Masterarbeit

Analyse von Deep-Learning-Verfahren zur semantischen Segmentierung von photogrammetrischen Punktwolken aus Luftbildern

Eingereicht von: Herr Markus Hülsen (B.Sc.)
Barkweg 3, 21762 Otterndorf
E-Mail: markus.huelsen@student.jade-hs.de
Matr.-Nr. 6026370

Studiengang: Geoinformationswissenschaften (M.Sc.)

Abgegeben am: 25.08.2023

Erstprüfender: Prof. Dr.-Ing. habil. Dr. h.c. Thomas Luhmann
Jade Hochschule Oldenburg
Ofener Straße 16/19, 26121 Oldenburg
Telefon: 0441 - 7708 – 3172
E-Mail: luhmann@jade-hs.de

Zweitprüfende: Frau Valentina Schmidt (M.Sc.)
Landesamt für Geoinformation und Landesvermessung Niedersachsen (LGLN)
Podbielskistraße 331, 30659 Hannover
E-Mail: valentina.schmidt@lgln.niedersachsen.de

Zusammenfassung

Die semantische Segmentierung von topographischen Punktwolken ist nach wie vor eine anspruchsvolle Aufgabe. Insbesondere für photogrammetrische Punktwolken, die aus Luftbildern erzeugt werden, wurden bisher kaum wissenschaftliche Untersuchungen durchgeführt. Im Rahmen dieser Arbeit wird das Potential von Deep-Learning-Verfahren zur semantischen Segmentierung von photogrammetrischen Punktwolken aus Luftbildern näher analysiert und ein Arbeitsablauf zur Erstellung eines solchen Modells aufgezeigt. In dieser Untersuchung wird zunächst eine umfangreiche Punktwolke für das Training eines Deep-Learning-Modells mit unüberwachten Segmentierungsverfahren annotiert. Der gewählte Ansatz ist auf andere topographische Punktwolken übertragbar und stellt einen angemessenen Kompromiss zwischen Zeitaufwand und Nutzen dar. Die annotierten Punktwolken decken einen heterogenen urbanen Raum ab und umfassen insgesamt 12 km² und 66,8 Millionen Punkte. Die Punktwolken sind in fünf Klassen unterteilt: Boden, Gebäude, Vegetation, menschgemacht und Brücken.

Die Deep-Learning-Architektur PointNet++ wird als Klassifikationsmodell angewendet und die optimalen Trainingsparameter werden anhand verschiedener Experimente bestimmt. Zusätzlich werden verschiedene Methoden zur Dataset-Augmentation und unterschiedliche Batchgrößen getestet, um eine optimale Performance zu erreichen. Es zeigt sich, dass mit dem gewählten Ansatz eine Validierungsgenauigkeit von 96,5% erreicht wird. Zusätzlich zur Klassifikation wird für jeden Punkt eine Wahrscheinlichkeit der Klassenzugehörigkeit berechnet, sodass eine Filterung für ungenaue Klassenzuordnungen möglich ist. Die hier berechneten Modelle bieten eine geeignete Grundlage für weitere Punktwolkenverarbeitungen wie die Ableitung von digitalen Gelände- und Oberflächenmodellen oder die Erstellung von Gebäudemodellen.

Abstract

The semantic segmentation of topographic point clouds is still a challenging task. Especially for photogrammetric point clouds generated from aerial images, hardly any scientific investigations have been carried out so far. In the context of this work, the potential of Deep Learning methods for semantic segmentation of photogrammetric point clouds from aerial images is analyzed in more detail and a workflow for creating such a model is shown. In this study, an extensive point cloud is first annotated for training a Deep Learning model using unsupervised segmentation techniques. The chosen approach is transferable to other topographic point clouds and represents a good compromise between time and benefit. The annotated point clouds cover a heterogeneous urban space and include a total of 12 km² and 66.8 million points. The point clouds are divided into five classes: Ground, Buildings, Vegetation, humanmade and Bridges.

The deep learning architecture PointNet++ is applied as a classification model and the optimal training parameters are determined based on various experiments. In addition, different methods for dataset augmentation and different batch sizes are tested to achieve optimal performance. It is shown that a validation accuracy of 96.5% is achieved with the chosen approach. In addition to classification, a probability of class membership is calculated for each point, allowing filtering for inaccurate class assignments. The models calculated here provide a suitable basis for further point cloud processing such as the derivation of digital terrain and surface models or the creation of building models.

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

Formelverzeichnis

1. Einleitung

Punktwolken haben sich im Fachbereich der Geodäsie zu einem der Standardprodukte etabliert. Dabei können Punktwolken sowohl durch Laserscanning als auch durch photogrammetrische Verfahren erhoben werden. Die Erhebung dieser Punktwolken ist ausgereift und weit erforscht. Die effiziente Weiterverarbeitung der Punktwolken in verschiedenen Anwendungsbereichen ist jedoch noch Gegenstand intensiver Forschung. Insbesondere die semantische Segmentierung von Punktwolken gewinnt in diversen Anwendungsfeldern zunehmend an Bedeutung, wie z. B. bei der Ableitung von Gebäudemodellen, der Berechnung von digitalen Gelände- und Oberflächenmodellen, sowie bei Veränderungsanalysen. Die automatisierte Segmentierung von Punktwolken in semantische Objekte stellt nach wie vor eine anspruchsvolle Herausforderung dar. Die manuelle Klassifikation ist äußerst zeitaufwendig und auf Geometrie basierende Ansätze zeigen Verbesserungspotential hinsichtlich ihrer Performance. Gängige Machine-Learning-Algorithmen stoßen bei der semantischen Segmentierung von Punktwolken an ihre Leistungsgrenzen, da die Auswahl und Berechnung von Nachbarschaften und Merkmalen individuell erfolgt.

In den vergangenen Jahren haben verschiedene Deep-Learning-Verfahren, insbesondere Convolutional-Neural-Networks (CNNs) für die Klassifikation und Objektsegmentierung im Bereich der Computer Vision, vielversprechende Fortschritte erzielt. Ebenso haben sich vielversprechende Deep-Learning-Verfahren in der Punktwolkenverarbeitung etabliert. Ein Großteil der aktuellen Forschung (bspw. WINIWARTER & MANDLBURGER, G. (2019) oder KADA & KURAMIN (2021)) fokussiert sich auf die Verarbeitung von Punktwolken aus dem Airborne Laserscanning. Im Gegensatz dazu gibt es nur wenige Veröffentlichungen, die die Methoden anhand von photogrammetrischen Punktwolken aus Luftbildern evaluieren.

Im Rahmen dieser Arbeit sollen Deep-Learning-Modelle zur semantischen Segmentierung von Punktwolken vorgestellt und der gesamte Workflow zur Erstellung und zum Training eines Modells beschrieben sowie evaluiert werden. Dies umfasst einerseits die Auswahl und Annotation einer geeigneten Punktwolke, die als Trainingsdatensatz dient, und andererseits die Anwendung und das Training eines geeigneten Deep-Learning-Modells. Die Annotation eines Trainingsdatensatzes ist ein entscheidender Schritt, der über die Qualität der Ergebnisse entscheidet. Die Menge und Qualität der Daten sind dabei entscheidend, sodass ein effizientes Vorgehen notwendig ist. Auch die Auswahl eines geeigneten Modells und der entsprechenden Parameter ist nicht trivial. Im Folgenden soll ein geeignetes Modell ausgewählt werden und anschließend anhand verschiedener Experimente die optimalen Parameter bestimmt werden, um eine bestmögliche Performance zu ermöglichen. Zur Evaluierung des Modells werden neben quantitativen Bewertungskriterien auch qualitative Analysen und visuelle Bewertungen durchgeführt.

