the form of eliminating all disagreeing samples and simultaneously querying most informative instances of this class from the target domain.

ATL with a Real Crowd Oracle

Since our main goal is establishing a hybrid intelligence system, we aim to verify these findings, which are based on simulated crowd oracles, by means of a real crowd oracle $\mathcal{O}_{CM}$. In this context, we limit ourselves to repeating the ATL run that has the highest potential to really profit from including source domain samples due to the greatest relatedness between source domain (epoch March 2018) and target domain (epoch November 2018), i.e., Scenario I. Precisely, we reiterate ATL^{FSD}_{RIU} but replace the simulated oracle with a real one. We conduct this experiment by relying on the CATEGORISE framework so that an operator is only tasked with passively monitoring the progress (cf. Section 3.8).

Figure 5.16 displays the obtained results in terms of both the crowd's performance as well as the one of the machine. Again, the labeling accuracy of the crowd, expressed by means of OA, is characterized by a decreasing trend. This is to be expected again since the classifier selects more and more complex points in the course of the iteration, which are more demanding for a human labeler as the machine gradually focuses on the interpretability of fine-grained details. As identified to be beneficial in Section 5.3.2, the crowd is again equipped with subsets from the 3D textured mesh instead of the point cloud to increase interpretability. However, with a mean OA over all iteration steps of about 86 % (cf. Figure B13 in the Appendix), the achieved accuracy is about 5 pp behind that of the pure AL run for epoch March 2018 (cf. Figure 5.13), which seems surprising as we are dealing with data of similar visual quality. However, the reason for this is that the ATL iteration is more advanced than those started from scratch in Section 5.3.2. Precisely, in ATL loops, a mixture of both typical but so far unknown samples, but also more complex samples that refine already well-defined separation hypothesis is selected. The latter are, of course, far more complex for interpretation than those in a conventional AL iteration with a limited number of iteration steps. Or, in other words, our ATL run continues the iteration in the target domain, which has already run for 10 iteration steps on the source domain. Thus, the decreasing trend in crowd performance already observed in Figure 5.11 *(a)&(b)* and especially Figure 5.12*(a)* is continuing.

This increased level of complexity is also implicitly visible when analyzing the confusion matrix in terms of crowd labels over the entire ATL iteration (cf. Figure B13 in the Appendix), where we can observe an increased level of confusion of representatives from other classes with quasi-class "*I don't know*" *Urban Furniture*, but also from the generally increased confusion level for the ATL run. Nevertheless, a comparison of the achieved mF1-scores between the ATL run powered by a real crowd oracle ($ATL^{FSD}_{RIU}(\mathcal{O}_{CM})$) and the run with a simulated crowd oracle considering labeling errors ($ATL^{FSD}_{RIU}(\mathcal{O}_N)$) demonstrates that the simulated oracle is a valid approximation of the true one, since accuracy curves differ only insignificantly (the reported mF1 curve for $ATL^{FSD}_{RIU}(\mathcal{O}_N)$ in Figure 5.16 is the same as in Figure 5.14*(a)*).

Just like for the pure AL runs, we are also interested in an automatic stopping criterion to terminate our ATL runs. For this purpose, we again compute the overall congruence measure C_o as well as the class-sensitive congruence measure C_{ac} for ATL^{FSD}_{RIU} both with a simulated noisy oracle $\mathcal{O}_N$ and real crowd oracle $\mathcal{O}_{CM}$. However, since we expect high similarity between successive predictions due to the continuation of the AL loop already begun in the source domain, we increase the distance of iteration steps to compare against to $d_{stop} = 2$ to emphasize differences (cf. Section 3.5), so that a too early termination of the iteration becomes unlikely (thus, we cannot compute congruence values for iterations steps 0 and 1). From Figure 5.16, we can observe that the overall congruence measures for both runs ($C_o(\mathcal{O}_N)$ & $C_o(\mathcal{O}_{CM})$) saturate from the fourth iteration step, which is well in accordance with the behavior of the OA of $ATL^{FSD}_{RIU}(\mathcal{O}_N)$ in Figure 5.14*(b)*. However, as we are interested in a well performing classifier for all classes, we focus on the class-sensitive congruence values C_{ac}. Drawing from the mF1-scores reported in Figure 5.16 (*middle*), we would like to terminate the iteration after the fifth iteration step in order to maximize cost-effectiveness. But since, for the lack of GT data to compare predictions against, we are dealing with an intrinsic measure defining termination of a loop, we can only detect whether there are changes in successive predictions (cf. Section 3.5). While our congruence curves approximate respective accuracy curves quite well, we require more iteration steps than actually necessary from a performance point of view for (reliably) detecting convergence. To do so, in our case, we compute the standard deviation of congruence values over the last $n_{stop} = 3$ iteration steps, which is compared against a user-defined threshold t_{stop} (cf. Section 3.5) that should be chosen close to 0 (but of course depending on the particular application and desired quality level at hand). Thus, termination is likely to be met in iteration step 7, since C_{ac} congruence values of steps 5-7 are rather similar (both for $\mathcal{O}_N$ and $\mathcal{O}_{CM}$). This means that the iteration has to run for 2 additional, but actually superfluous iteration steps - but with the advantage of automatic termination without requiring labeled reference data for comparison.

Unlike the congruence curves for our conventional AL runs (cf. Figure 5.11*(a)&(b)* and Figure 5.12*(a)*), the ones in ATL show a decreasing trend towards the end of the iteration (cf. Figure 5.16). This is caused by removing too many samples from the source domain D_S as the iteration progresses. In Scenario I, most

samples of D_S are also decisive for D_T, so excessive removal of samples from D_S only forces the classifier to relearn these "lost" concepts by sampling respective points of the target domain D_T again. This causes the decision borders to alter again as, in principle, they shift from being supported only by source domain samples to being based on a mixture of source and target domain samples to a state where decision borders are defined solely by samples from the target domain. However, considering the congruence values in Figure 5.16, we argue that a stable state is reached before complete removal of samples from D_S at a moment profiting as much as possible from D_S, but where necessary adaptions to D_T have already been achieved.

Chapter 6

Conclusion, Limitations and Outlook

The motivation driving this thesis was to generate a framework that is capable of teaching a supervised ML system to automatically enrich an arbitrary point cloud with semantic information by only providing it with the respective cloud as well as access to a (fixed) labeling budget. This was achieved by setting up a hybrid intelligence system with an AL backbone in which both an ML model and humans work together, with the latter being considered as *human processing units* in reference to *electronic processing units* typically encountered in automated systems. When paid crowdworkers take this role, there are still humans working in the AL loop, but from an operator's perspective, we are dealing with a fully automated system. The work of crowdworkers can be considered a non-deterministic subroutine of our program for timely returning labels, but behaving just like other routines that are accomplished by *electronic processing units* (such as training our RF or SCN classifier). This concept was the design blueprint for the CATEGORISE framework, which was applied to a variety of three ALS point clouds representing spatially distinct areas (with two of them being state-of-the-art benchmark data sets) with different characteristics. For those, we achieved accuracies at a quality level similar to that of PL by annotating only a small fraction of potentially available training points (typically $\ll 1\,\%$), thus causing minimal monetary expenses.

The machine part of this system comprises both the classifier as well as the link to the crowd component, that is AL, with the latter being the focus of this thesis especially. We found that common sampling strategies proposed in literature can be significantly boosted with respective sampling add-ons, such as applying a dedicated weighting scheme as well as a measure to guarantee diversity of samples queried within a batch in an AL iteration step. While such strategies can be employed in tandem with classifiers from both the feature-driven as well as the data-driven domain, for our purpose, we identified a feature-driven RF model to be more suitable for the AL scenario than a data-driven SCN classifier. This is due to the inherent working principle of the two classifiers, with CNN-based approaches having to relearn or at least refine all features in each iteration step, while for the RF classifier they only have to be computed once and can be reused in any following iteration step, leading to faster processing at a similar level of accuracy compared to the SCN approach. As a remedy to boost the performance of CNN-based classifiers, future work could focus on easing the computational load by not only guiding sampling based on informativeness regardless the location, but encouraging points that are spatially clustered (in object space) so that spatially compact point cloud subsets can be provided to the DL system. Contrary to our current approach, where queried points may be scattered over the whole data set, thus requiring presentation of all data to the network, and in reference to the work of Lin et al. (2020a), we see this as means to enable CNN approaches to become computationally competitive.

Furthermore, we applied a DA-appropriate variant of AL, namely ATL, showing the potential to further minimize costs by leveraging information that is already known from another domain. Precisely, in our ATL

runs, samples from a source domain can be included in any case, and the algorithm detects which of these samples are actually helpful for classification of a target domain, leading to faster convergence of the ATL loop, and which should be discarded, respectively. In this way, we aim to guarantee that ATL loops do not require more iteration steps than conventional AL loops launched directly on the target domain. This is an important property as it is often hard to decide whether two point clouds captured from different sensors and/or depicting different scenes are actually related in a manner where reusing labels is appropriate.

With respect to the oracle responsible for generating labels for queried samples, we did not end our experiments with simulated AL runs assuming unrealistic omniscient oracles, but evaluated the impact of different error behaviors of the same, namely oracles generating labels with random errors and such with systematic errors following a dedicated but undesirable mapping function. We observed that while our classifiers are well capable of compensating random errors, they are extremely sensitive to systematic errors. Thus, we should assist our oracle in avoiding such errors whenever possible, leading to our other main component, i.e., the crowd for labeling 3D point clouds. Since such systematic errors often occur when points are sampled that are situated close to the decision border, with this decision-border-proximity typically being strongly correlated with distance to the class border in object space, we developed the *RIU* technique to increase the distance of sampled points to this very border. Although from the view of the machine this seems counterintuitive, as the machine is always interested in most informative points, it often even boosts convergence of AL loops in early iteration steps due to a better generalization and especially helps crowdworkers in comprehending sampled points.

Since the basic idea of the *RIU* concept is identifying a point that is easier to interpret for humans while still being sufficiently related to the queried seed point with respect to its representation in feature space, for an alternative approach with similar reasoning, we could also argue that such points can be found in the seed point's cluster formed by an unsupervised segmentation technique such as the one described by Vosselman et al. (2017). Provided optimal results of this pre-processing step, in our crowd tools we could also highlight the complete segment the queried point belongs to, instead of only one single point. This might help to further ease labeling, as crowdworkers would be tasked to set labels for complete objects or object parts (e.g., a full roof face instead of a single point close to the class border), while implicitly setting point-wise class labels. However, the effectiveness and accuracy would be determined by the performance of the algorithm for unsupervised segmentation, also raising the question of the robustness of its parametrization, so that it might come with the overhead of a human operator having to validate the result at least visually.

Another branch of future work could follow a promising approach from the weakly supervised domain that was recently adapted by Lin et al. (2022) for the automatic interpretation of ALS point clouds. In this concept, an ML model is enabled to derive point-wise predictions, although only learning from so-called scene-level weak labels for point cloud subsets indicating *that* there are representatives of a specific class included, but not *which* point(s) actually belong to this class. Thus, when our crowdworkers are required to only provide such scene-level labels, from our point of view, this offers the potential to drastically ease labeling complexity. Another recent approach that also facilitates labeling by avoiding to query single most informative points, which are also most complex for interpretation, is presented by Wang and Yao (2022). The method expects that annotators provide one point each for *all* classes that are included in a given subcloud (thus also implicitly generating scene-level labels). Incorporating such a labeling scheme into a respective hybrid intelligence system with crowd annotators would lead to crowd tasks similar to our *Type A* labeling tasks (cf. Section 3.6.2), but would also come with the inherent challenge of reliable quality control, since majority voting schemes are hard to implement when it is likely that each worker would select a different point set. In contrast to the work of Lin et al. (2022), where majority voting could be easily implemented for the scene-level label vector, this approach would for example require an additional *Type B*-like campaign for verification of crowd labels (cf. Section 3.6.2), thus being slightly more cumbersome.

In terms of the accuracy of our crowd oracle, relying directly on generated labels without any means of controlling the quality is a daring choice due to the well-known quality inhomogeneity in crowdsourced work. Thus, measures were implemented for both *quality control on task designing*, where check points have

proven to be most effective, and *quality improvement after data collection*. For the latter, we rely on the idea of the *Wisdom of the Crowds*, realized in a way that each point is labeled by multiple crowdworkers, with the resulting label being obtained from majority voting. But since such a majority voting scheme can substantially increase labeling costs, it might be reasonable to conduct a pre-study on the number of multiple acquisitions required for the specific task at hand, like the one discussed in this thesis or that recently presented by Walter et al. (2022b). Both aforementioned measures for quality control offer the important property of being easily automatized, which is a prerequisite for automatizing our hybrid intelligence system as a whole.

Apart from these techniques, we have observed that alternative data modalities can help to further boost the performance of the crowd (and consequently the one of the machine learning from the crowd), as visualizing a selected point to be labeled in a 3D textured mesh environment led to a significant gain in accuracy. This is not at least because some crowdworkers are probably familiar with this modality from online geospatial services. However, these results rely solely on one single data set as, to the best of our knowledge, apart from the newly introduced H3D benchmark, no other data set offers both point cloud and mesh data to conduct such a study. But with hybrid sensor forms being a current trend in geospatial data acquisition, and considering having observed a clear interpretation gain from using this modality, we have high hopes that textured mesh data will become the quasi-standard whenever working with geospatial lay people. As the raw point cloud is often more suited for processing (e.g., semantic segmentation) and the mesh has clear advantages for visualization, it is also desirable to be able to transfer information between the two modalities, especially to enrich mesh data with semantic information, which can be accomplished simultaneously with the point cloud in an extended point cloud-based AL loop, as discussed by Kölle et al. (2021b).

While 3D data forms and especially meshes lead to the truest possible representation of geospatial data, it still comes with the risk that individual crowdworkers will be overwhelmed with 3D data at all, thus completely failing in interpretation. In this respect, and again encouraged by the nowadays often conducted joint acquisition of imagery and LiDAR data from the same platform, as alternate approach, we could project AL-sampled 3D points into the image data and provide this information to crowdworkers when requesting labels. The challenge of such a technique, on the other hand, is that not every LiDAR point is necessarily depicted occlusion-free in the imagery, and even if so, the question remains of which image (or which set of images) containing this point is best suited for crowdworkers. However, turning from displaying 3D data and embracing 2D representations instead would also foster a transfer of our labeling tools to a state where jobs can be successfully completed with mobile devices, so that not only dedicated "full-time" workers but also occasional workers could contribute.

Apart of trying to tailor the data to the needs of the crowdworkers, we have also focused on increasing the motivation of our workers by bringing them joy through gamification elements. While we were able to trick our workers into contributing more labels, this effect did not last for more than one labeling round (or high score), and the relatively simple game elements lost their appeal, as stressed to be a major challenge by Thaler et al. (2012). Since it would require a far more complex game to trigger a long-term motivation similar to the *ESP* game (von Ahn and Dabbish, 2004) or the *Peekaboom* game (von Ahn et al., 2006b), a remedy might be to add a pool of different game elements from which individual ones are randomly drawn in every playthrough for keeping up the excitement. But despite our game elements, unfortunately, we were not able to also trick the crowd to perform better. However, considering that our workers have voluntarily contributed more, they seem to have actually enjoyed gamified labeling, which raises the question of whether they are even capable of producing better results than those obtained in this thesis.