2. Theoretische Grundlagen

Vor der detaillierten Darstellung verschiedener Deep-Learning-Architekturen zur semantischen Segmentierung von Punktwolken ist es notwendig, ein grundlegendes Verständnis von Deep Learning im Allgemeinen zu schaffen. Darüber hinaus werden im Folgenden die charakteristischen Eigenschaften der photogrammetrischen Datengrundlage vermittelt. Zunächst wird die Erhebung photogrammetrischer Punktwolken aus Luftbildern detailliert erläutert, gefolgt von einer Darstellung der fundamentalen Grundlagen und Methodiken des Deep Learning. Anschließend werden die theoretischen Grundlagen von Punktwolken, unter besonderer Berücksichtigung der Punktwolkenverarbeitung erörtert. Dies umfasst einerseits unüberwachte Verfahren, die ohne die Verwendung von Trainingsdaten auskommen, und andererseits überwachte Verfahren, die auf Standard-Machine-Learning-Klassifikatoren basieren. Ein weiterer Schwerpunkt liegt zudem in der Beschreibung der Deep-Learning-Methoden zur semantischen Segmentierung von Punktwolken.

2.1. Photogrammetrische Erhebung von Punktwolken

Die Generierung von Punktwolken aus Luftbildern stellt eine bedeutende Technik in der photogrammetrischen Datenerfassung dar und wird häufig durch den kombinierten Einsatz von *Structure-from-Motion* (SfM) und *Dense-Image-Matching* (DIM) realisiert. SfM ist ein Verfahren, das nicht signalisierte Luftbilder orientiert, eine Simultankalibrierung durch Bündelblockausgleichung schätzt und durch Stereobildmessung und Punktwolkenverarbeitung (z. B. DIM) eine dichte Punktwolke erzeugt. Dabei werden charakteristische Merkmale durch einen Interest-Operator (oft *Scale Invariant Feature Transform*, SIFT) in den Bildern erkannt und homologe Punkte in verschiedenen Bildern zugeordnet. Dies ermöglicht die Berechnung von 3D-Punktkoordinaten und die Generierung einer dünnen Punktwolke. Durch vorhandene Näherungswerte werden alle Bilder in einer gemeinsamen Bündelausgleichung orientiert – die so ermittelten Objektkoordinaten der charakteristischen Punkte bilden hier bereits eine (dünne) Punktwolke (LUHMANN 2018:492). Zur Ableitung einer dichten Punktwolke wird beim *Dense Image Matching* z. B. mittels Semi-Global Matching (SGM) (HIRSCHMÜLLER 2005) versucht, für jedes Pixel eines Bildes das korrespondierende Pixel in allen anderen Bildern zu identifizieren. Mit Hilfe der korrespondierenden Pixel können die Korrespondenzen bestimmt und im Idealfall für jedes Pixel eine dreidimensionale Koordinate berechnet werden. Das SGM stellt dabei eine Stereo-Matching-Methode dar, mit einer approximierten globalen Glattheitsbedingung, die zu homogenen Oberflächen führt. Darüber hinaus ermöglicht es die Auflösung von Mehrdeutigkeiten bei simultaner Reduktion des Rauschens und der Interpolation kleiner, schwach texturierter Bereiche (WENZEL, K. 2016:32). Die Korrespondenzen können auch als Disparitäten (auch Parallaxen genannt) betrachtet werden. Diese repräsentieren die Verschiebungen, die sich aus Tiefenvariationen ergeben, welche aus verschiedenen Ansichten beobachtet werden. Die Disparitäten sind proportional zur Tiefe selbst und können zur Tiefenrekonstruktion verwendet werden (WENZEL, K.

2016:10). Dazu wird ein Bild als das Basisbild ausgewählt, um aus den *N* überlappenden Bildern die Disparitätsbilder zu berechnen. Für jedes Pixel können so bis zu *N* Disparitätswerte entstehen, die mit der Basislänge gewichtet zu einem dichten Disparitätsbild mit reduzierter Anzahl an Ausreißern zusammengefasst werden (HIRSCHMÜLLER 2017:254). Die Disparitätsinformationen können zur Bestimmung von dreidimensionalen Objektkoordinaten mittels Multi-Stereo-Triangulation verwendet werden.

Zu den größten Herausforderungen beim Dense Image Matching gehören Mehrdeutigkeiten, schwache Texturen, radiometrische und projektive Unterschiede, Diskontinuitäten, bewegte Objekte, Tiefenvariationen und Skalierbarkeit. Mehrdeutigkeiten treten auf, wenn Texturmuster häufiger im Bild vorkommen, sodass sie nicht zur eindeutigen Identifikation verwendet werden können. Treten in den Bildern nur schwach texturierte Oberflächen auf, können auch in benachbarten Bildern keine homologen Bereiche identifiziert werden. Radiometrische und projektive Unterschiede erschweren die Zuordnung ebenfalls. Um den Einfluss von Tiefenvariationen zu minimieren, werden Glattheitsterme eingeführt, die scharfe Kanten abrunden (WENZEL, K. 2016:10). Dies unterscheidet die DIM-Punktwolken von den Punktwolken aus dem Airborne Laserscanning (ALS), bei dem die Punkte polar gemessen werden. Außerdem erschweren Verdeckungen die Rekonstruktion von 3D-Punkten, da Punkte in mehreren Bildern gemessen werden müssen. Die Unterschiede der Erhebungsverfahren sind in Abbildung 1 schematisch dargestellt.

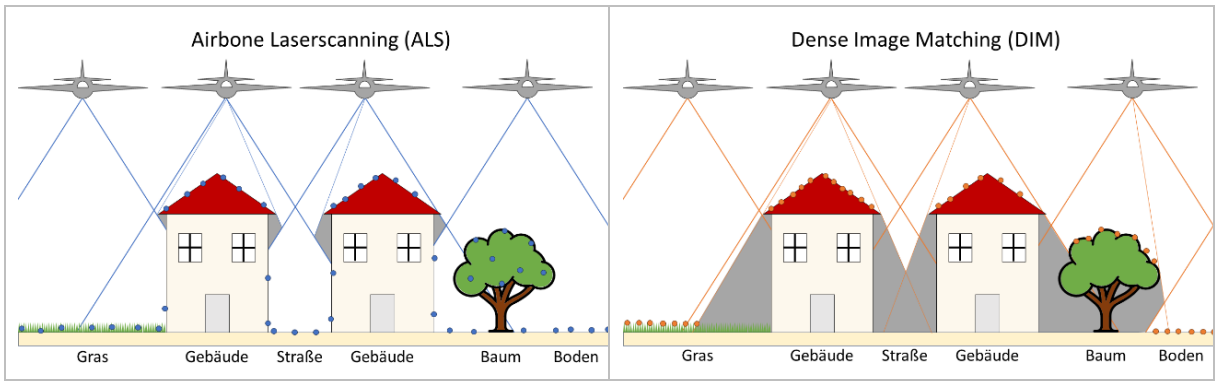


Abbildung 1: Schematische Darstellung der Erhebung mit ALS (links) und DIM (rechts) (MANDLBURGER, G. et al. 2017:2)

2.2. Deep Learning

Ein Teilgebiet der künstlichen Intelligenz ist das maschinelle Lernen (*Machine Learning*, ML), bei dem es um die Entwicklung von Algorithmen und Modellen geht, die es Computern ermöglichen, komplexe Aufgaben zu erlernen und auszuführen. Dabei kann Machine Learning grundsätzlich als die Wissenschaft der Programmierung von Systemen bezeichnet werden, die aus Daten lernen können, ohne dass das Wissen explizit programmiert werden muss (GÉRON 2019:2). Ein Lernprozess entsteht dann, wenn die einstellbaren Parameter des Modells an die beobachteten Daten angepasst werden können (VANDERPLAS 2018:394). Innerhalb des Machine Learning nimmt das *Deep Learning* (DL) eine

wichtige Rolle ein und basiert auf künstlichen neuronalen Netzen mit mehreren Schichten, die auch als *Deep Neural Networks* bezeichnet werden (LECUN et al. 2015). Die Unterschiede zwischen der Pipeline eines herkömmlichen Algorithmus, der Pipeline eines ML-Algorithmus und der Pipeline eines Deep Learning-Algorithmus sind in Abbildung 2 dargestellt.

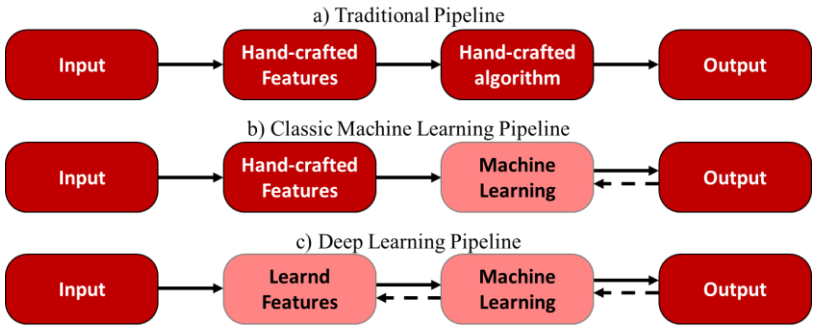


Abbildung 2: Traditionelle, Machine Learning und Deep Learning Pipeline (SZELISKI 2022:238)

Im Gegensatz zu herkömmlichen Programmen, die sowohl individuell programmierte Algorithmen als auch individuell programmierte Merkmale verwenden, lernt das maschinelle Lernen selbstständig aus den Daten und passt die Parameter an. Deep Learning vermeidet auch die Notwendigkeit, einzelne Merkmale auszuwählen und zu berechnen, indem eine hierarchische Repräsentation der Informationen erzeugt wird, die es ermöglicht, abstrakte Merkmale und Muster in den Daten zu erkennen und zu lernen.

2.2.1. Multilayer Perceptron

Das Fundament eines Deep-Learning-Modells ist das *Feedforward-Deep-Network*, bzw. das *Multilayer-Perceptron* (MLP). Das MLP ist eine mathematische Funktion, die sich aus vielen einfachen Funktionen zusammensetzt. Die Anwendung jeder dieser einfachen Funktionen beschreibt eine neue Repräsentation der Eingabewerte (GOODFELLOW et al. 2016:163). Wie in Abbildung 2 dargestellt, verwenden Deep-Learning-Modelle einen End-to-End-Ansatz: Nach jedem Verarbeitungsschritt werden die Parameter optimiert, um den Trainingsverlust zu minimieren. Eine wichtige Voraussetzung für die Optimierung ist die Differenzierbarkeit der Verlustfunktion. MLPs bestehen aus zahlreichen miteinander verbundenen Neuronen (Einheiten), die gewichtete Summen ihrer Eingaben bilden und diese über eine Aktivierungsfunktion weiterleiten. Die Gewichte und der Bias jedes Neurons x sind lernfähige Parameter, welche während des Lernprozesses optimiert werden. In neueren Netzwerken wird häufig die *ReLU*-Aktivierungsfunktion *h (Rectified Linear Units)* verwendet, die in Formel 1 definiert ist als:

Formel 1: Rectified Linear Units (ReLU)

$$h(y) = \max(0, y)$$

Ein sogenannter *Layer* besteht aus einer festgelegten Anzahl von Neuronen. Die Kombination verschiedener Layer mit spezifischen Anzahlen an Neuronen bildet ein neuronales Netz, wie in Abbildung 3 dargestellt. Alle Eingangswerte des Netzes werden auf alle Neuronen des ersten Layers

übertragen. Die Ausgabewerte jedes Layers, die häufig als *Aktivierungen* (engl. *Activations*) bezeichnet werden, dienen als Eingabewerte für den nachfolgenden Layer. Der letzte Layer gibt schließlich die Ausgabewerte zurück.

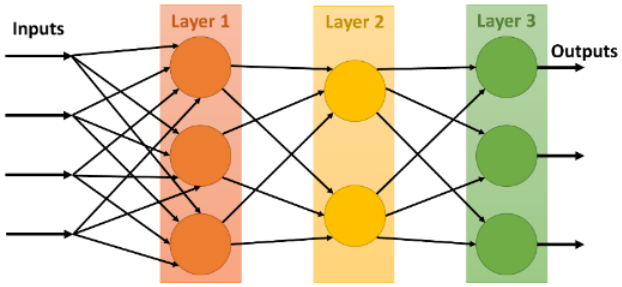


Abbildung 3: Visualisierung eines Multilayer Perceptron (SZELISKI 2022:272)

2.2.2. Backpropagation

Als *Forward-Propagation* wird der Prozess bezeichnet, bei dem ein Datensatz x in eine MLP überführt wird, um eine Ausgabe $\hat{y}$ zu erzeugen. Die anschließende Berechnung einer Verlustfunktion liefert eine Bewertung der Modellierungsleistung. Der *Back-Propagation*-Algorithmus (RUMELHART et al. 1986) ermöglicht es, die Information der Verlustfunktion rückwärts durch das Netz zu übertragen, um die Gradienten der Verlustfunktion in Abhängigkeit von den Parameter zu berechnen. Anschließend verwenden Optimierungsalgorithmen wie *Stochastic Gradient Descent (SGD)* oder *Adaptive Moment Estimation (Adam)* diese Gradienten, um die Parameter des Netzes zu optimieren und die Verlustfunktion zu minimieren. Dieser Prozess wird iterativ wiederholt, wobei die Parameter der Neuronen sukzessive modifiziert werden, bis das Netz konvergiert (GOODFELLOW et al. 2016:197f.).

2.2.3. Verlustfunktionen

Um die Gewichte in einem neuronalen Netz zu optimieren, ist die Definition einer *Verlustfunktion* (*Loss-Function*, *Loss*) notwendig, welche anhand der Trainingsdaten minimiert wird. Die Verlustfunktion dient dazu, die prädizierte Ausgabe mit der Ground Truth zu vergleichen und Abweichungen zu bestrafen. Für Klassifikationsaufgaben wird häufig der *Cross-Entropy-Loss* (CEL) verwendet (SZELISKI 2022:280), der abhängig von der Wahrscheinlichkeit der Ground-Truth p_t den Verlust nach Formel 2 berechnet.