This goes hand in hand with our observation that the crowd is unable to annotate points according to an arbitrarily complex and fine-grained class catalog (also identified by Bayas et al., 2016), why we actually had to simplify V3D's and H3D's class catalog. According to our findings, the crowd performs well for classes that are easy and straightforward to understand from the pure class names, but fails whenever classes are non-intuitive or are rather concerned with a detailed and maybe even subjective description (e.g., *Shrub*

vs. *Hedge*). Furthermore, unspecific classes such as *Urban Furniture* are often misused as quasi-class *Other* whenever crowdworkers are unsure about the class affiliation of a specific point. Thus, it remains an open question *if* and *how* complex class catalogs could be taught to the crowd. In this regard, we recommend consulting experts from the field of work ergonomics and work psychology, which might also help to design labeling tools that are more suited for crowdworkers than those we have implemented according to our liking and experience with crowdworkers in an iterative engineering process.

Another limitation of all crowd-based results in this thesis is that experiments were conducted with a certain group of workers who were active users of the *MW* platform at the time of posting the respective campaign. While some workers in our studies contributed to many campaigns over the years, the vast majority of workers participated solely in one campaign. This underlines a certain level of robustness of our approach, but at the same time this observation also means that we cannot guarantee that the results reported in this thesis are identically reproducible. They may actually vary within certain bounds when being repeated (Section 5.2.4 illustrates the variety that is to be expected), although we are ensuring a minimum level of quality by means of our mechanisms for automated quality control. This variation of results might also be amplified when switching to another crowdsourcing platform such as *MTurk*. However, such an outcome uncertainty is an inherent issue for any human-centered research. It would be interesting to see whether the geospatial understanding of lay crowdworkers improves over the years due to the wider dissemination of online geospatial services. Thus, a branch of future research focusing on long-term experiments (in the range of years) could also provide new insights for crowdsourced geospatial data collection.

Despite these limitations, we have succeeded in formulating a fully automated hybrid intelligence system based on AL with subprocesses being carried out by inherently non-deterministic *human processing units* (i.e., crowdworkers) and that does not involve an expert in any labeling task. By conducting this work, we hope to have paved the way for a wider acceptance and dissemination of hybrid intelligence systems relying on (paid) crowdsourcing in the field of geospatial data analysis.

Bibliography

Actueel Hoogtebestand Nederland (2021). Dataset: Actueel Hoogtebestand Nederland (AHN3) [WWW Document]. URL: https://www.pdok.nl/introductie/-/article/actueel-hoogtebestand-nederland-ahn3- (accessed February 2, 2021).

Allahbakhsh, M., Benatallah, B., Ignjatovic, A., Motahari-Nezhad, H. R., Bertino, E., and Dustdar, S. (2013). Quality control in crowdsourcing systems: Issues and directions. *IEEE Internet Computing*, 17(2):76–81.

Angluin, D. (1988). Queries and concept learning. *Machine Learning*, 2(4):319–342.

Anguelov, D., Taskar, B., Chatalbashev, V., Koller, D., Gupta, D., Heitz, G., and Ng, A. (2005). Discriminative learning of markov random fields for segmentation of 3D scan data. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2:169–176.

Antoniou, V. and Skopeliti, A. (2015). Measures and indicators of VGI quality: An overview. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-3/W5:345–351.

Argamon-Engelson, S. and Dagan, I. (1999). Committee-Based Sample Selection For Probabilistic Classifiers. *Journal of Artificial Intelligence Research*, 11:335–360.

Ash, J. T., Zhang, C., Krishnamurthy, A., Langford, J., and Agarwal, A. (2019). Deep batch active learning by diverse, uncertain gradient lower bounds. *CoRR*, abs/1906.03671.

Atlas, L., Cohn, D., and Ladner, R. (1989). Training connectionist networks with queries and selective sampling. In Touretzky, D., editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann.

Basiri, A., Haklay, M., Foody, G., and Mooney, P. (2019). Crowdsourced geospatial data quality: challenges and future directions. *International Journal of Geographical Information Science*, 33(8):1588–1593.

Bayas, J. L., See, L., Fritz, S., Sturn, T., Perger, C., Dürauer, M., Karner, M., Moorthy, I., Schepaschenko, D., Domian, D., and McCallum, I. (2016). Crowdsourcing in-situ data on land cover and land use using gamification and mobile technology. *Remote Sensing*, 8(11):905.

Becker, C., Häni, N., Rosinskaya, E., d'Angelo, E., and Strecha, C. (2017). Classification of Aerial Photogrammetric 3D Point Clouds. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-1/W1:3–10.

Bell, S., Upchurch, P., Snavely, N., and Bala, K. (2013). OpenSurfaces. *ACM Transactions on Graphics*, 32(4):1–17.

Bell, S., Upchurch, P., Snavely, N., and Bala, K. (2015). Material recognition in the wild with the materials in context database. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Belton, D. and Lichti, D. (2006). Classification and segmentation of terrestrial laser scanner point clouds using local variance information. *IAPRS*, 36:44–49.

Beluch, W. H., Genewein, T., Nurnberger, A., and Kohler, J. M. (2018). The power of ensembles for active learning in image classification. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517.

Bernardo, I. (2022). Random Forests Walkthrough — Wisdom of the Crowds and Why They Are Better than Decision Trees [WWW Document]. URL: https://towardsdatascience.com/random-forests-walkthrough-why-are-they-better-than-decision-trees-22e02a28c6bd (accessed August 26, 2022).

Bezos, J. (2007). Artificial Intelligence, With Help From the Humans [WWW Document]. URL:

https://www.nytimes.com/2007/03/25/business/yourmoney/25Stream.html (accessed February 18, 2022).

Blomley, R., Jutzi, B., and Weinmann, M. (2016). Classification of airborne laser scanning data using geometric multi-scale features and different neighbourhood types. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, III-3:169–176.

Blomley, R., Weinmann, M., Leitloff, J., and Jutzi, B. (2014). Shape distribution features for point cloud analysis - a geometric histogram approach on multiple scales. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, II.

Bloodgood, M. and Vijay-Shanker, K. (2009). A method for stopping active learning based on stabilizing predictions and the need for user-adjustable stopping. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 39–47, Boulder, Colorado. Association for Computational Linguistics.

Bomford, A. (1980). The role of a national mapping organisation. *Survey Review*, 25:195–210.

Bossler, J. D., Finnie, T. C., Petchenik, B. B., and Usselman, T. M. (1990). Spatial data needs: The future of the national mapping program. *Cartography and Geographic Information Systems*, 17(3):237–242.

Branson, S., Wah, C., Schroff, F., Babenko, B., Welinder, P., Perona, P., and Belongie, S. (2010). Visual recognition with humans in the loop. In Daniilidis, K., Maragos, P., and Paragios, N., editors, *Computer Vision – ECCV 2010*, pages 438–451, Berlin, Heidelberg. Springer Berlin Heidelberg.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1):5–32.

Breiman, L., Friedman, J., Stone, C., and Olshen, R. (1984). *Classification and Regression Trees*. Taylor & Francis.

Brodu, N. and Lague, D. (2012). 3D terrestrial lidar data classification of complex natural scenes using a multi-scale dimensionality criterion: Applications in geomorphology. *ISPRS Journal of Photogrammetry and Remote Sensing*, 68:121–134.

Budhathoki, N. R. and Haythornthwaite, C. (2012). Motivation for Open Collaboration: Crowd and Community Models and the Case of OpenStreetMap. *American Behavioral Scientist*, 57(5):548–575.

Cabello, R. (2019). Three.js [WWW Document]. URL: https://threejs.org (accessed January 15, 2020).

Carillo, K. and Okoli, C. (2008). The open source movement: A revolution in software development. *Journal of Computer Information Systems*, 49(2):1–9.

Chamberlain, J., Fort, K., Kruschwitz, U., Lafourcade, M., and Poesio, M. (2013). Using games to create language resources: Successes and limitations of the approach. In *The People's Web Meets NLP*, pages 3–44. Springer Berlin Heidelberg.

Chan, Y. S. and Ng, H. T. (2007). Domain adaptation with active learning for word sense disambiguation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 49–56, Prague, Czech Republic. Association for Computational Linguistics.

Chandler, D. and Kapelner, A. (2013). Breaking monotony with meaning: Motivation in crowdsourcing markets. *Journal of Economic Behavior & Organization*, 90:123–133.

Chandler, J., Paolacci, G., and Mueller, P. (2013). Risks and rewards of crowdsourcing marketplaces. In *Handbook of Human Computation*, pages 377–392. Springer New York.

Chandler, J. J. and Paolacci, G. (2017). Lie for a dime. *Social Psychological and Personality Science*, 8(5):500–508.

Charaniya, A., Manduchi, R., and Lodha, S. (2004). Supervised parametric classification of aerial LiDAR data. In *2004 Conference on Computer Vision and Pattern Recognition Workshop*. IEEE.

Chattopadhyay, R., Fan, W., Davidson, I., Panchanathan, S., and Ye, J. (2013). Joint transfer and batch-mode active learning. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28(3) of *Proceedings of Machine Learning Research*, pages 253–261, Atlanta, Georgia, USA. PMLR.

Chehata, N., Guo, L., and Mallet, C. (2009). Airborne LiDAR Feature Selection For Urban Classification Using Random Forests. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIII-3/W8:207–212.

Chellapilla, K., Puri, S., and Simard, P. (2006). High Performance Convolutional Neural Networks for Document Processing. In Lorette, G., editor, *Tenth International Workshop on Frontiers in Handwriting Recognition*, La

Baule (France). Université de Rennes 1.

Chen, X., Golovinskiy, A., and Funkhouser, T. (2009). A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3).

Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., and Shelhamer, E. (2014). cudnn: Efficient primitives for deep learning. *CoRR*, abs/1410.0759.

Cho, G. and Crompvoets, J. (2019). Prod-users of geospatial information: some legal perspectives. *Journal of Spatial Science*, 64(2):341–358.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Cramer, M. (2010). The DGPF-Test on Digital Airborne Camera Evaluation – Overview and Test Design. *Photogrammetrie - Fernerkundung - Geoinformation*, 2010(2):73–82.

Crawford, M. M., Tuia, D., and Yang, H. L. (2013). Active learning: Any value for classification of remotely sensed data? *Proceedings of the IEEE*, 101(3):593–608.

Criminisi, A. and Shotton, J., editors (2013). *Decision Forests for Computer Vision and Medical Image Analysis*. Springer London.

Dai, W., Yang, Q., Xue, G.-R., and Yu, Y. (2007). Boosting for transfer learning. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 193–200, New York, NY, USA. Association for Computing Machinery.

Daniel, F., Kucherbaev, P., Cappiello, C., Benatallah, B., and Allahbakhsh, M. (2018). Quality control in crowdsourcing. *ACM Computing Surveys*, 51(1):1–40.

Dasgupta, S. and Hsu, D. (2008). Hierarchical sampling for active learning. In *Proceedings of the 25th international conference on Machine learning - ICML '08*. ACM Press.

Dawid, A. P. and Skene, A. M. (1979). Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):20–28.

De Albuquerque, J., Eckle, M., Herfort, B., and Zipf, A. (2016). *European Handbook of Crowdsourced Geographic Information*, chapter Crowdsourcing geographic information for disaster management and improving urban resilience: an overview of recent developments and lessons learned, pages 309–321. Ubiquity Press.

Demantké, J., Mallet, C., David, N., and Vallet, B. (2012). Dimensionality based scale selection in 3D lidar point clouds. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIII-5/W12:97–102.

Demir, B., Bovolo, F., and Bruzzone, L. (2013). Updating land-cover maps by classification of image time series: A novel change-detection-driven transfer learning approach. *IEEE Transactions on Geoscience and Remote Sensing*, 51(1):300–312.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Li, F. F. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR 2009*, pages 248–255.

Deng, J., Krause, J., and Fei-Fei, L. (2013). Fine-grained crowdsourcing for fine-grained recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

Deng, J., Krause, J., Stark, M., and Fei-Fei, L. (2016). Leveraging the wisdom of the crowd for fine-grained recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4):666–676.

Deng, J., Russakovsky, O., Krause, J., Bernstein, M. S., Berg, A., and Fei-Fei, L. (2014). Scalable multi-label annotation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM.

Deterding, S., Sicart, M., Nacke, L., O'Hara, K., and Dixon, D. (2011). Gamification. using game-design elements in non-gaming contexts. In *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems - CHI EA '11*. ACM Press.

Dittrich, A., Weinmann, M., and Hinz, S. (2017). Analytical and numerical investigations on the accuracy and robustness of geometric features extracted from 3D point cloud data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 126:195–208.

Donmez, P., Carbonell, J. G., and Schneider, J. (2009). Efficiently learning the accuracy of labeling sources for selective sampling. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09*. ACM Press.

Dorn, H., Törnros, T., and Zipf, A. (2015). Quality evaluation of VGI using authoritative data—a comparison with land use data in southern germany. *ISPRS International Journal of Geo-Information*, 4(3):1657–1671.

Duggan, M. (2019). Cultures of enthusiasm: An ethnographic study of amateur map-maker communities. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 54(3):217–229.

Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1).

Eickhoff, C., Harris, C. G., de Vries, A. P., and Srinivasan, P. (2012). Quality through flow and immersion. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval - SIGIR '12*. ACM Press.

Elkan, C. (2001). The foundations of cost-sensitive learning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'01, page 973–978, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Endres, I., Farhadi, A., Hoiem, D., and Forsyth, D. A. (2010). The benefits and challenges of collecting richer object annotations. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, pages 1–8.

Ertekin, S., Huang, J., Bottou, L., and Giles, L. (2007). Learning on the Border: Active Learning in Imbalanced Data Classification. In *CIKM 2007*, pages 127–136, New York, NY, USA. ACM.

Estes, L., McRitchie, D., Choi, J., Debats, S., Evans, T., Guthe, W., Luo, D., Ragazzo, G., Zempleni, R., and Caylor, K. (2016). A platform for crowdsourcing the creation of representative, accurate landcover maps. *Environmental Modelling & Software*, 80:41–53.

Fan, H., Zipf, A., Fu, Q., and Neis, P. (2014). Quality assessment for building footprints data on OpenStreetMap. *International Journal of Geographical Information Science*, 28(4):700–719.

Feng, D., Wei, X., Rosenbaum, L., Maki, A., and Dietmayer, K. (2019). Deep active learning for efficient training of a LiDAR 3d object detector. In *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE.

Feyisetan, O., Simperl, E., Van Kleek, M., and Shadbolt, N. (2015). Improving paid microtasks through gamification and adaptive furtherance incentives. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, page 333–343, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Filin, S. and Pfeifer, N. (2005). Neighborhood systems for airborne laser data. *Photogrammetric Engineering & Remote Sensing*, 71(6):743–755.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24:381–395.

Flanagin, A. J. and Metzger, M. J. (2008). The credibility of volunteered geographic information. *GeoJournal*, 72(3-4):137–148.

Fleischer, A., Mead, A. D., and Huang, J. (2015). Inattentive responding in MTurk and other online samples. *Industrial and Organizational Psychology*, 8(2):196–202.

Fonte, C., Antoniou, V., Bastin, L., Estima, J., Jokar Arsanjani, J., Laso Bayas, J., See, L., and Vatseva, R. (2017). *Assessing VGI Data Quality*, pages 137–163. Ubiquity Press.

Frey, B. S. and Jegen, R. (2001). Motivation crowding theory. *Journal of Economic Surveys*, 15(5):589–611.

Fu, B., Cao, Z., Wang, J., and Long, M. (2021). Transferable query selection for active domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7272–7281.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *ICML 2016*, volume 48, pages 1050–1059, New York, NY, USA. PMLR.

Gal, Y., Islam, R., and Ghahramani, Z. (2017). Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1183–1192. JMLR.org.