Formel 2: Cross-Entropy-Loss

$$CEL(p_t) = -\log(p_t)$$

Formel 3: Focal Loss

$$FL(p_t) = -(1 - p_t)^{\gamma} \cdot \log(p_t)$$

Es ist jedoch wichtig zu beachten, dass die Wahl der Verlustfunktion von der spezifischen Aufgabe des Netzes und den charakteristischen Merkmalen der Daten abhängt. Für stark unbalancierte Segmentierungsaufgaben können fokusbasierte Verlustfunktionen eine bessere Wahl sein (JADON

2020). Ein Beispiel hierfür ist der *Focal Loss* (LIN, T.-Y. et al. 2017), der nach Formel 3 berechnet wird und dem CEL einen zusätzlichen Faktor hinzufügt. Dieser Faktor ist hängig von einem Parameter γ ab und reduziert den relativen Verlust für gut klassifizierte Daten, bei denen die Wahrscheinlichkeit der Ground Truth $> 0{,}5$ ist, und legt so einen Fokus auf schwer zu klassifizierende Datensätze.

2.2.4. Trainingsdaten

Insbesondere für Deep-Learning-Modelle ist eine große Menge an Trainingsdaten erforderlich, um das Modell effizient zu trainieren. Darüber hinaus sollten sie eine große Vielfalt an Beispielen abdecken, die den gesamten Anwendungsbereich repräsentieren. Je mehr Trainingsdaten zur Verfügung stehen, desto besser kann das Modell generalisieren (MUMUNI, A. & MUMUNI, F. 2022:1). Dabei ist nicht nur die Menge und die Korrektheit der Daten entscheidend, sondern auch deren Heterogenität. Die Trainingsdaten sollten die verschiedenen Variationen und Randfälle enthalten, die in der realen Anwendung auftreten können. Es ist wichtig sicherzustellen, dass die Daten die gesamte Bandbreite möglicher Eingaben abdecken, um das Modell auf alle relevanten Szenarien vorzubereiten. Die Qualität der Ground Truth ist ebenfalls von elementarer Bedeutung. Es muss sichergestellt werden, dass die Labels korrekt und konsistent vergeben werden. Zudem führt eine starke Heterogenität innerhalb einer semantischen Klasse zu einer geringeren Performance. Die Qualität der Annotation hat einen erheblichen Einfluss auf die Leistungsfähigkeit eines Modells, da dieses auf Grundlage dieser Daten trainiert wird (BUDACH et al. 2022).

2.2.5. Regularisierung und Normalisierung

Eines der Hauptprobleme des maschinellen Lernens besteht darin, ein Modell zu erstellen, das auch bei ungesehenen Daten gute Leistung erbringt, was als Generalisierung bezeichnet wird. Bei der Beurteilung dieser Frage ist es wichtig nicht nur den Verlust während des Trainings (Trainingsverlust) zu berücksichtigen, sondern auch den Verlust auf Validierungsdaten (Validierungsverlust), die nicht für das Training verwendet werden. Die Leistung eines Algorithmus des Machine Learning wird von seiner Fähigkeit beeinflusst, (1) den Trainingsverlust zu minimieren und (2) die Differenz zwischen Trainings- und Validierungsverlust gering zu halten. Diese Faktoren korrelieren mit den zentralen Herausforderungen der *Unteranpassung (underfitting)* und der *Überanpassung (overfitting)*. Unteranpassung tritt auf, wenn das Modell nicht in der Lage ist, einen minimalen Trainingsverlust zu erreichen. Überanpassung tritt auf, wenn die Differenz zwischen Trainingsverlust und Validierungsverlust zu groß wird. Um eine Überanpassung zu vermeiden, werden verschiedene Regularisierungs- und andere Techniken verwendet, um sicherzustellen, dass das Netz gut auf ungesehene Daten generalisiert. Dazu werden beispielsweise Trainingsdaten in Gruppen (sogenannte Batches) zusammengefasst, die gleichzeitig durch das neuronale Netz propagiert werden. Dies

verbessert die Effizienz und die Auswirkungen von Rauschen in den Daten werden gemittelt, was stabilere Gewichtsaktualisierungen ermöglicht (GOODFELLOW et al. 2016:267). Eine weitere Technik ist der *Dropout* (SRIVASTAVA et al. 2014), bei dem während des Trainings in jedem Batch zufällig eine bestimmte Anzahl (z. B. 50%) der Neuronen deaktiviert wird. Dies simuliert ein Rauschen im Netz und hilft, die Abhängigkeiten zwischen den Neuronen zu reduzieren (SZELISKI 2022:276). Solche Techniken tragen dazu bei, nicht nur den Trainingsverlust zu reduzieren, sondern auch den Validierungsverlust gering zu halten (GOODFELLOW et al. 2016:107f.).

Der beste Weg, die Generalisierungsfähigkeit eines Modells zu verbessern, besteht darin, es auf größeren Datenmengen zu trainieren. Die Idee der *Dataset Augmentation* (DA) besteht darin, zusätzliche Trainingsdaten zu erzeugen, indem die Eingaben und/oder Ausgaben der vorhandenen Datensätze in realistischer Weise verändert werden. Dies kann insbesondere für Klassifikationsaufgaben vorteilhaft sein, da häufig nur eine begrenzte Anzahl von Trainingsdatensätzen zur Verfügung steht und das Label auch bei geringfügigen Änderungen des Eingabedatensatzes unverändert bleibt. Beispielsweise können neue Trainingsdatensätze durch geometrische Transformationen, wie Rotation, oder radiometrische Transformationen, wie Änderung der Farbsättigung erzeugt werden (SZELISKI 2022). Diese Transformationen müssen jedoch sorgfältig ausgewählt werden. In einigen Fällen, wie bei der Klassifizierung von Zahlen und Buchstaben, kann eine vertikale oder horizontale Spiegelung dazu führen, dass aus einem „b“ ein „d“ oder aus einer „6“ eine „9“ wird (GOODFELLOW et al. 2016). Daher müssen die angewendeten Dataset-Augmentation-Techniken darauf abzielen, die Datenvielfalt zu erhöhen, ohne die grundlegende Klassifikationseigenschaft der Eingabedaten zu beeinträchtigen.

Insbesondere bei dreidimensionalen Punktwolken können verschiedene Dataset-Augmentation-Methoden eingesetzt werden, wie z. B. in HAHNER et al. (2020) beschrieben. Je nach Anwendungsfall können diese Techniken die Performance des Modells deutlich steigern – gleichzeitig kann die Wahl ungeeigneter Methoden die Performance auch negativ beeinflussen. Eine häufig verwendete Methode ist die zufällige Drehung einer Punktwolke um ihre Z-Achse, wodurch die Rotationsinvarianz des Modells erhöht wird. Drehungen um die X- und/oder die Y-Achse sind insbesondere bei topographischen Punktwolken häufig nicht plausibel. Ein weiterer Ansatz ist das Hinzufügen von zufälligem Rauschen zu den Koordinaten und den Farbinformationen, um verschiedene Aufnahmesensoren und Aufnahmekonfigurationen zu simulieren. Zusätzlich kann das zufällige Entfernen von Farbinformationen für einzelne Punkte dazu beitragen, dass das Modell stärker auf die geometrischen Eigenschaften der Punktwolke fokussiert. Die Auswahl der geeigneten Methoden hängt immer vom jeweiligen Anwendungsfall ab.