Galton, F. (1907). Vox populi. *Nature*, 75(1949):450–451.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17(1):2096–2030.

Gao, M., Xu, W., and Callison-Burch, C. (2015). Cost optimization in crowdsourcing translation: Low cost translations

made even cheaper. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Gao, W., Nan, L., Boom, B., and Ledoux, H. (2021). SUM: A benchmark dataset of semantic urban meshes. *ISPRS Journal of Photogrammetry and Remote Sensing*, 179:108–120.

Gebru, T., Krause, J., Deng, J., and Fei-Fei, L. (2017). Scalable annotation of fine-grained categories without experts. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*.

Geiger, D., Seedorf, S., Schulze, T., Nickerson, R. C., and Schader, M. (2011). Managing the crowd: Towards a taxonomy of crowdsourcing processes. In *AMCIS*.

Gerke, M., Rottensteiner, F., Wegner, J. D., and Gunho Sohn (2014). ISPRS semantic labeling contest. In *PCV - Photogrammetric Computer Vision*. Unpublished.

Giles, J. (2005). Internet encyclopaedias go head to head. *Nature*, 438(7070):900–901.

Gingold, Y., Shamir, A., and Cohen-Or, D. (2012). Micro perceptual human computation for visual tasks. *ACM Transactions on Graphics*, 31(5):1–12.

Girardeau-Montaut, D. (2021). CloudCompare 3D point cloud and mesh processing software open source project [WWW Document]. URL: https://www.danielgm.net/cc/ (accessed February 2, 2021).

Girardeau-Montaut, D., Roux, M., Marc, R., and Thibault, G. (2005). Change detection on point cloud data acquired with a ground laser scanner. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36.

Glira, P., Pfeifer, N., and Mandlburger, G. (2019). Hybrid orientation of airborne lidar point clouds and aerial images. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W5:567–574.

Goodchild, M. F. (2007). Citizens as sensors: the world of volunteered geography. *GeoJournal*, 69(4):211–221.

Goodchild, M. F. and Li, L. (2012). Assuring the quality of volunteered geographic information. *Spatial Statistics*, 1:110–120.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Goutte, C. and Gaussier, E. (2005). A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation. In *Lecture Notes in Computer Science*, pages 345–359. Springer Berlin Heidelberg.

Graham, B. (2015). Sparse 3D convolutional neural networks.

Graham, B., Engelcke, M., and v. d. Maaten, L. (2018). 3D Semantic Segmentation with Submanifold Sparse Convolutional Networks. In *CVPR 2018*, pages 9224–9232.

Griffiths, D. and Boehm, J. (2019). A review on deep learning techniques for 3d sensed data classification. In *Remote Sensing*, volume 11.

Gross, H. and Thoennessen, U. (2006). Extraction of lines from laser point clouds. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVI-3:86–91.

Guo, B., Huang, X., Zhang, F., and Sohn, G. (2015). Classification of airborne laser scanning data using JointBoost. *ISPRS Journal of Photogrammetry and Remote Sensing*, 100:71–83.

Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., and Bennamoun, M. (2021). Deep learning for 3D point clouds: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(12):4338–4364.

Haala, N., Hastedt, H., Wolf, K., Ressl, C., and Baltrusch, S. (2010). Digital photogrammetric camera evaluation generation of digital elevation models. *Photogrammetrie - Fernerkundung - Geoinformation*, 2010(2):99–115.

Haala, N., Kölle, M., Cramer, M., Laupheimer, D., Mandlburger, G., and Glira, P. (2020). Hybrid georeferencing, enhancement and classification of ultra-high resolution UAV LiDAR and image point clouds for monitoring applications. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, V-2-2020:727–734.

Haala, N., Kölle, M., Cramer, M., Laupheimer, D., and Zimmermann, F. (2022). Hybrid georeferencing of images and LiDAR data for UAV-based point cloud collection at millimetre accuracy. *ISPRS Open Journal of Photogrammetry and Remote Sensing*, 4:100014.

Hackel, T., Wegner, J. D., and Schindler, K. (2016). Fast semantic segmentation of 3d point clouds with strongly varying density. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 177–184.

Haklay, M. and Weber, P. (2008). OpenStreetMap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–

18.

Haklay, M. M., Basiouka, S., Antoniou, V., and Ather, A. (2010). How many volunteers does it take to map an area well? the validity of linus' law to volunteered geographic information. *The Cartographic Journal*, 47(4):315–322.

Hara, K., Azenkot, S., Campbell, M., Bennett, C. L., Le, V., Pannella, S., Moore, R., Minckler, K., Ng, R. H., and Froehlich, J. E. (2015). Improving public transit accessibility for blind riders by crowdsourcing bus stop landmark locations with google street view: An extended analysis. *ACM Transactions on Accessible Computing*, 6(2):1–23.

Haralabopoulos, G., Wagner, C., McAuley, D., and Anagnostopoulos, I. (2019). Paid crowdsourcing, low income contributors, and subjectivity. In *IFIP Advances in Information and Communication Technology*, pages 225–231. Springer International Publishing.

Harris, C. G. (2011). You're hired! an examination of crowdsourcing incentive models in human resource tasks. *Crowdsourcing for Search and Data Mining (CSDM 2011)*, pages 15–18.

Hashemi, P. and Abbaspour, R. A. (2015). Assessment of logical consistency in OpenStreetMap based on the spatial similarity concept. In *Lecture Notes in Geoinformation and Cartography*, pages 19–36. Springer International Publishing.

He, Y., Yu, H., Liu, X., Yang, Z., Sun, W., Wang, Y., Fu, Q., Zou, Y., and Mian, A. (2021). Deep learning based 3D segmentation: A survey.

Hecht, R., Kalla, M., and Krüger, T. (2018). Crowd-sourced data collection to support automatic classification of building footprint data. *Proceedings of the ICA*, 1:1–7.

Heckman, J. J. (1979). Sample selection bias as a specification error. *Econometrica*, 47(1):153–161.

Herfort, B., Eckle, M., and de Albuquerque, J. P. (2016). Being specific about geographic information crowdsourcing: A typology and analysis of the missing maps project in south kivu. In *ISCRAM*.

Herfort, B., Höfle, B., and Klonner, C. (2018). 3D micro-mapping: Towards assessing the quality of crowdsourcing to support 3D point cloud analysis. *ISPRS Journal of Photogrammetry and Remote Sensing*, 137:73–83.

Hiep, V. H., Keriven, R., Labatut, P., and Pons, J.-P. (2009). Towards high-resolution large-scale multi-view stereo. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE.

Hillen, F. and Höfle, B. (2015). Geo-reCAPTCHA: Crowdsourcing large amounts of geographic information from earth observation data. *International Journal of Applied Earth Observation and Geoinformation*, 40:29–38.

Hirth, M., Hoßfeld, T., and Tran-Gia, P. (2011). Anatomy of a Crowdsourcing Platform - Using the Example of Microworkers.com. In *IMIS 2011*, pages 322–329, Washington, DC, USA. IEEE Computer Society.

Hirth, M., Hoßfeld, T., and Tran-Gia, P. (2013). Analyzing costs and accuracy of validation mechanisms for crowdsourcing platforms. *Mathematical and Computer Modelling*, 57(11-12):2918–2932.

Hitlin, P., Rainie, L., Hatley, N., Smith, A., van Kessel, P., Broderick, B., Duggan, M., and Perrin, A. (2016). Research in the Crowdsourcing Age, a Case Study [WWW Document]. URL: https://www.pewresearch.org/internet/2016/07/11/research-in-the-crowdsourcing-age-a-case-study/ (accessed 15.1.20).

Ho, C.-J., Slivkins, A., Suri, S., and Vaughan, J. W. (2015). Incentivizing high quality crowdwork. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee.

Ho, T. K. and Baird, H. (1997). Large-scale simulation studies in image pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(10):1067–1079.

Horton, J. J. and Chilton, L. B. (2010). The labor economics of paid crowdsourcing. In *Proceedings of the 11th ACM conference on Electronic commerce - EC '10*. ACM Press.

Houlsby, N., Huszár, F., Ghahramani, Z., and Lengyel, M. (2011). Bayesian active learning for classification and preference learning.

Howe, J. (2006). The rise of crowdsourcing. *Wired Magazine*, 6(14):1–4.

Hoßfeld, T., Hirth, M., and Tran-Gia, P. (2011). Modeling of crowdsourcing platforms and granularity of work organization in future internet. In *2011 23rd International Teletraffic Congress (ITC)*, pages 142–149.

Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N., and Markham, A. (2020). RandLA-net: Efficient semantic segmentation of large-scale point clouds. In *2020 IEEE/CVF Conference on Computer Vision and Pattern*

Recognition (CVPR). IEEE.

Huang, A. F. . R. (2021). 3D Scene Understanding with TensorFlow 3D [WWW Document]. URL: https://ai.googleblog.com/2021/02/3d-scene-understanding-with-tensorflow.html (accessed May 4, 2022).

Hui, Z., Jin, S., Cheng, P., Ziggah, Y. Y., Wang, L., Wang, Y., Hu, H., and Hu, Y. (2019). An Active Learning Method for DEM Extraction from Airborne LiDAR Point Clouds. *IEEE Access*, 7:89366–89378.

Hüllermeier, E. and Waegeman, W. (2021). Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Machine Learning*, 110(3):457–506.

Ikeda, K., Morishima, A., Rahman, H., Roy, S. B., Thirumuruganathan, S., Amer-Yahia, S., and Das, G. (2016). Collaborative crowdsourcing with crowd4u. *Proceedings of the VLDB Endowment*, 9(13):1497–1500.

Ipeirotis, P. (2010a). Demographics of mechanical turk. *NYU Working Paper*, CEDER-10-01.

Ipeirotis, P. G. (2010b). Analyzing the amazon mechanical turk marketplace. *XRDS: Crossroads, The ACM Magazine for Students*, 17(2):16–21.

Jain, S. D. and Grauman, K. (2013). Predicting sufficient annotation strength for interactive foreground segmentation. In *2013 IEEE International Conference on Computer Vision*. IEEE.

Jiang, J. and Zhai, C. (2007). Instance weighting for domain adaptation in NLP. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 264–271, Prague, Czech Republic. Association for Computational Linguistics.

Jospin, L. V., Laga, H., Boussaid, F., Buntine, W., and Bennamoun, M. (2022). Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2):29–48.

Jun, G. and Ghosh, J. (2008). An efficient active learning algorithm with knowledge transfer for hyperspectral data analysis. In *IGARSS 2008 - 2008 IEEE International Geoscience and Remote Sensing Symposium*, volume 1, pages I52–I55.

Juni, M. Z. and Eckstein, M. P. (2017). The wisdom of crowds for visual search. *Proceedings of the National Academy of Sciences*, 114(21):E4306–E4315.

Jutzi, B. and Gross, H. (2009). Nearest neighbour classification on laser point clouds to gain object structures from buildings. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIII-1-4-7/W5.

Kamar, E. (2016). Directions in hybrid intelligence: Complementing ai systems with human intelligence. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 4070–4073. AAAI Press.

Kamar, E., Hacker, S., and Horvitz, E. (2012). Combining human and machine intelligence in large-scale crowdsourcing. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, page 467–474, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Kazhdan, M., Bolitho, M., and Hoppe, H. (2006). Poisson Surface Reconstruction. In Sheffer, A. and Polthier, K., editors, *Symposium on Geometry Processing*. The Eurographics Association.

Kellenberger, B., Marcos, D., Lobry, S., and Tuia, D. (2019). Half a Percent of Labels is Enough: Efficient Animal Detection in UAV Imagery Using Deep CNNs and Active Learning. *TRGS*, 57(12):9524–9533.

Khoshelham, K., Elberink, S. O., and Xu, S. (2013). Segment-based classification of damaged building roofs in aerial laser scanning data. *IEEE Geoscience and Remote Sensing Letters*, 10:1258–1262.

Kim, H. and Sohn, G. (2010). 3D classification of power-line scene from airborne laser scanning data using random forests. In *PCV2010*.

Kirsch, A., van Amersfoort, J., and Gal, Y. (2019). BatchBALD: Efficient and Diverse Batch Acquisition for Deep Bayesian Active Learning. In *NIPS 2019*, pages 7026–7037. Curran Associates, Inc.

Kittur, A., Chi, E. H., and Suh, B. (2008). Crowdsourcing user studies with mechanical turk. In *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*. ACM Press.

Kittur, A., Nickerson, J. V., Bernstein, M., Gerber, E., Shaw, A., Zimmerman, J., Lease, M., and Horton, J. (2013). The future of crowd work. In *Proceedings of the 2013 conference on Computer supported cooperative work - CSCW '13*. ACM Press.

Kittur, A., Smus, B., Khamkar, S., and Kraut, R. E. (2011). CrowdForge. In *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11*. ACM Press.

Kölle, M., Laupheimer, D., Schmohl, S., Haala, N., Rottensteiner, F., Wegner, J. D., and Ledoux, H. (2021a). The hessigheim 3D (H3D) benchmark on semantic segmentation of high-resolution 3D point clouds and textured meshes from uav lidar and multi-view-stereo. *ISPRS Open Journal of Photogrammetry and Remote Sensing*, 1:100001.

Kölle, M., Laupheimer, D., Walter, V., Haala, N., and Soergel, U. (2021b). Which 3D data representation does the crowd like best? crowd-based active learning for coupled semantic segmentation of point clouds and textured meshes. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, V-2-2021:93–100.

Kölle, M., Walter, V., Schmohl, S., and Soergel, U. (2021a). Remembering both the machine and the crowd when sampling points: Active learning for semantic segmentation of ALS point clouds. In *ICPR International Workshops and Challenges*, pages 505–520, Cham. Springer International Publishing.

Kölle, M., Walter, V., Shiller, I., and Soergel, U. (2021b). Categorise: An automated framework for utilizing the workforce of the crowd for semantic segmentation of 3D point clouds. In *ICPR International Workshops and Challenges*, pages 505–520, Cham. Springer International Publishing.

Konyushkova, K., Sznitman, R., and Fua, P. (2017). Learning active learning from data. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Korpela, E., Werthimer, D., Anderson, D., Cobb, J., and Leboisky, M. (2001). Seti@home-massively distributed computing for seti. *Computing in Science Engineering*, 3(1):78–83.

Kovashka, A., Russakovsky, O., Fei-Fei, L., and Grauman, K. (2016). Crowdsourcing in Computer Vision. *Foundations and Trends in Computer Graphics and Vision*, 10(3):177–243.

Krause, J., Ruxton, G. D., and Krause, S. (2010). Swarm intelligence in animals and humans. *Trends in Ecology & Evolution*, 25(1):28–34.

Krause, S., James, R., Faria, J. J., Ruxton, G. D., and Krause, J. (2011). Swarm intelligence in humans: diversity can trump ability. *Animal Behaviour*, 81(5):941–948.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Kumar, N., Berg, A. C., Belhumeur, P. N., and Nayar, S. K. (2009). Attribute and simile classifiers for face verification. In *2009 IEEE 12th International Conference on Computer Vision*. IEEE.

Kölle, M., Walter, V., and Soergel, U. (2022). Learning from the past: Crowd-driven active transfer learning for semantic segmentation of multi-temporal 3D point clouds. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, V-2-2022:259–266.

Labatut, P., Pons, J.-P., and Keriven, R. (2007). Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In *2007 IEEE 11th International Conference on Computer Vision*. IEEE.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6405–6416, Red Hook, NY, USA. Curran Associates Inc.