2.2.6. Metriken zur Evaluation des Modells zur semantischen Segmentierung

Zur Bewertung eines Modells stehen verschiedene Metriken zur Verfügung, die unterschiedliche Aspekte des Modells beleuchten. Die Genauigkeit (engl. *Accuracy*) ist das gebräuchlichste Maß und gibt den Anteil der richtig klassifizierten Datensätze an. Dieses Maß ist einfach zu berechnen, seine Aussagekraft ist jedoch bei unbalancierten Klassen begrenzt. Einen tieferen Einblick bietet die Konfusionsmatrix, die Kombinationen von Prädiktion und Ground Truth erfasst. *True Positives* (TP) sind übereinstimmende Prädiktionen und Ground Truth. Diese Werte werden auf der Diagonalen der Matrix angezeigt. *False Negatives* (FN) geben an, wie häufig eine tatsächliche Klasse als eine andere Klasse klassifiziert wird, während *False Positives* (FP) beschreiben, wie häufig eine Prädiktion einer anderen tatsächlichen Klasse zugeordnet wird. Die Präzision (engl. *Precision*) misst den Anteil der tatsächlich positiven Vorhersagen an allen positiven Vorhersagen, während der Recall den Anteil der tatsächlich positiven Vorhersagen an allen tatsächlich positiven Datensätzen beschreibt. Um diese Metriken zu kombinieren, wird der F1-Score berechnet (Formel 4), der das harmonische Mittel aus Recall und Precision ist (GOODFELLOW et al. 2016:411f.).

Formel 4: F1-Score (GOODFELLOW et al. 2016:412)

$$F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

Ein weiteres Maß zur Bestimmung der Ähnlichkeit zwischen zwei Datensätzen ist der Jaccard-Index (JACCARD 1901), der häufig auch als *Intersection over Union* (IoU) bezeichnet wird. Der Jaccard-Index für eine Ground-Truth y und eine Prädiktion $\hat{y}$ wird nach Formel 5 berechnet und ist das Verhältnis zwischen der Größe der Schnittmenge von y und $\hat{y}$ zur Vereinigung von y und $\hat{y}$. Abbildung 4 visualisiert einen Index von $IoU = 3/8$.

Formel 5: Jaccard Index (Intersection over Union) (LESKOVEC et al. 2014:74)

$$IoU = \frac{|y \cap \hat{y}|}{|y \cup \hat{y}|} = \frac{|y \cap \hat{y}|}{|y| + |\hat{y}| - |y \cap \hat{y}|}$$

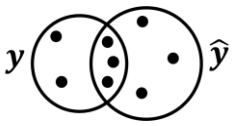


Abbildung 4: Visualisierung des Jaccard Index. Hier $IoU = 3/8$. (LESKOVEC et al. 2014:74)

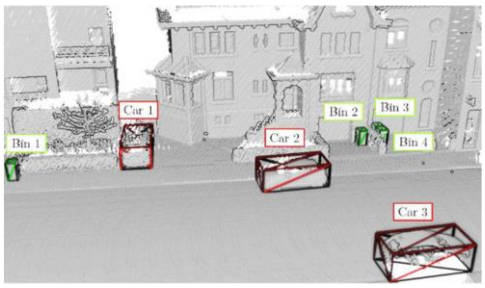
Darüber hinaus kann eine *ROC-Kurve* (*Receiver Operating Characteristic*) als Qualitätsmaß herangezogen werden. Sie zeigt den Trade-off zwischen Recall (*True Positiv Rate*, TPR) und *False Positiv Rate* (FPR) für verschiedene Schwellwerte. Eine ROC-Kurve, die entlang der Diagonalen verläuft, deutet auf einen Zufallsprozess hin. Je weiter die ROC-Kurve oberhalb der Diagonalen verläuft, desto besser sind die Vorhersagen des Modells. Ein optimaler Klassifikator hätte eine TPR von eins für alle Schwellwerte, sodass die Kurve über der linken oberen Ecke verläuft ($TPR = 1$, $FPR = 0$). Als Qualitätsmaß kann die Fläche unter dieser Kurve (*Area under ROC*, *AUROC*) berechnet werden, wobei ein Wert von eins optimal ist. Eine Fläche von 0,5 entspricht einem Zufallsprozess (AGGARWAL 2015:340f.).

2.3. Punktwolkenverarbeitung

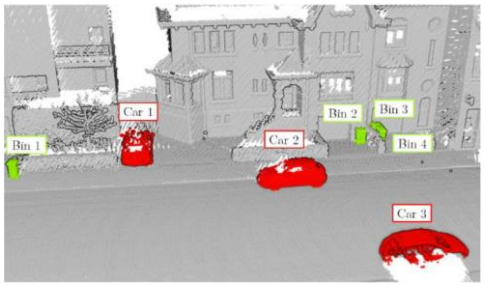
In den letzten Jahren haben sich 3D-Erhebungsmethoden erheblich weiterentwickelt, was zur Etablierung von Punktwolken als Standardprodukt in der Geodäsie geführt hat. Die Erzeugung hat inzwischen ein Stadium der Effizienz und Kosteneffektivität erreicht (GUO, Y. et al. 2021:4338). Punktwolken enthalten eine Vielzahl geometrischer Informationen, einschließlich des Maßstabs und der Form von Objekten. Die Verarbeitung von Punktwolken umfasst verschiedene Disziplinen, darunter die Klassifikation, bei der die gesamte Punktwolke einer spezifischen Objektklasse zugeordnet wird, und die (semantische) Segmentierung, bei der die Punktwolke in Regionen mit ähnlichen Eigenschaften aufgeteilt wird, wobei jedem Punkt eine semantische Bedeutung zugeordnet wird. Bei der Segmentierung von Punktwolken kommen verschiedene Methoden des maschinellen Lernens oder andere Algorithmen zum Einsatz, darunter sowohl lernbasierte als auch nicht-lernbasierte Ansätze. Lernbasierte Verfahren umfassen sowohl Methoden des Machine Learnings als auch des Deep Learning. Nicht-Lernbasierte Verfahren werden häufig zur Generierung eines geeigneten Trainingsdatensatzes eingesetzt. Im Weiteren werden die Grundlagen der Punktwolkenverarbeitung, sowie die angewandten Verfahren näher erläutert. Die Arbeitsabläufe der Punktwolkenverarbeitung bestehen sowohl bei lernbasierten als auch bei nicht-lernbasierten Verfahren aus verschiedenen Komponenten: der Definition einer Nachbarschaft, der Extraktion von Merkmalen, der Selektion von Merkmalen und der darauffolgenden Segmentierung bzw. dem Clustering (WEINMANN, M. et al. 2015b).