Lalonde, J., Unnikrishnan, R., Vandapel, N., and Hebert, M. (2005). Scale selection for classification of point-sampled 3-D surfaces. In *Fifth International Conference on 3-D Digital Imaging and Modeling (3DIM'05)*. IEEE.

Lampel, J. and Bhalla, A. (2007). The role of status seeking in online communities: Giving the gift of experience. *Journal of Computer-Mediated Communication*, 12(2):434–455.

Landrieu, L., Mallet, C., and Weinmann, M. (2017). Comparison of belief propagation and graph-cut approaches for contextual classification of 3D lidar point cloud data. *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 2768–2771.

Lecun, Y. (1989). Generalization and network design strategies. Computer Sciences Technical Report CRG-TR-89-4, University of Toronto.

Lee, I. and Schenk, A. (2002). Perceptual organization of 3D surface points. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 34.

Leibovici, D., Rosser, J., Hodges, C., Evans, B., Jackson, M., and Higgins, C. (2017). On data quality assurance and conflation entanglement in crowdsourcing for environmental studies. *ISPRS International Journal of Geo-Information*, 6(3):78.

Leorke, D. (2019). *Location-Based Gaming*. Springer Singapore.

Lewis, D. D. and Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *SIGIR '94*, pages 3–12. Springer London.

Li, H. and Zipf, A. (2022). A conceptual model for converting openstreetmap contribution to geospatial machine learning training data. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLIII-B4-2022:253–259.

Li, N., Kahler, O., and Pfeifer, N. (2021). A comparison of deep learning methods for airborne lidar point clouds classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14:6467–6486.

Li, N. and Pfeifer, N. (2019). Active learning to extend training data for large area airborne lidar classification. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W13:1033–1037.

Lichtenberg, S., , Lembcke, T.-B., Brening, M., Brendel, A. B., and Trang, S. (2020). Can gamification lead to increase paid crowdworkers output? In *WI2020 Zentrale Tracks*, pages 1188–1202. GITO Verlag.

Lin, Y., Vosselman, G., Cao, Y., and Yang, M. Y. (2020a). Active and incremental learning for semantic ALS point cloud segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 169:73–92.

Lin, Y., Vosselman, G., Cao, Y., and Yang, M. Y. (2020b). Efficient training of semantic point cloud segmentation via active learning. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, V-2-2020:243–250.

Lin, Y., Vosselman, G., and Yang, M. Y. (2022). Weakly supervised semantic segmentation of airborne laser scanning point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 187:79–100.

Linsen, L. and Prautzsch, H. (2001). Local Versus Global Triangulations. In *Eurographics 2001 - Short Presentations*. Eurographics Association.

Liu, Z., Shabani, S., Balet, N. G., Sokhn, M., and Cretton, F. (2018). How to motivate participation and improve quality of crowdsourcing when building accessibility maps. In *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE.

Lloyd, S. P. (1982). Least squares quantization in PCM. *IEEE Trans. Inf. Theory*, 28(2):129–137.

Lockhart, J., Assefa, S., Balch, T., and Veloso, M. (2020). Some people aren't worth listening to: periodically retraining classifiers with feedback from a team of end users. *CoRR*, abs/2004.13152.

Lodha, S. K., Fitzpatrick, D. M., and Helmbold, D. P. (2007). Aerial lidar data classification using AdaBoost. In *Sixth International Conference on 3-D Digital Imaging and Modeling (3DIM 2007)*. IEEE.

Lodha, S. K., Kreps, E. J., Helmbold, D. P., and Fitzpatrick, D. (2006). Aerial LiDAR data classification using support vector machines (SVM). In *Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*. IEEE.

Long, C. and Hua, G. (2015). Multi-class multi-annotator active learning with robust gaussian process for visual recognition. In *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE.

Long, C., Hua, G., and Kapoor, A. (2013). Active visual recognition with expertise estimation in crowdsourcing. In *2013 IEEE International Conference on Computer Vision*. IEEE.

Louppe, G. (2014). Understanding random forests: From theory to practice. *CoRR*, abs/1407.7502.

Lu, Y. and Rasmussen, C. (2012). Simplified markov random fields for efficient semantic labeling of 3D point clouds. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE.

Luo, H., Wang, C., Wen, C., Chen, Z., Zai, D., Yu, Y., and Li, J. (2018). Semantic labeling of mobile LiDAR point clouds via active learning and higher order MRF. *TGRS*, 56(7):3631–3644.

Ma, L., Liu, Y., Zhang, X., Ye, Y., Yin, G., and Johnson, B. A. (2019). Deep learning in remote sensing applications: A meta-analysis and review. *ISPRS Journal of Photogrammetry and Remote Sensing*, 152:166–177.

Maas, H.-G. and Vosselman, G. (1999). Two algorithms for extracting building models from raw laser altimetry data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54(2-3):153–163.

Mackowiak, R., Lenz, P., Ghori, O., Diego, F., Lange, O., and Rother, C. (2018). CEREALS - Cost-Effective REgion-based Active Learning for Semantic Segmentation. *BMVC 2018*.

Maddalena, E., Ibáñez, L.-D., and Simperl, E. (2020). Mapping points of interest through street view imagery and paid crowdsourcing. *ACM Transactions on Intelligent Systems and Technology*, 11(5):1–28.

Mallet, C., Bretar, F., Roux, M., Soergel, U., and Heipke, C. (2011). Relevance assessment of full-waveform lidar data for urban area classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(6):S71–S84.

Mallet, C., Soergel, U., and Bretar, F. (2008). Analysis of full-waveform lidar data for classification of urban areas. *Photogrammetrie Fernerkundung GeoInformation (PFG)*, 5.

Mandlburger, G., Lehner, H., and Pfeifer, N. (2019). A comparison of single photon and full waveform lidar. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W5:397–404.

Mao, A., Kamar, E., Chen, Y., Horvitz, E., Schwamb, M. E., Lintott, C. J., and Smith, A. M. (2013). Volunteering versus work for pay: Incentives and tradeoffs in crowdsourcing. In *In Proceedings of the First AAAI Conference on Human Computation and Crowdsourcing (HCOMP '13*.

Marcus, A. and Parameswaran, A. (2015). Crowdsourced data management: Industry and academic perspectives. *Foundations and Trends in Databases*, 6(1-2):1–161.

Marczewski, A. (2013). *Gamification: A Simple Introduction and a Bit More*. Lulu.com Digital.

Martella, R., Kray, C., and Clementini, E. (2015). A gamification framework for volunteered geographic information. In *Lecture Notes in Geoinformation and Cartography*, pages 73–89. Springer International Publishing.

Mason, W. and Watts, D. J. (2009). Financial incentives and the "performance of crowds". In *Proceedings of the ACM SIGKDD Workshop on Human Computation - HCOMP '09*. ACM Press.

Matasci, G., Tuia, D., and Kanevski, M. (2012). Svm-based boosting of active learning strategies for efficient domain adaptation. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 5(5):1335–1343.

Matyas, S., Kiefer, P., Schlieder, C., and Kleyer, S. (2011). Wisdom about the crowd: Assuring geospatial data quality collected in location-based games. In *Entertainment Computing – ICEC 2011*, pages 331–336. Springer Berlin Heidelberg.

McCallum, A. and Nigam, K. (1998). Employing EM and pool-based active learning for text classification. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, page 350–358, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Morschheuser, B., Hamari, J., Koivisto, J., and Maedche, A. (2017). Gamified crowdsourcing: Conceptualization, literature review, and future agenda. *International Journal of Human-Computer Studies*, 106:26–43.

Najafi, M., Namin, S. T., Salzmann, M., and Petersson, L. (2014). Non-associative higher-order markov networks for point cloud classification. In *Computer Vision – ECCV 2014*, pages 500–515. Springer International Publishing.

Ng, A. (2021). The batch - weekly issue 84 [WWW Document]. URL: https://www.deeplearning.ai/the-batch/issue-84/ (accessed October 18, 2022).

Nguyen, V.-L., Shaker, M. H., and Hüllermeier, E. (2021). How to measure uncertainty in uncertainty sampling for active learning. *Machine Learning*, 111(1):89–122.

Niemeyer, J. (2017). *Verwendung von Kontext zur Klassifikation luftgestützter Laserdaten urbaner Gebiete*. PhD thesis, Leibniz University Hannover.

Niemeyer, J., Rottensteiner, F., and Soergel, U. (2012). Conditional Random Fields For LiDAR Point Cloud Classification In Complex Urban Areas. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, I-3:263–268.

Niemeyer, J., Rottensteiner, F., and 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.

Okolloh, O. (2009). Ushahidi, or 'testimony': Web 2.0 tools for crowdsourcing crisis information. *Participatory Learning and Action*, 59:65–70.

Olsson, F. and Tomanek, K. (2009). An intrinsic stopping criterion for committee-based active learning. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, page 138–146, USA. Association for Computational Linguistics.

Osada, R., Funkhouser, T., Chazelle, B., and Dobkin, D. (2002). Shape distributions. *ACM Transactions on Graphics*,

21(4):807–832.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Parhami, B. (1994). Voting algorithms. *IEEE Transactions on Reliability*, 43(4):617–629.

Paris, C. and Bruzzone, L. (2018). A sensor-driven hierarchical method for domain adaptation in classification of remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 56(3):1308–1324.

Patel, V. M., Gopalan, R., Li, R., and Chellappa, R. (2015). Visual domain adaptation: A survey of recent advances. *IEEE Signal Processing Magazine*, 32(3):53–69.

Patterson, G., Xu, C., Su, H., and Hays, J. (2014). The SUN attribute database: Beyond categories for deeper scene understanding. *International Journal of Computer Vision*, 108(1-2):59–81.

Paul, A., Rottensteiner, F., and Heipke, C. (2016). Iterative re-weighted instance transfer for domain adaptation. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, III-3:339–346.

Pauly, M., Keiser, R., and Gross, M. (2003). Multi-scale feature extraction on point-sampled surfaces. *Computer Graphics Forum*, 22(3):281–289.

Penatti, O. A. B., Nogueira, K., and dos Santos, J. A. (2015). Do deep features generalize from everyday objects to remote sensing and aerial scenes domains? In *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 44–51.

Persello, C. (2013). Interactive domain adaptation for the classification of remote sensing images using active learning. *IEEE Geoscience and Remote Sensing Letters*, 10(4):736–740.

Persello, C. and Bruzzone, L. (2012). Active learning for domain adaptation in the supervised classification of remote sensing images. *IEEE transactions on geoscience and remote sensing*, 50(11):4468–4483.

Pfeifer, N., Mandlburger, G., Otepka, J., and Karel, W. (2014). OPALS – a framework for airborne laser scanning data analysis. *Computers, Environment and Urban Systems*, 45:125–136.

Pfeifer, N., Stadler, P., and Briese, C. (2001). Derivation of digital terrain models in the scop++ environment. In *OEEPE Workshop on Airborne Laserscanning and Interferometric SAR for Digital Elevation Models*.

Prabhu, V., Chandrasekaran, A., Saenko, K., and Hoffman, J. (2021). Active domain adaptation via clustering uncertainty-weighted embeddings. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8485–8494.

Prandi, C., Nisi, V., Salomoni, P., and Nunes, N. J. (2015). From gamification to pervasive game in mapping urban accessibility. In *Proceedings of the 11th Biannual Conference on Italian SIGCHI Chapter*. ACM.

Qi, C. R., Su, H., Kaichun, M., and Guibas, L. J. (2017a). PointNet: Deep learning on point sets for 3D classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017b). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 5105–5114, Red Hook, NY, USA. Curran Associates Inc.

Quiñonero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. D., editors (2008). *Dataset Shift in Machine Learning*. The MIT Press.

Radanovic, G., Faltings, B., and Jurca, R. (2016). Incentives for effort in crowdsourcing using the peer truth serum. *ACM Transactions on Intelligent Systems and Technology*, 7(4):1–28.

Rai, P., Saha, A., Daumé, H., and Venkatasubramanian, S. (2010). Domain adaptation meets active learning. In *Proceedings of the NAACL HLT 2010 Workshop on Active Learning for Natural Language Processing*, ALNLP '10, page 27–32, USA. Association for Computational Linguistics.

Rajan, S., Ghosh, J., and Crawford, M. M. (2008). An active learning approach to hyperspectral data classification. *IEEE Transactions on Geoscience and Remote Sensing*, 46(4):1231–1242.

Raykar, V. C., Yu, S., Zhao, L. H., Valadez, G. H., Florin, C., Bogoni, L., and Moy, L. (2010). Learning from crowds. *Journal of Machine Learning Research*, 11(43):1297–1322.

Redi, J. and Povoa, I. (2014). Crowdsourcing for rating image aesthetic appeal. In *Proceedings of the 2014 International ACM Workshop on Crowdsourcing for Multimedia - CrowdMM '14*. ACM Press.

Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Gupta, B. B., Chen, X., and Wang, X. (2022). A survey of deep

active learning. *ACM Computing Surveys*, 54(9):1–40.

Resch, B. (2013). People as sensors and collective sensing-contextual observations complementing geo-sensor network measurements. In *Lecture Notes in Geoinformation and Cartography*, pages 391–406. Springer Berlin Heidelberg.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F., editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham. Springer International Publishing.

Rothermel, M., Wenzel, K., Fritsch, D., and Haala, N. (2012). SURE: Photogrammetric Surface Reconstruction From Imagery. In *Proceedings LC3D Workshop*, volume 8, Berlin.

Rottensteiner, F., Sohn, G., Gerke, M., Wegner, J. D., Breitkopf, U., and Jung, J. (2014). Results of the ISPRS benchmark on urban object detection and 3D building reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 93:256–271.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015a). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

Russakovsky, O., Li, L.-J., and Fei-Fei, L. (2015b). Best of both worlds: Human-machine collaboration for object annotation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Russell, B. C., Torralba, A., Murphy, K. P., and Freeman, W. T. (2007). LabelMe: A database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1-3):157–173.

Rusu, R. B., Marton, Z. C., Blodow, N., Beetz, M., Systems, I. A., and München, T. U. (2008). Persistent point feature histograms for 3D point clouds. In *In Proceedings of the 10th International Conference on Intelligent Autonomous Systems (IAS-10*.

Saha, A., Rai, P., Daumé, H., Venkatasubramanian, S., and DuVall, S. L. (2011). Active supervised domain adaptation. In *Machine Learning and Knowledge Discovery in Databases*, pages 97–112. Springer Berlin Heidelberg.

Sailer, M., Hense, J., Mandl, H., and Klevers, M. (2013). Psychological perspectives on motivation through gamification. *Interaction Design and Architecture(s) Journal*, 19:18–37.

Sailer, M., Hense, J. U., Mayr, S. K., and Mandl, H. (2017). How gamification motivates: An experimental study of the effects of specific game design elements on psychological need satisfaction. *Computers in Human Behavior*, 69:371–380.

Salehi, N., Irani, L. C., Bernstein, M. S., Alkhatib, A., Ogbe, E., Milland, K., and Clickhappier (2015). We are dynamo. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM.

Salk, C. F., Sturn, T., See, L., Fritz, S., and Perger, C. (2015). Assessing quality of volunteer crowdsourcing contributions: lessons from the cropland capture game. *International Journal of Digital Earth*, 9(4):410–426.

Salomoni, P., Prandi, C., Roccetti, M., Nisi, V., and Nunes, N. J. (2015). Crowdsourcing urban accessibility:. In *Proceedings of the 11th Biannual Conference on Italian SIGCHI Chapter*. ACM.

Scheffer, T., Decomain, C., and Wrobel, S. (2001). Active hidden markov models for information extraction. In *Advances in Intelligent Data Analysis*, pages 309–318. Springer Berlin Heidelberg.

Schindler, K. (2012). An overview and comparison of smooth labeling methods for land-cover classification. *IEEE Transactions on Geoscience and Remote Sensing*, 50(11):4534–4545.

Schmidt, A., Niemeyer, J., Rottensteiner, F., and Soergel, U. (2014). Contextual classification of full waveform lidar data in the wadden sea. *IEEE Geoscience and Remote Sensing Letters*, 11(9):1614–1618.

Schmohl, S. and Sörgel, U. (2019). Submanifold Sparse Convolutional Networks For Semantic Segmentation of Large-Scale ALS Point Clouds. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W5:77–84.

Schmohl, S., Vallejo, A. N., and Soergel, U. (2022). Individual tree detection in urban ALS point clouds with 3D convolutional networks. *Remote Sensing*, 14(6):1317.

See, L., Comber, A., Salk, C., Fritz, S., van der Velde, M., Perger, C., Schill, C., McCallum, I., Kraxner, F., and Obersteiner, M. (2013). Comparing the quality of crowdsourced data contributed by expert and non-experts. *PLoS ONE*, 8(7):e69958.

See, L., Mooney, P., Foody, G., Bastin, L., Comber, A., Estima, J., Fritz, S., Kerle, N., Jiang, B., Laakso, M., Liu, H.-Y., Milčinski, G., Nikšič, M., Painho, M., Pődör, A., Olteanu-Raimond, A.-M., and Rutzinger, M. (2016).

Crowdsourcing, citizen science or volunteered geographic information? the current state of crowdsourced geographic information. *ISPRS International Journal of Geo-Information*, 5(5):55.

Senaratne, H., Mobasheri, A., Ali, A. L., Capineri, C., and Haklay, M. M. (2016). A review of volunteered geographic information quality assessment methods. *International Journal of Geographical Information Science*, 31(1):139–167.

Sener, O. and Savarese, S. (2018). Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*.

Settles, B. (2009). Active Learning Literature Survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

Shakarji, C. (1998). Least-squares fitting algorithms of the NIST algorithm testing system. *Journal of Research of the National Institute of Standards and Technology*, 103(6).

Shaker, M. H. and Hüllermeier, E. (2020). Aleatoric and epistemic uncertainty with random forests. In *Lecture Notes in Computer Science*, pages 444–456. Springer International Publishing.

Shan, J. and Toth, C. K., editors (2008). *Topographic Laser Ranging and Scanning: Principles and Processing*. CRC Press, Boca Raton, FL.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423.

Shao, F., Luo, Y., Liu, P., Chen, J., Yang, Y., Lu, Y., and Xiao, J. (2022). Active learning for point cloud semantic segmentation via spatial-structural diversity reasoning. *CoRR*, abs/2202.12588.

Shapovalov, R., Velizhev, E., and Barinova, O. (2010). Non-associative markov networks for 3D point cloud classification. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XXXVIII, Part 3A*, pages 103–108.

Sharpe, K., Huber, J., and Netzer, O. (2017). Mturk character misrepresentation: Assessment and solutions. *Journal of Consumer Research*, 44.

Shaw, A. D., Horton, J. J., and Chen, D. L. (2011). Designing incentives for inexpert human raters. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work - CSCW '11*. ACM Press.

Sheng, V. S., Provost, F., and Ipeirotis, P. G. (2008). Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 08*. ACM Press.

Shi, X., Fan, W., and Ren, J. (2008). Actively transfer domain knowledge. In *Machine Learning and Knowledge Discovery in Databases*, pages 342–357. Springer Berlin Heidelberg.

Shi, X., Xu, X., Chen, K., Cai, L., Foo, C. S., and Jia, K. (2021). Label-efficient point cloud semantic segmentation: An active learning approach. *CoRR*, abs/2101.06931.

Shimodaira, H. (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90:227–244.

Simons, A. M. (2004). Many wrongs: the advantage of group navigation. *Trends in Ecology & Evolution*, 19(9):453–455.

Sinha, S., Ebrahimi, S., and Darrell, T. (2019). Variational adversarial active learning. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5971–5980, Los Alamitos, CA, USA. IEEE Computer Society.

Sithole, G. and Vosselman, G. (2004). Experimental comparison of filter algorithms for bare-earth extraction from airborne laser scanning point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 59(1-2):85–101.

Smith, A. R. (1978). Color gamut transform pairs. *SIGGRAPH Computer Graphics*, 12(3):12–19.

Snow, R., O'Connor, B., Jurafsky, D., and Ng, A. (2008). Cheap and fast – but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 254–263, Honolulu, Hawaii. Association for Computational Linguistics.

Sorokin, A. and Forsyth, D. (2008). Utility data annotation with amazon mechanical turk. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE.

Spencer, R. W. (2012). Open innovation in the eighteenth century. *Research-Technology Management*, 55(4):39–43.

Spiro, I., Taylor, G., Williams, G., and Bregler, C. (2010). Hands by hand: Crowd-sourced motion tracking for gesture annotation. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition -*

Workshops. IEEE.

Stork, D. (1999). Character and document research in the open mind initiative. In *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR '99 (Cat. No.PR00318)*. IEEE.

Su, H., Deng, J., and Fei-Fei, L. (2012). Crowdsourcing annotations for visual object detection. In *Human Computation - Papers from the 2012 AAAI Workshop, Technical Report*, AAAI Workshop - Technical Report, pages 40–46. 2012 AAAI Workshop ; Conference date: 23-07-2012 Through 23-07-2012.

Su, J.-C., Tsai, Y.-H., Sohn, K., Liu, B., Maji, S., and Chandraker, M. (2019). Active adversarial domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Suri, S., Goldstein, D. G., and Mason, W. A. (2011). Honesty in an online labor market. In *Proceedings of the 11th AAAI Conference on Human Computation*, AAAIWS'11-11, page 61–66. AAAI Press.

Surowiecki, J. (2004). *The Wisdom of Crowds*. Anchor.

Thaler, S., Simperl, E., and Wölger, S. (2012). An experiment in comparing human-computation techniques. *IEEE Internet Computing*, 16(5):52–58.

Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F., and Guibas, L. (2019). KPConv: Flexible and deformable convolution for point clouds. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE.

Thoreau, R., Achard, V., Risser, L., Berthelot, B., and Briottet, X. (2022). Active learning for hyperspectral image classification: A comparative review. *IEEE Geoscience and Remote Sensing Magazine*, pages 2–24.

Tombari, F., Salti, S., and Di Stefano, L. (2010). Unique signatures of histograms for local surface description. In *Proceedings of the 11th European Conference on Computer Vision Conference on Computer Vision: Part III*, ECCV'10, page 356–369, Berlin, Heidelberg. Springer-Verlag.

Tuia, D. and Munoz-Mari, J. (2013). Learning user's confidence for active learning. *IEEE Transactions on Geoscience and Remote Sensing*, 51(2):872–880.

Tuia, D., Pasolli, E., and Emery, W. J. (2011a). Using active learning to adapt remote sensing image classifiers. *Remote Sensing of Environment*, 115:2232–2242.

Tuia, D., Persello, C., and Bruzzone, L. (2016). Domain adaptation for the classification of remote sensing data: An overview of recent advances. *IEEE Geoscience and Remote Sensing Magazine*, 4(2):41–57.

Tuia, D., Volpi, M., Copa, L., Kanevski, M., and Munoz-Mari, J. (2011b). A survey of active learning algorithms for supervised remote sensing image classification. *IEEE Journal of Selected Topics in Signal Processing*, 5(3):606–617.

van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

van Dijk, T. C., Fischer, N., and Häussner, B. (2020). Algorithmic improvement of crowdsourced data. In *Proceedings of the 28th International Conference on Advances in Geographic Information Systems*. ACM.

Varney, N., Asari, V. K., and Graehling, Q. (2020). Dales: A large-scale aerial lidar data set for semantic segmentation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 717–726.

Vaughan, J. W. (2018). Making better use of the crowd: How crowdsourcing can advance machine learning research. *Journal of Machine Learning Research*, 18(193):1–46.

Vignieri, V. (2021). Crowdsourcing as a mode of open innovation: Exploring drivers of success of a multisided platform through system dynamics modelling. *Systems Research and Behavioral Science*, 38(1):108–124.

Vijayanarasimhan, S. and Grauman, K. (2009). What's it going to cost you?: Predicting effort vs. informativeness for multi-label image annotations. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE.

Vijayanarasimhan, S. and Grauman, K. (2011). Large-scale live active learning: Training object detectors with crawled data and crowds. In *CVPR 2011*, pages 1449–1456.

Vlachos, A. (2008). A stopping criterion for active learning. *Computer Speech & Language*, 22(3):295–312.

von Ahn, L., Blum, M., Hopper, N. J., and Langford, J. (2003). CAPTCHA: Using hard AI problems for security. In *Lecture Notes in Computer Science*, pages 294–311. Springer Berlin Heidelberg.

von Ahn, L. and Dabbish, L. (2004). Labeling images with a computer game. In *Proceedings of the 2004 conference on Human factors in computing systems - CHI '04*. ACM Press.

von Ahn, L. and Dabbish, L. (2008). Designing games with a purpose. *Communications of the ACM*, 51(8):58–67.

von Ahn, L., Ginosar, S., Kedia, M., Liu, R., and Blum, M. (2006a). Improving accessibility of the web with a computer game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM.

von Ahn, L., Liu, R., and Blum, M. (2006b). Peekaboom. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM.

von Ahn, L., Maurer, B., McMillen, C., Abraham, D., and Blum, M. (2008). reCAPTCHA: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468.

Vondrick, C., Patterson, D., and Ramanan, D. (2012). Efficiently scaling up crowdsourced video annotation. *International Journal of Computer Vision*, 101(1):184–204.

Vosselman, G., Coenen, M., and Rottensteiner, F. (2017). Contextual segment-based classification of airborne laser scanner data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 128:354–371.

Vosselman, G. and Maas, H., editors (2010). *Airborne and terrestrial laser scanning*. CRC Press, United Kingdom.

Wagner, W., Ullrich, A., Ducic, V., Melzer, T., and Studnicka, N. (2006). Gaussian decomposition and calibration of a novel small-footprint full-waveform digitising airborne laser scanner. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60(2):100–112.

Wah, C., Branson, S., Perona, P., and Belongie, S. (2011). Multiclass recognition and part localization with humans in the loop. In *2011 International Conference on Computer Vision*. IEEE.

Wah, C., Horn, G. V., Branson, S., Maji, S., Perona, P., and Belongie, S. (2014). Similarity comparisons for interactive fine-grained categorization. In *Computer Vision and Pattern Recognition (CVPR)*, Columbus, OH.

Waldhauser, C., Hochreiter, R., Otepka, J., Pfeifer, N., Ghuffar, S., Korzeniowska, K., and Wagner, G. (2014). Automated classification of airborne laser scanning point clouds. In *Solving Computationally Expensive Engineering Problems*, pages 269–292. Springer International Publishing.

Waldner, F., Schucknecht, A., Lesiv, M., Gallego, J., See, L., Pérez-Hoyos, A., d'Andrimont, R., de Maet, T., Bayas, J. C. L., Fritz, S., Leo, O., Kerdiles, H., Díez, M., Van Tricht, K., Gilliams, S., Shelestov, A., Lavreniuk, M., Simões, M., Ferraz, R., Bellón, B., Bégué, A., Hazeu, G., Stonacek, V., Kolomaznik, J., Misurec, J., Verón, S. R., de Abelleyra, D., Plotnikov, D., Mingyong, L., Singha, M., Patil, P., Zhang, M., and Defourny, P. (2019). Conflation of expert and crowd reference data to validate global binary thematic maps. *Remote Sensing of Environment*, 221:235–246.

Walter, V., Kölle, M., Collmar, D., and Zhang, Y. (2021). A two-level approach for the crowd-based collection of vehicles from 3D point clouds. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, V-4-2021:97–104.

Walter, V., Kölle, M., and Yin, Y. (2020). Evaluation and Optimisation of Crowd-Based Collection of Trees from 3D Point Clouds. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, V-4-2020:49–56.

Walter, V., Kölle, M., and Collmar, D. (2022a). A gamification approach for the improvement of paid crowd-based labelling of geospatial data. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, V-4-2022:113–120.

Walter, V., Kölle, M., and Collmar, D. (2022b). Measuring the wisdom of the crowd: How many is enough? *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 90:269–291.

Walter, V., Laupheimer, D., and Fritsch, D. (2016). Use And Optimisation Of Paid Crowdsourcing For The Collection Of Geodata. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLI-B4:253–257.

Walter, V. and Soergel, U. (2018). Implementation, Results, and Problems of Paid Crowd-Based Geospatial Data Collection. *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 86:187–197.

Wang, J., Faridani, S., and Ipeirotis, P. (2011). Estimating the completion time of crowdsourced tasks using survival analysis models. *Crowdsourcing for search and data mining (CSDM 2011)*, 31.

Wang, P. and Yao, W. (2022). One class one click: Quasi scene-level weakly supervised point cloud semantic segmentation with active learning. *CoRR*, abs/arXiv.2211.12657.

Wattenberg, M., Viégas, F., and Johnson, I. (2016). How to use t-sne effectively. *Distill*.

Weinmann, M., Jutzi, B., Hinz, S., and Mallet, C. (2015a). Semantic point cloud interpretation based on optimal

neighborhoods, relevant features and efficient classifiers. *ISPRS Journal of Photogrammetry and Remote Sensing*, 105:286–304.

Weinmann, M., Jutzi, B., and Mallet, C. (2013). Feature relevance assessment for the semantic interpretation of 3D point cloud data. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-5/W2:313–318.

Weinmann, M., Jutzi, B., and Mallet, C. (2014). Semantic 3D scene interpretation: A framework combining optimal neighborhood size selection with relevant features. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-3:181–188.

Weinmann, M., Schmidt, A., Mallet, C., Hinz, S., Rottensteiner, F., and Jutzi, B. (2015b). Contextual classification of point cloud data by exploiting individual 3D neighborhoods. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-3/W4:271–278.

Weinmann, M., Urban, S., Hinz, S., Jutzi, B., and Mallet, C. (2015c). Distinctive 2D and 3D features for automated large-scale scene analysis in urban areas. *Computers & Graphics*, 49:47–57.

Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., and Perona, P. (2010a). Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology.

Welinder, P., Branson, S., Perona, P., and Belongie, S. (2010b). The multidimensional wisdom of crowds. In Lafferty, J., Williams, C., Shawe-Taylor, J., Zemel, R., and Culotta, A., editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc.

Welinder, P. and Perona, P. (2010). Online crowdsourcing: Rating annotators and obtaining cost-effective labels. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*. IEEE.

West, K. F., Webb, B. N., Lersch, J. R., Pothier, S., Triscari, J. M., and Iverson, A. E. (2004). Context-driven automated target detection in 3D data. In Sadjadi, F. A., editor, *Automatic Target Recognition XIV*, volume 5426, pages 133 – 143. International Society for Optics and Photonics, SPIE.

Whitehill, J., Wu, T.-f., Bergsma, J., Movellan, J., and Ruvolo, P. (2009). Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C., and Culotta, A., editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc.

Wilson, D. W., Lin, X., Longstreet, P., and Sarker, S. (2011). Web 2.0: A definition, literature review, and directions for future research. In *AMCIS 2011 - Proceedings*.

Wu, T.-H., Liu, Y.-C., Huang, Y.-K., Lee, H.-Y., Su, H.-T., Huang, P.-C., and Hsu, W. H. (2021). Redal: Region-based and diversity-aware active learning for point cloud semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15510–15519.