2.3.1. Semantische Segmentierung

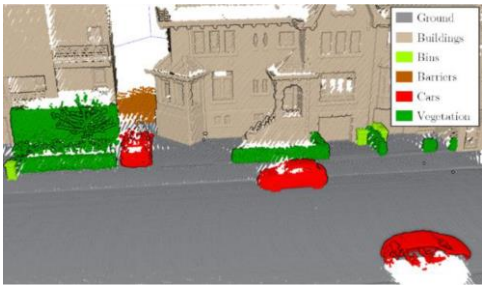
Die Segmentierung ist eines der Schlüsselthemen im Bereich der Computer Vision, bei dem Bilder in Bereiche (Segmente) mit ähnlichen Eigenschaften eingeteilt werden. Das simple Ziel dieser Aufgabe ist es, ein semantisches Verständnis der Rolle jedes Pixels zu erhalten. Diese Überlegungen für den zweidimensionalen Raum lassen sich auch auf den dreidimensionalen Raum übertragen, nur dass die Pixel durch 3D-Punkte ersetzt werden. Sowohl in der Bild- als auch in der Punktwolkenverarbeitung wird nach Abbildung 5 unterschieden in (THOMAS 2019:23):



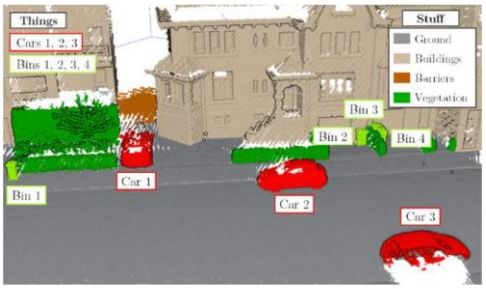
Objekterkennung: Lokalisierung und Klassifizierung von Objekten im Raum (z. B. mit einer Bounding Box). Die Objekte werden dabei nicht isoliert. Diese Aufgabe ist vor allem von Interesse, wenn es um die Anzahl und die Position von Objekten geht.



Instanzbasierte Segmentierung: Abgrenzung einzelner Objekte (Instanzen) in der Szene. Im Gegensatz zur Objekterkennung werden hier die Punkte gesucht, die zum gesuchten Objekt gehören. Dies ist z. B. interessant, wenn präzisere Messungen erforderlich sind.



Semantische Segmentierung: Jedem Pixel/Punkt wird eine semantische Klasse zugeordnet, um die Szene in semantisch sinnvolle und zusammengehörige Bereiche zu unterteilen.



Panoptische Segmentierung: Jedem Punkt wird eine Klasse und eine Identifikationsnummer (ID) zugeordnet. Es wird zwischen *Sachen* und *Dingen* unterschieden, wobei Sachen nicht zählbar sind und nur Dinge eine ID erhalten.

Abbildung 5: Teilbereiche der Segmentierung von Punktwolken (THOMAS 2019:23)

2.3.2. Punktwolken in der Theorie

Es existieren diverse Möglichkeiten zur Repräsentation von 3D-Daten – beispielweise Tiefenkarten, Punktwolken, Meshes oder volumetrische Gitter (Voxel). Die Punktwolke ist jedoch die am weitesten verbreitete Repräsentation. Punktwolken stellen einfache Datentypen dar, die aus einer ungeordneten Menge von Punkten im Raum bestehen und üblicherweise im dreidimensionalen Raum verwendet werden. Die Besonderheiten von Punktwolken zeigen sich in einer Reihe von Eigenschaften, die bei der Verarbeitung berücksichtigt werden müssen. Im Vergleich zu anderen 3D-Repräsentationen bleiben die ursprünglichen geometrischen Informationen im 3D-Raum erhalten, ohne dass eine Diskretisierung durchgeführt werden muss (GUO, Y. et al. 2021:4338). Parallel dazu bleibt eben auch das Messrauschen des Aufnahmeverfahrens erhalten. Abhängig von der Erhebungsmethode stehen neben den räumlichen Koordinaten oftmals zusätzliche Informationen wie Farbinformationen, Intensität (bei Laserscanning-Erhebungen) oder Konfidenz (bei Punktwolken aus Dense Image Matching) zur Verfügung (AKAGIC et al. 2022:1). Diese zusätzlichen Merkmale verleihen der Punktwolke eine erhöhte Informationsdichte und ermöglichen eine verbesserte Beschreibbarkeit.

Im Bereich der zweidimensionalen Bildverarbeitung sind die Segmentierungsprobleme weitgehend gelöst. Verfahren zur Segmentierung dreidimensionaler Punktwolken stehen dagegen vor neuen Herausforderungen. Dies resultiert vor allem aus der komplexen Darstellung und Struktur von Punktwolken im dreidimensionalen Raum im Vergleich zur zweidimensionalen Anordnung von Bildern (FEI et al. 2022:22863). Daraus ergibt sich die Notwendigkeit spezifischer Algorithmen und Techniken zur Erfassung und Verarbeitung der in den Punktwolken enthaltenen Informationen. Ein zweiter zentraler Aspekt ist die typischerweise hohe Punktanzahl und Punktdichte in Punktwolken, die effiziente Algorithmen und skalierbare Lösungen zur Bewältigung dieser Datensätze erfordern (WANG, W. et al. 2018:2569). Hinzu kommt die Tatsache, dass Punktwolken anfällig für Rauschen, Ungenauigkeiten und Lücken sind, die aus der Erfassungsmethode oder den Umweltbedingungen resultieren. Nach BELLO et al. (2020) weisen Punktwolken grundsätzlich folgende Charakteristiken auf:

- **Unregelmäßig**: Die Punkte sind unregelmäßig über das Objekt hinweg verteilt, sodass verschiedene Punktdichten auftreten.
- **Unstrukturiert**: Die Anordnung der Punkte entspricht keinem regelmäßigen Raster. Jeder Punkt wird individuell erhoben und die Abstände zu den Nachbarpunkten sind variabel.
- **Ungeordnet**: Die Punkte innerhalb der Punktwolke haben keine feste Reihenfolge. Die geometrische Beschaffenheit der erfassten Szene ändert sich nicht, wenn die Reihenfolge der Punkte verändert wird.

Die Segmentierung von dreidimensionalen Punktwolken erfordert einerseits ein Verständnis der globalen geometrischen Struktur und andererseits auch der feinen, detaillierten Struktur jedes einzelnen Punktes. Klassische Faltungsoperatoren, die in der Computer Vision häufig verwendet werden, können daher nicht ohne weiteres auf Punktwolken angewendet werden.

2.3.3. Definition lokaler Nachbarschaften

Da Punktwolken ungeordnet und nicht in einem regelmäßigen Raster angeordnet sind, bedarf es der Definition der Nachbarschaft einzelner Punkte im metrischen Raum. Klassischerweise werden Nachbarschaften zylindrisch (FILIN & PFEIFER, N. 2005) oder sphärisch (LINSEN & PRAUTZSCH 2001, LEE & SCHENK 2002) definiert und mit Hilfe eines Maßstabsfaktors parametrisiert – entweder durch einen Radius oder durch die Anzahl der k nächsten Nachbarn (kNN). Die Wahl der Anzahl der k nächsten Nachbarn erweist sich als flexibler und erlaubt eine bessere Anpassungsfähigkeit an variierende Punktdichten (WEINMANN, M. et al. 2015b). Im Gegensatz dazu gewährleistet ein fester metrischer Radius einen konstanten Regionsmaßstab, wodurch lokale Regionsmerkmale räumlich besser generalisierbar werden (QI, C. R. et al. 2017:4).

Die Wahl des Maßstabs erfordert Wissen über die jeweilige Szene und ist abhängig von der lokalen 3D-Struktur (WEINMANN, M. et al. 2018). Aus diesem Grund wurden adaptive Nachbarschaften entwickelt,

die Maßstabsparameter für jeden Punkt individuell anpassen. Viele Ansätze optimieren das k für jeden 3D-Punkt. Zur Optimierung werden Energiefunktionen definiert, die auf den Dimensionen der Merkmale (DEMANTKÉ et al. 2011) oder auf der Eigenentropie (WEINMANN, M. et al. 2015a) basieren. Die Energiefunktionen zur Bestimmung der optimalen Nachbarschaft basieren auf dem Strukturtensor (siehe Formel 7) und werden für jeden 3D-Punkt mit verschiedenen k-Werten, z. B. $k \in [10, 100]$, berechnet. Der k-Wert mit der geringsten Entropie bestimmt das optimale k.