Xavier, E. M. A., Ariza-López, F. J., and Ureña-Cámara, M. A. (2016). A survey of measures and methods for matching geospatial vector datasets. *ACM Computing Surveys*, 49(2):1–34.

Xie, B., Yuan, L., Li, S., Liu, C. H., Cheng, X., and Wang, G. (2022). Active learning for domain adaptation: An energy-based approach. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(8):8708–8716.

Xu, S., Vosselman, G., and Elberink, S. O. (2014). Multiple-entity based classification of airborne laser scanning data in urban areas. *ISPRS Journal of Photogrammetry and Remote Sensing*, 88:1–15.

Ye, T., You, S., and Robert Jr., L. (2017). When does more money work? examining the role of perceived fairness in pay on the performance quality of crowdworkers. *Proceedings of the International AAAI Conference on Web and Social Media*, 11(1):327–336.

Ye, Z., Xu, Y., Huang, R., Tong, X., Li, X., Liu, X., Luan, K., Hoegner, L., and Stilla, U. (2020). LASDU: A large-scale aerial LiDAR dataset for semantic labeling in dense urban areas. *ISPRS International Journal of Geo-Information*, 9(7):450.

Zhang, C. and Chaudhuri, K. (2015). Active learning from weak and strong labelers. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 703–711, Cambridge, MA, USA. MIT Press.

Zhang, J., Wu, X., and Sheng, V. S. (2016). Learning from crowdsourced labeled data: a survey. *Artificial Intelligence Review*, 46(4):543–576.

Zhdanov, F. (2019). Diverse mini-batch Active Learning. *CoRR*, abs/1901.05954.

Zheng, Y., Li, G., Li, Y., Shan, C., and Cheng, R. (2017). Truth inference in crowdsourcing. *Proceedings of the VLDB*

Endowment, 10(5):541–552.

Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A. (2014). Learning deep features for scene recognition using places database. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Zhou, D., Platt, J. C., Basu, S., and Mao, Y. (2012). Learning from the wisdom of crowds by minimax entropy. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, page 2195–2203, Red Hook, NY, USA. Curran Associates Inc.

Zhou, F., Shui, C., Yang, S., Huang, B., Wang, B., and Chaib-draa, B. (2021). Discriminative active learning for domain adaptation. *Knowledge-Based Systems*, 222:106986.

Zhu, X. X., Tuia, D., Mou, L., Xia, G.-S., Zhang, L., Xu, F., and Fraundorfer, F. (2017). Deep learning in remote sensing: A comprehensive review and list of resources. *IEEE Geoscience and Remote Sensing Magazine*, 5(4):8–36.

Zolanvari, S. M. I., Ruano, S., Rana, A., Cummins, A., da Silva, R. E., Rahbar, M., and Smolic, A. (2019). Dublincity: Annotated lidar point cloud and its applications. *CoRR*, abs/1909.03613.

Appendix

A Methodology Supplements

An Intuition *why* AL Works

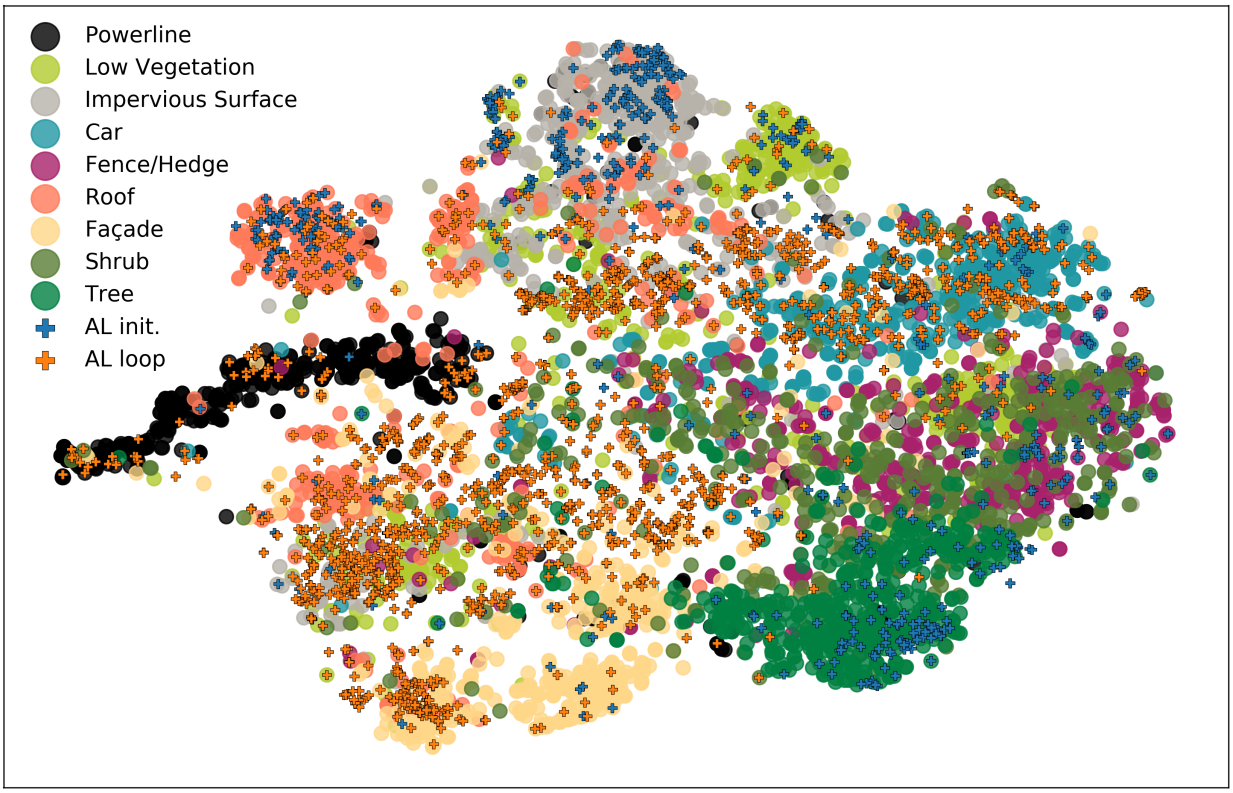
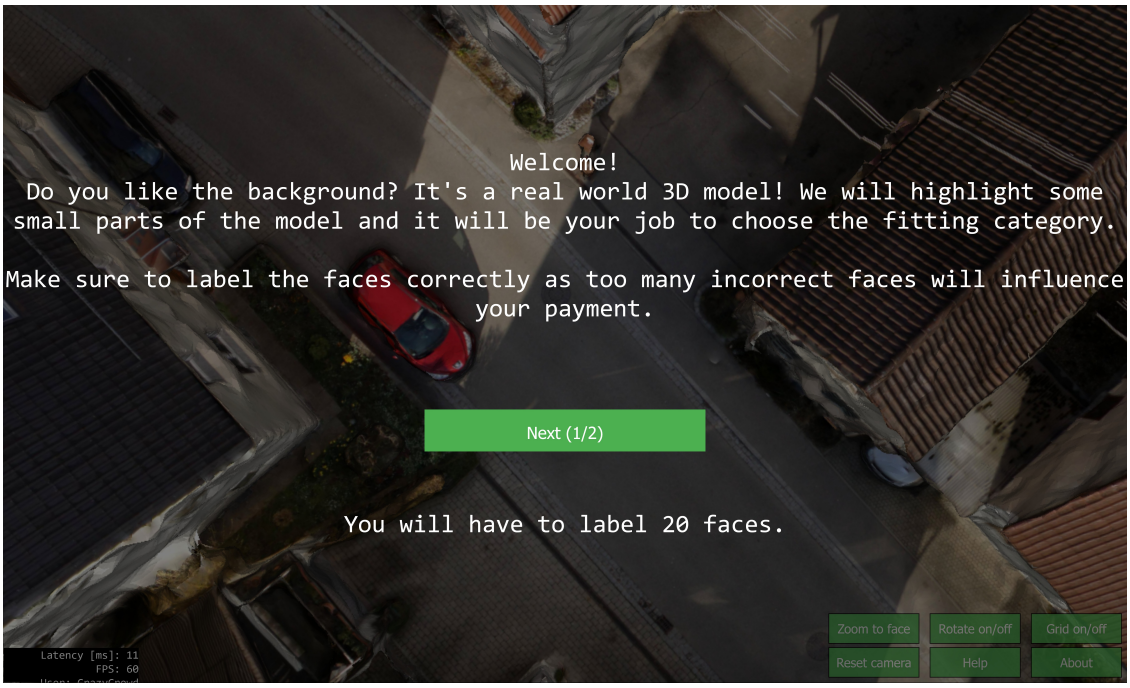


Figure A1: t-SNE embedding of feature vectors of V3D training data into 2D space along with depiction of AL points.

Designing Labeling Tools for the Crowd

Figure A2: A full snapshot of our labeling tool *Type A*.

Stimulate Motivation by Gamification

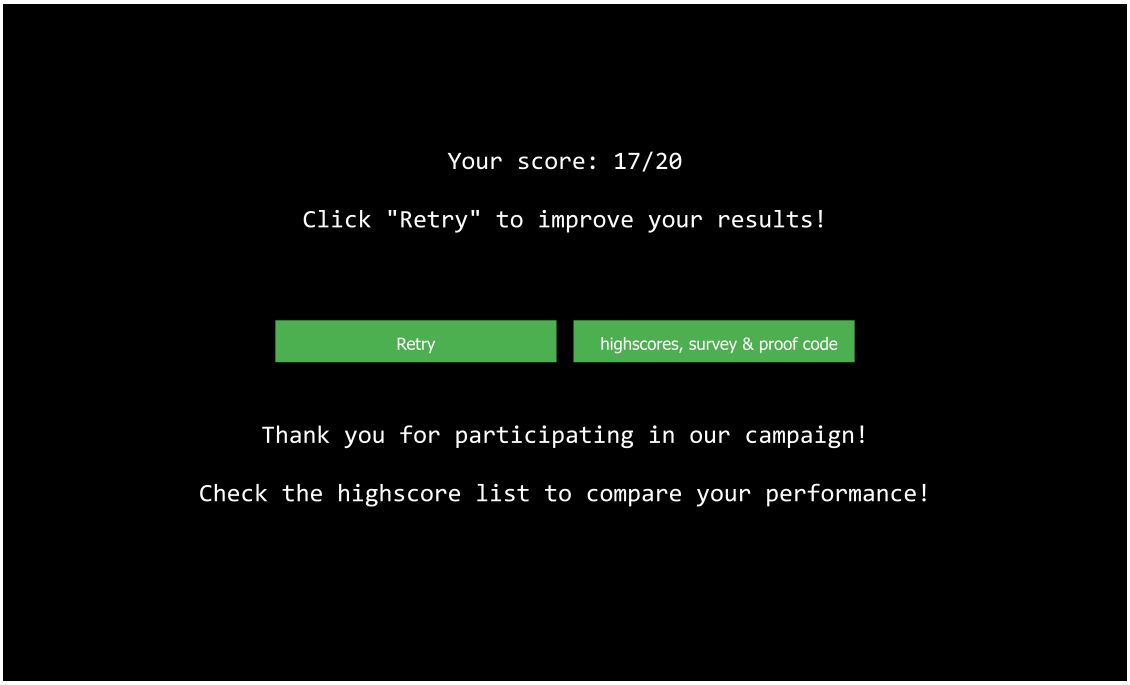


(a) Introduction page

(b) Main game interface

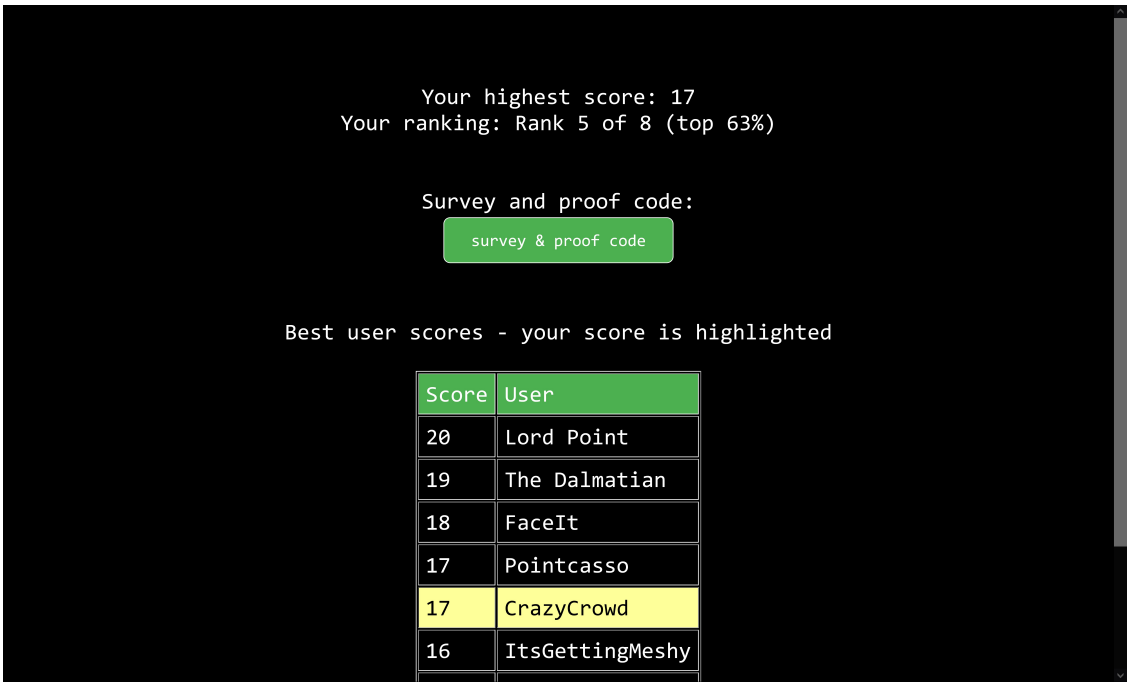
Figure A3: Snapshots of our labeling tool with a game-like appeal - enlargement of Figure 3.12.

(a) Progress bar (*top left*)



(b) Score

Figure A4: Overview of our implemented gamification elements - enlargement of Figure 3.13, Part 1.



(a) High score list

(b) Audiovisual effect

Figure A5: Overview of our implemented gamification elements - enlargement of Figure 3.13, Part 2.