Um die Berechnungszeiten dieser Nachbarschaftsberechnungen zu minimieren, werden häufig sogenannte *Kd-Trees* (BENTLEY 1975) verwendet. Ein Kd-Tree ist eine binäre Baumstruktur zur räumlichen Partitionierung und Zuordnung von Datenpunkten zu bestimmten Regionen. Er dient der Strukturierung von Daten im mehrdimensionalen Raum. Der Kd-Tree verwendet den Median einer beliebigen Achse im Merkmalsraum, um die Daten in zwei gleich große Bereiche zu unterteilen. Der resultierende Trennwert dient als Wurzelknoten der Baumstruktur. Werte, die kleiner oder gleich dem Median sind, werden im linken Kind des Wurzelknotens gespeichert, während größere Werte im rechten Kind gespeichert werden. Diese Aufteilung wird mit dem Median der nächsten Achse fortgesetzt. Dieser Prozess wiederholt sich, bis nur noch ein Punkt in einem Bereich verbleibt oder die maximale Anzahl an Verzweigungen erreicht ist (BERG, M. de et al. 2008:99ff.). Diese Baumstruktur ist nützlich für multidimensionale Abfragen wie Intervallabfragen oder die Suche nach den nächsten Nachbarn.

2.3.4. Ableitung geometrischer Merkmale

Dreidimensionale geometrische Merkmale basieren auf der lokalen 3D-Nachbarschaft eines Punktes X_0. Wichtige geometrische Eigenschaften eines 3D-Punktes $(X, Y, Z)^T$ sind die absolute Höhe Z, die maximale Differenz ΔZ und die Standardabweichung σ_Z der Z-Koordinaten innerhalb der Nachbarschaft. Abhängig von der verwendeten Nachbarschaftsdefinition umfassen die geometrischen 3D-Informationen die Anzahl der Punkte innerhalb des jeweiligen Radius (bei sphärischer oder zylindrischer Definition) oder den Radius r_{k-NN} der sphärischen Nachbarschaft, der die k nächsten Nachbarn einschließt. Bei der Rauheit wird der Abstand des jeweiligen Punktes X_0 zu einer Ebene berechnet, die in die lokale Nachbarschaft angepasst wird. Weiterhin kann die lokale Punktdichte berechnet werden, z. B. für kNN nach WEINMANN, M. et al. (2013) wie in Formel 6 beschrieben.

Formel 6: lokale Punktdichte einer kNN Nachbarschaft (WEINMANN, M. et al. 2013)

$$D = \frac{k+1}{{}^{4}/_{3} \cdot \pi \cdot {r_{k-NN}}^{3}}$$

Weiterhin basieren viele Ansätze zur Beschreibung der lokalen Struktur auf der Verwendung des 3D-Strukturtensors S, der durch die 3D-Kovarianzmatrix dargestellt wird, die aus den 3D-Koordinaten aller Punkte innerhalb der lokalen Nachbarschaft eines 3D-Punktes abgeleitet wird. Nach WEINMANN, M. et al. (2015a) wird der Strukturtensor nach Formel 7 abgeleitet mit:

Formel 7: Berechnung des Strukturtensors (WEINMANN, M. et al. 2015a:290)

$$S = \frac{1}{k+1}\sum_{i=0}^{k}(X_i - \bar{X})(X_i - \bar{X})^T \; mit \; \bar{X} = \frac{1}{k+1}\sum_{i=0}^{k} X_i$$

Basierend auf dem Strukturtensor wurde von JUTZI, B. & GROSS, H. (2009) die analytische Betrachtung seiner Eigenwerte zur Charakterisierung spezifischer Formprimitive vorgeschlagen. Weiterhin werden die normierten Eigenwerte e_i mit $i \in \{1,2,3\}$ des Strukturtensors S zur Ableitung lokaler 3D-Formmerkmale (PAULY et al. 2003, WEST et al. 2004) verwendet, die eine intuitivere Beschreibung enthalten. Diese beinhalten die Linearität L_λ, die Ebenheit P_λ, die Streuung S_λ, die Omnivarianz O_λ, die Anisotropie A_λ, die Eigenentropie E_λ, die Summe der Eigenwerte Σ_λ und die Änderung der Krümmung C_λ (Formel 8 bis Formel 15).

Formel 8: Linearität

$$L_\lambda = \frac{e_1 - e_2}{e_1}$$

Formel 9: Planarität

$$P_\lambda = \frac{e_2 - e_3}{e_1}$$

Formel 10: Streuung

$$S_\lambda = \frac{e_3}{e_1}$$

Formel 11: Omnivarianz

$$O_\lambda = \sqrt[3]{e_1 e_2 e_3}$$

Formel 12: Anisotropie

$$A_\lambda = \frac{e_1 - e_3}{e_1}$$

Formel 13: Eigenentropie

$$E_\lambda = -\sum_{i=1}^{3} e_i \cdot \ln(e_i)$$

Formel 14: Summe der Eigenwerte

$$\Sigma_\lambda = \lambda_1 + \lambda_2 + \lambda_3$$

Formel 15: Änderung der Krümmung

$$C_\lambda = \frac{e_3}{e_1 + e_2 + e_3}$$

Diese Merkmale werden zudem häufig ergänzt durch Höhen- und lokale Ebeneneigenschaften (MALLET et al. 2011), grundlegende Eigenschaften der Nachbarschaft und Eigenschaften einer 2D-Projektion (WEINMANN, M. et al. 2013). Neben den genannten Merkmalen, gibt es auch komplexere Merkmale wie *Spin Images* (JOHNSON & HEBERT 1999), Formverteilungen (OSADA et al. 2002, BLOMLEY et al. 2014) oder Punkt-Merkmals-Histogramme (RUSU et al. 2009). Der Rechenaufwand für diese Merkmale ist jedoch hoch und der Mehrwert relativ gering. WEINMANN, M. et al. (2015a) und HACKEL et al. (2016) bewiesen, dass einfache Kovarianzmerkmale einen deutlich geringeren Rechenaufwand bieten und gleichzeitig eine sehr aussagekräftige Beschreibung liefern (THOMAS 2019:61).

2.3.5. Ableitung radiometrischer Eigenschaften

Punktwolken enthalten häufig Farbinformationen im RGB-Farbraum, welche auf der additiven Farbmischung der Primärfarben Rot, Grün und Blau basiert. Der Farbraum kann in einem kartesischen Koordinatensystem dargestellt werden, wobei alle möglichen Farben innerhalb eines Würfels abgebildet werden (siehe Abbildung 6). Die Primärfarben liegen dabei auf den Achsen des Koordinatensystems in den drei Ecken des Würfels. Schwarz liegt im Ursprung des Koordinatensystems und Weiß in der am weitesten vom Ursprung entfernten Ecke.