Hand-Crafted Features & Random Forest

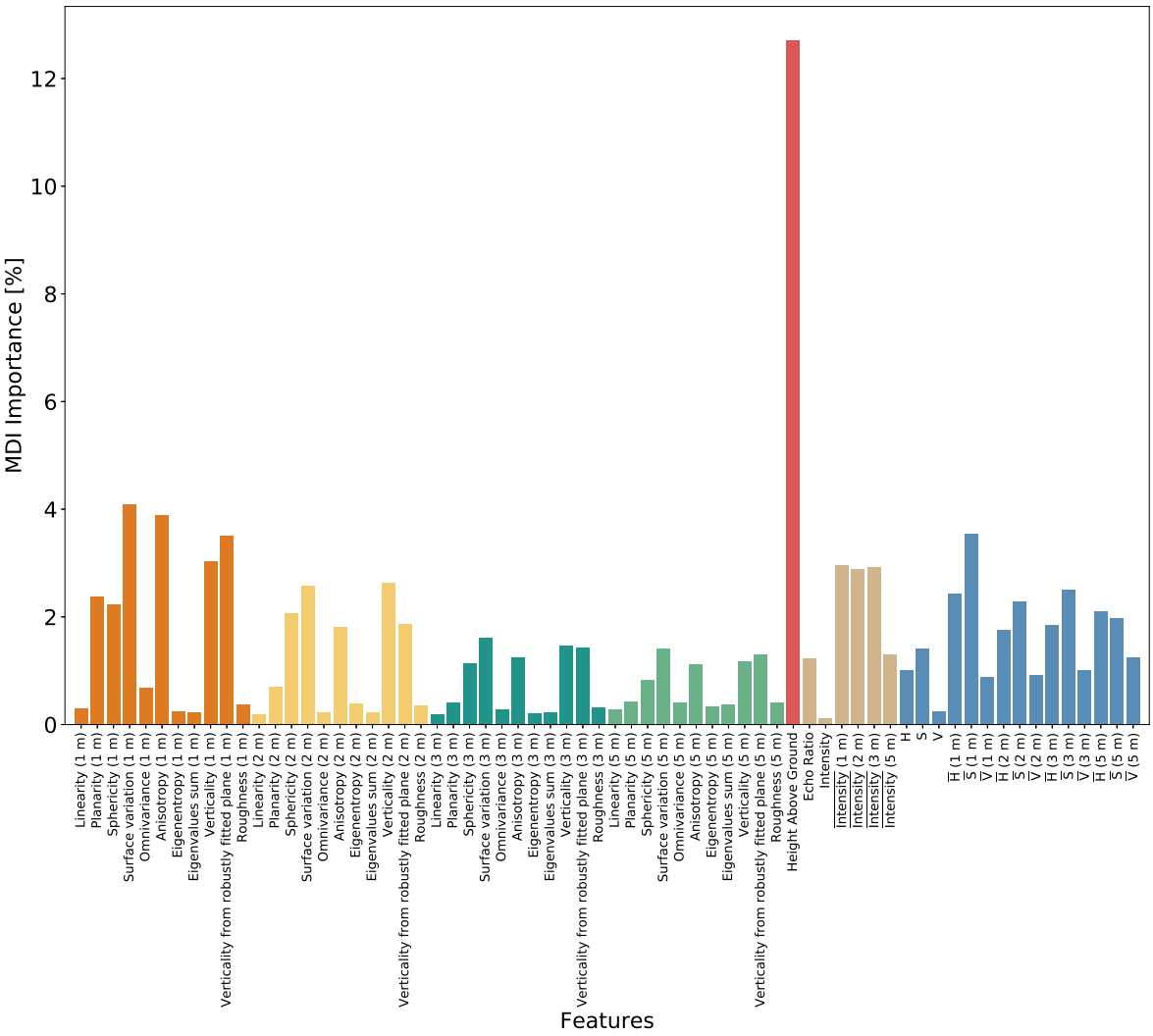


Figure A6: Assessment of the relative importance of the features presented in Table 3.1 measured by means of Mean Decrease in Impurity (MDI) for our RF classifier trained on the H3D data set, epoch March 2018 (Louppe, 2014). Numbers in *brackets* characterize the radius r of utilized spherical support volumes $\mathcal{N}$ to derive the respective feature. Importance bars are colored in accordance to the feature groups in Table 3.1.

B Experiments Supplements

Comparing Sampling Strategies

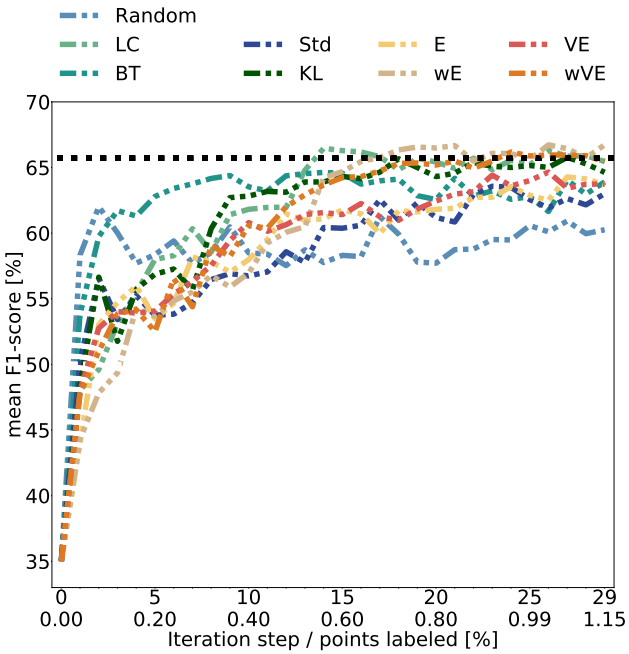


Figure B1: Comparison of different sampling strategies found in literature evaluated for the V3D data set in tandem with our RF classifier for $n_i = 30$ iteration steps with a batch size of $n^+ = 300$. We contrast the chosen *entropy* and *vote entropy* sampling as well as their weighted variants (wE & wVE) to other commonly used strategies such as *least certainty sampling* (LC), *breaking ties sampling* (BT) as well as epistemic disagreement measures such as the *average standard deviation* of ensemble predictions (Std) or the average *Kullback-Leibler divergence* (KL) (definitions can be found in the work of Settles, 2009).

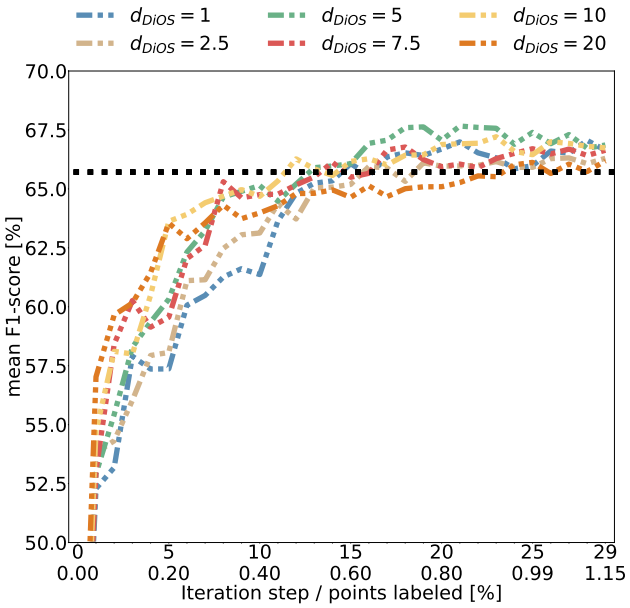


Figure B2: Comparison between different values for d_{DiOS}[m] for the V3D benchmark data set utilizing *weighted entropy* sampling in tandem with our RF classifier for $n_i = 30$ iteration steps and a batch size of $n^+ = 300$. Accordingly, we empirically selected $d_{DiOS} = 5\,\text{m}$ as it offers a good trade-off between a steep increase in early iteration steps and long-term accuracy. However, all $d_{DiOS} \geq 5\,\text{m}$ behave quite similar.

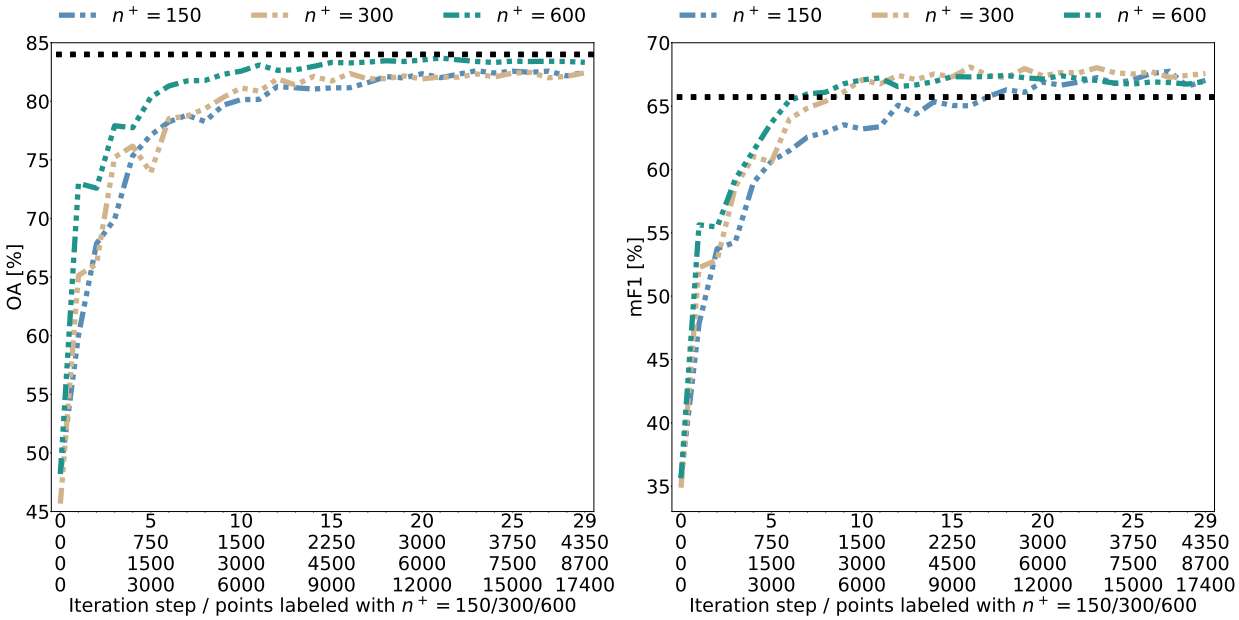


Figure B3: Comparison of different batch sizes in an AL loop utilizing *weighted entropy* sampling strategies in corporation with an RF classifier applied to the V3D data set. With a similar number of labeled samples, also similar accuracies can be achieved. For instance, consider the loop with batch size $n^+ = 150$ @ iteration step 20 with an mF1-score similar to the loop with batch size $n^+ = 300$ @ iteration step 10. Greater batch sizes, e.g., $n^+ = 600$ might help to reach a faster convergence of the AL loop and thus offers the chance to reduce the number of training cycles beneficial for computation-intensive training processes, e.g., see OA curves for the AL loop with $n^+ = 600$ @ iteration step 5 compared to the one with $n^+ = 300$ @ iteration step 10. With respect to the mF1-score, however, with large batch sizes, e.g. $n^+ = 600$, performance might be slightly harmed in later iteration steps since the classifier already focuses on fine-grained details that are only possible to be solved with frequent classifier updates.

Considering Imperfect Oracles

true label	L. Veg.	I. Surf.	Car	U. Furn.	Roof	Façade	Shrub	Tree	Gravel	Vert. Surf.	Chim.
confused with	Shrub	Gravel	U. Furn.	I. Surf.	Façade	Roof	Tree	Shrub	L. Veg.	Façade	Roof

Table B1: Assumed behavior of a confused oracle $\mathcal{O}_S$ regarding the H3D data set.

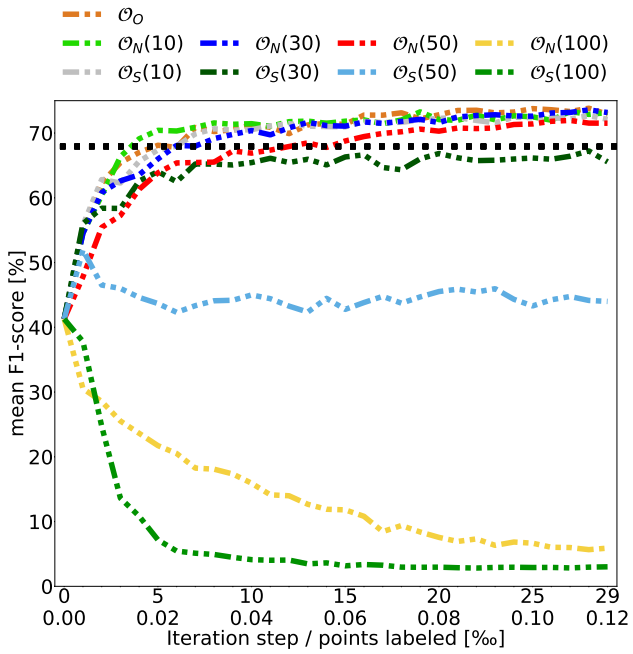


Figure B4: Comparison of different imperfect oracles utilizing an RF classifier with $wE + DiFS$ sampling for the H3D data set. Arguments in brackets represent the error rate σ [%]. Respective AL loops are conducted for $n_i = 30$ iteration steps utilizing a batch size of $n^+ = 300$. The assumed systematic mismapping of classes is given in Table B1.

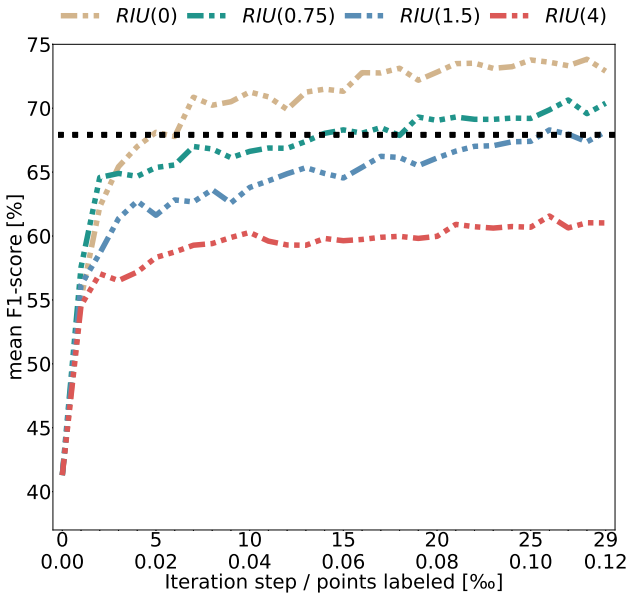


Figure B5: Comparison of different values for the maximum distance to the decision border d_{RIU}[m] utilizing an RF classifier with $wE + DiFS$ sampling for the H3D data set. Respective AL loops are conducted for $n_i = 30$ iteration steps utilizing a batch size of $n^+ = 300$.

Simulation of the Hybrid Intelligence System

Method	Sampl.	Oracle	F1-score				**mF1**	**OA**
			U. Furn.	Ground	Building	Tree		
PL			**75.30**	**98.63**	**96.82**	**93.97**	**91.18**	**95.51**
AL	*wE*	$\mathcal{O}_{GT}$	61.12	96.97	92.45	88.65	84.80	91.53
	wE+DiFS	$\mathcal{O}_{GT}$	62.17	98.23	94.93	91.62	86.74	93.81
	wE+DiFS+RIU	$\mathcal{O}_{GT}$	61.24	97.06	94.27	91.57	86.04	92.88
	wE+DiFS+RIU	$\mathcal{O}_{N}$	64.32	98.20	94.68	92.12	87.33	93.67

Table B2: Comparison of reachable accuracies [%] for different training approaches and simulated oracles using RF for the S3D data set after $n_i = 10$ iteration steps.

Employing the Crowd-Driven AL Loop for Efficient Segmentation of ALS Point Clouds

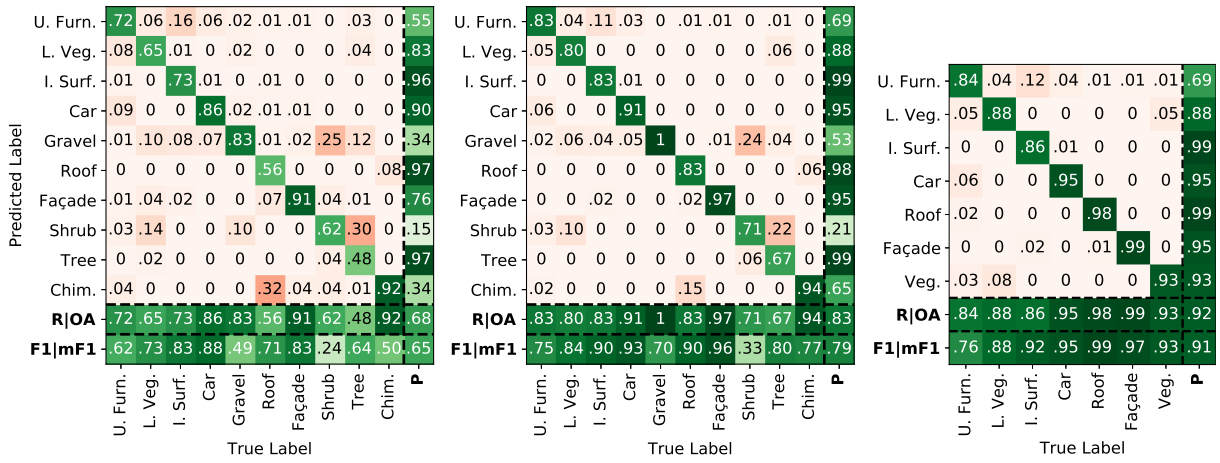


Figure B6: Labeling accuracy of the crowd in context of the initialization of an AL loop on the H3D data set. The accuracy of the raw and unchecked crowd labels *(a)* are verified by a second group of crowdworkers in a crowd campaign of *Type B* to obtain refined results *(b)*. To further ease the labeling complexity, we restrict the class catalog and merge classes accordingly *(c)*.

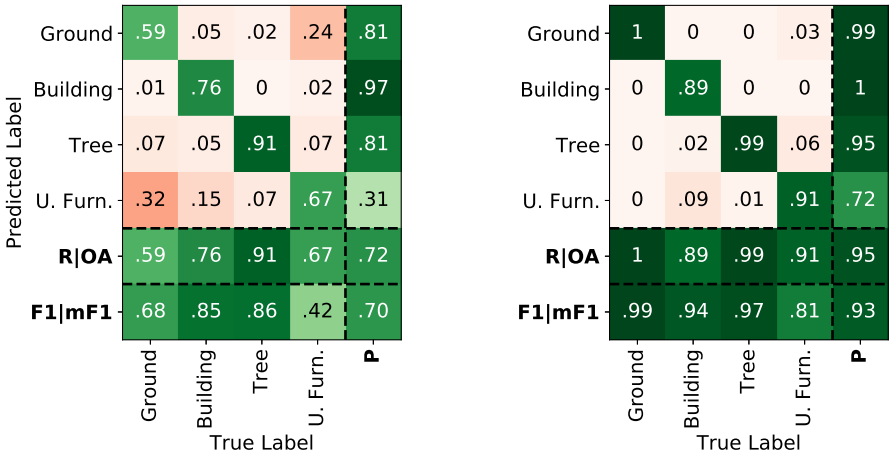


Figure B7: Labeling accuracy of the crowd in context of the initialization of an AL loop on the S3D data set. The accuracy of the raw and unchecked crowd labels *(a)* are verified by a second group of crowdworkers in a crowd campaign of Type B to obtain refined results *(b)*.

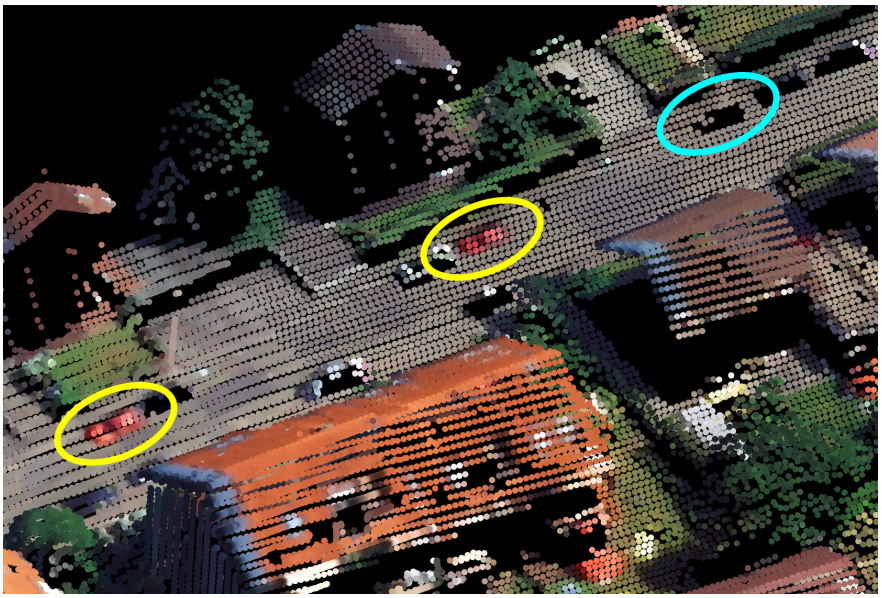


Figure B8: A closer view on the colorization of the V3D data set. Dynamic objects such as cars suffer from the projection of a temporally disjoint orthophoto. This results in effects where either street points are colored with car RGB tuples (*yellow*) or car points getting colored like a street (*cyan*).

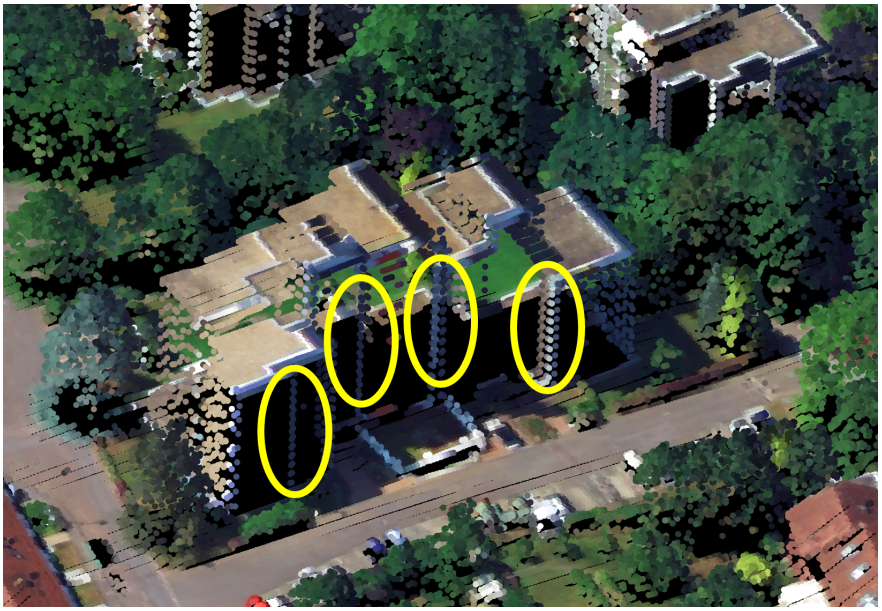
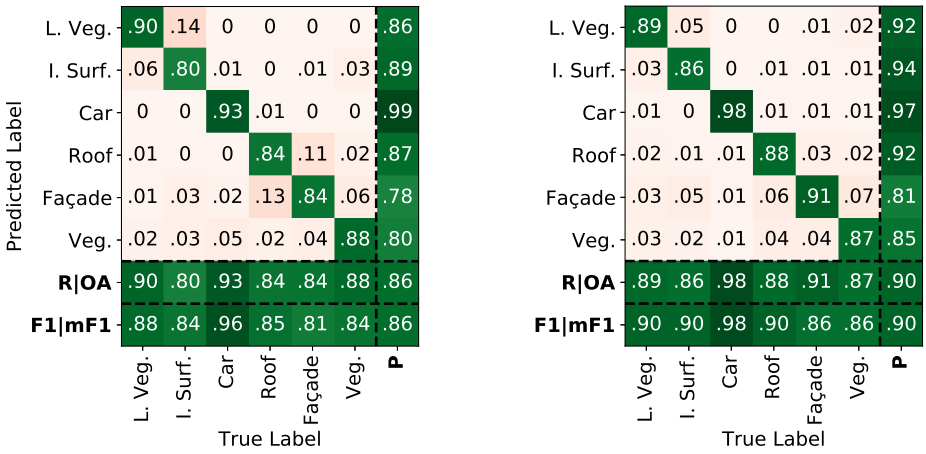


Figure B9: A close-up of the multi-family buildings in the V3D data set. Due to the narrow field of view and the flight planning (cf. Section 4.2.1), façades are often scarcely depicted, leading to frequent confusion of respective points with class *Powerline*.

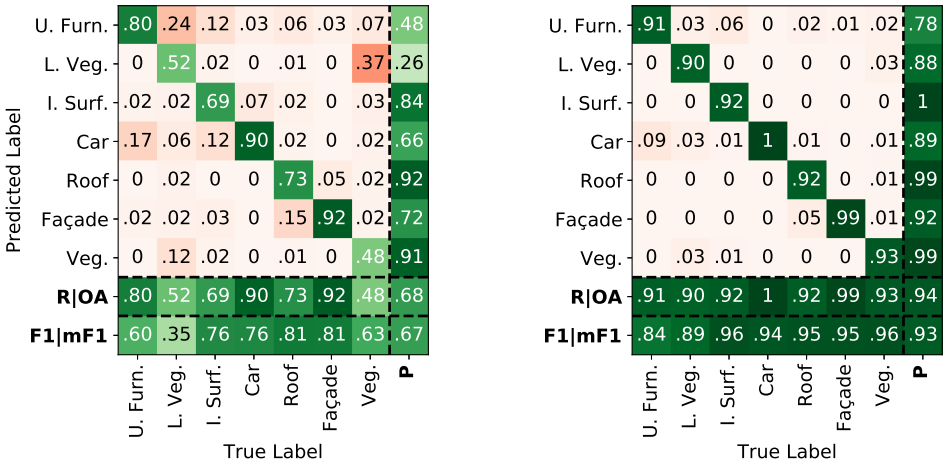


(a) $d_{RIU} = 0\,\mathrm{m}$ *(b)* $d_{RIU} = 1.5\,\mathrm{m}$

(c) $d_{RIU} = 0\,\mathrm{m}$ *(d)* $d_{RIU} = 1.5\,\mathrm{m}$

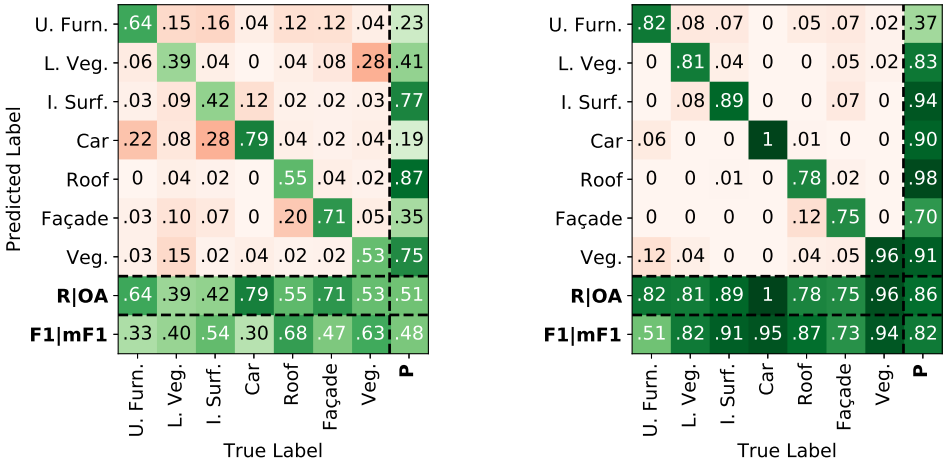
Figure B10: Obtained confusion matrices of crowd-generated labels computed from all labels set over the complete iteration for both the V3D (*top row*) and S3D data set (*bottom row*), and evaluating the impact of *RIU* for different maximum distance d_{RIU} values that are used by the *RIU* sampling add-on to select alternative points instead of those suggested by the machine.

Assessing the Long-Term Flexibility of ATL



(a) *(b)*

Figure B11: Labeling accuracy of the crowd in context of the initialization of an AL loop on the H3D data set, epoch November 2018. The accuracy of the raw and unchecked crowd labels *(a)* are verified by a second group of crowdworkers in a crowd campaign of *Type B (b)*. In total, 84.00 % of points remained after filtering.



(a) *(b)*

Figure B12: Labeling accuracy of the crowd in context of the initialization of an AL loop on the H3D data set, epoch March 2016. The accuracy of the raw and unchecked crowd labels *(a)* are verified by a second group of crowdworkers in a crowd campaign of *Type B (b)*. In total, 69.71 % of points remained after filtering.

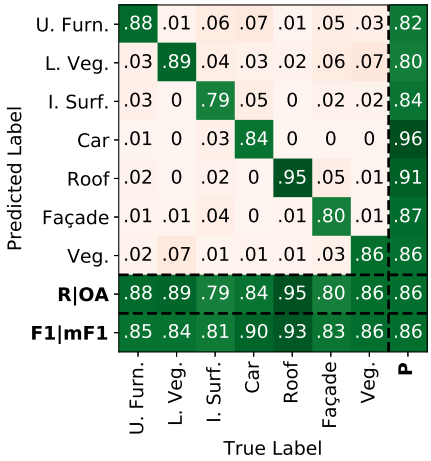


Figure B13: Obtained confusion matrix for crowd-generated labels over the complete ATL iteration for Scenario I.

Acknowledgments

Not only the unknown crowdworkers contributed to completing this work, but there is also another *crowd* standing behind me. First, this is Uwe and Norbert, who only gave me the opportunity to join the ifp in the first place and who constantly provided valuable feedback and new ideas for directions to dig into. This also applies to Volker, who confidently guided me through the entire development process of this work. Then, I would like to thank all my colleagues at the institute over the years, where pretty much everyone contributed in one way or another. Especially to be mentioned are Stefan, Dominik, David and Ivan, who got actively involved, provided some of their tools utilized within this work, and with whom respective publications were created. Also, all the sometimes professional and sometimes entertaining conversations with Pippo really enriched everyday work. A big thanks goes to Markus as well, who tirelessly meets all of the wishes for technical infrastructure. Furthermore, I would like to show my gratitude to Bernhard for his valuable suggestions in his role as co-referee as well as to Volker (Schwieger) for agreeing to act as chairman of the examination.

Finally, it only remains for me to thank my parents for pretty much everything. Additionally, there is the newest pack member Radi, our Husky, who never fails to create a relaxed environment and who is also a great fan of active learning - but in his own way.

Data acknowledgments:

The authors would like to show their gratitude to the State Office for Spatial Information and Land Development Baden-Wuerttemberg (Landesamt für Geoinformation und Landentwicklung Baden-Württemberg) for providing the ALS point clouds of the city of Stuttgart.

The Vaihingen data set was provided by the German Society for Photogrammetry, Remote Sensing and Geoinformation (DGPF) (Cramer, 2010).

Curriculum Vitae

Personal

Name	Michael Kölle
Date of Birth	September 11, 1994
Place of Birth	Geislingen an der Steige, Germany

Education

2018 - 2023	Doctoral Studies, Faculty for Aerospace Engineering and Geodesy, University of Stuttgart
2016 - 2018	Master of Science, Geodesy and Geoinformatics, University of Stuttgart
2013 - 2016	Bachelor of Science, Geodesy and Geoinformatics, University of Stuttgart
2005 - 2013	Abitur, Michelberg-Gymnasium, Geislingen an der Steige

Experience

11/2018 - current	Research Associate, Institute for Photogrammetry, University of Stuttgart
06/2018 - 08/2018	Research Assistant, Institute for Photogrammetry, University of Stuttgart