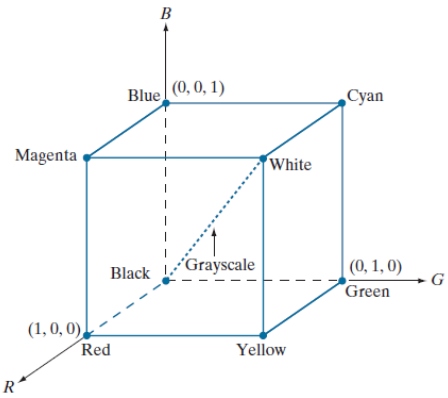


Abbildung 6: RGB-Farbraum im kartesischen Koordinatensystem (GONZALEZ & WOODS 2018:407)

Die Grauwerte liegen in diesem Modell entlang der Linie von Schwarz nach Weiß. Erst die Kombination der drei Farbwerte beschreibt die konkreten Farben (GONZALEZ & WOODS 2018:407).

Der HSV-Farbraum (Hue, Saturation, Value) korrespondiert stark mit der menschlichen Farbempfinden und ist daher oft leichter zu interpretieren als RGB-Farben. In diesem Raum werden die Grauwerte (*Value*, bzw. Intensität) von den farbtragenden Merkmalen (*Saturation*, bzw. Sättigung und *Hue*, bzw. Farbwert) getrennt. Zur Bestimmung der Farbwerte im HSV-Farbraum wird ein zylindrisches Koordinatensystem definiert, dessen Ursprung im Punkt $(0, 0, 0)$ liegt und dessen Z-Achse auf die Grauwertachse des RGB-Farbraums ausgerichtet ist. Der Abszissenwert eines Farbpunkts gibt die Intensität (Value) und die Ordinate die Farbsättigung (Saturation) an. Der Farbwert (Hue) entspricht dem Polarwinkel eines Farbpunkts, wobei Rot, $(1, 0, 0)$ im RGB-Farbraum, üblicherweise als Bezugsrichtung festgelegt wird (GONZALEZ & WOODS 2018:411f.).

Zur Konvertierung von RGB zu HSV können folgende Formeln angewendet werden. Der Farbwert H ergibt sich nach Formel 16, wobei angenommen wird, dass die RGB-Werte normiert sind und der Winkel θ auf die Rot-Achse bezogen ist.

Formel 16: Berechnung des Farbwertes (Hue) aus gegebenen RGB-Farbwerten (GONZALEZ & WOODS 2018:413)

$$H = \begin{cases} \theta \; wenn \; B \leq G \\ 360 - \theta \; wenn \; B > G \end{cases} \quad mit \;\; \theta = \arccos\left(\frac{\frac{1}{2}\left((R-G)+(R-B)\right)}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right)$$

Die Saturation S wird mit Formel 17 und der Grauwert (Value) I mit Formel 18 bestimmt.

Formel 17: Berechnung der Farbsättigung (Saturation) (GONZALEZ & WOODS 2018:413)

$$S = 1 - \frac{3}{R+G+B} \cdot \min(R, G, B)$$

Formel 18: Berechnung der Grauwerte (Value) (GONZALEZ & WOODS 2018:415)

$$I = \frac{1}{3}(R+G+B)$$

2.3.6. Unüberwachte Segmentierung und Clustering

Algorithmen der unüberwachten Segmentierung erfordern im Gegensatz zu überwachten Verfahren keine annotierten Trainingsdaten. Sie nutzen die Geometrie und verschiedene Merkmale, um Bereiche mit ähnlichen Eigenschaften zu identifizieren. Dabei kommen sowohl verschiedene Verfahren des maschinellen Lernens als auch speziell für Punktwolken entwickelte Algorithmen zum Einsatz. Letztere verwenden üblicherweise nur die Geometrie der Punkte und prüfen anhand verschiedener Analysen, ob bestimmte Eigenschaften erfüllt sind. Beim maschinellen Lernen soll der Abstand innerhalb der Cluster minimal und zwischen den Clustern maximal sein (DÖRRE et al. 2001:439). Diese Verfahren können zur effizienten Erstellung von Trainingsdaten genutzt werden, indem nach der Gruppierung semantische Klassen zugeordnet werden. Klassische Clustering-Verfahren lassen sich gemäß DOBILAS (2021) in vier Hauptkategorien einteilen:

1. Centroid-basierte Verfahren – verwenden Distanzmetriken, um jeden Punkt zum nächsten Cluster hinzuzufügen. Beispiel: *K-Means* (MACQUEEN 1967).
2. Konnektivitätsbasierte Verfahren – gehen davon aus, dass nahe Objekte stärker miteinander verbunden sind als weiter entfernte. Beispiel: *Hierarchisches Clustering*.
3. Dichtebasierte Verfahren – definieren Cluster als dichte Regionen, die von Bereichen mit geringer Dichte separiert werden. Beispiel: *DBSCAN* (ESTER et al. 1996).
4. Verteilungsbasierte Verfahren – nehmen an, dass die Daten aus verschiedenen Verteilungen mit eigenem Mittelwert und eigener Varianz stammen. Beispiel: *Gaussian Mixture Models* (GMM).

Speziell für Punktwolken entwickelte Techniken zur Segmentierung sind insbesondere kantenbasierte Algorithmen, *Region-Growing*-Methoden oder *Model-Fitting*-Verfahren (GRILLI et al. 2017). Die kantenbasierte Segmentierung besteht dabei aus zwei Hauptschritten: der Kantendetektion zur Identifikation von Grenzen zwischen Regionen und der anschließenden Gruppierung von Punkten innerhalb der Grenzen. Kanten werden durch Punkte definiert, an denen lokale Oberflächeneigenschaften einen festgelegten Grenzwert überschreiten. Häufig verwendete Eigenschaften sind Normalenvektoren, Gradienten oder Krümmungen. Region-Growing-Methoden starten von bestimmten Punkten (sog. *Seed Points*) aus und fügen dem Segment iterativ benachbarte Punkte mit ähnlichen Eigenschaften hinzu. Dieser Prozess umfasst zwei Schritte: die Identifizierung von *Seed Points* basierend auf der Krümmung jedes Punktes und das anschließende „Wachsen“ der Region gemäß vordefinierter Kriterien wie Punktnähe oder Oberflächenplanarität. Die Segmentierung mittels Model Fitting basiert auf der Idee, dass künstliche Objekte durch Primitive approximiert werden können. Bei diesen Methoden werden verschiedene Primitive (z. B. Kugeln, Zylinder, Ebenen) bestmöglich eingepasst und Punkte, die zu den entsprechenden Formen passen, werden demselben Segment zugeordnet. Häufig eingesetzte Algorithmen sind die *Hough-Transformation* (BALLARD 1981) oder *Random Sample Consensus* (*RANSAC*) (FISCHLER & BOLLES 1981).

Im Folgenden werden das zentrumsbasierte Verfahren *K-Means* und das dichtebasierte Verfahren *DBSCAN*, die zu den Machine Learning Verfahren gehören, näher erläutert. Zusätzlich wird der *Cloth-Simulation-Filter* (CSF) vorgestellt, der speziell für die Klassifikation von LiDAR-Punktwolken in Bodenpunkte und Nicht-Bodenpunkte entwickelt wurde.

2.3.6.1. Clustering basierend auf K-Means

Der K-Means-Algorithmus (MACQUEEN 1967) stellt eines der ersten unüberwachten maschinellen Lernverfahren dar, um 3D-Punkte anhand ihrer Eigenschaften zu gruppieren. Die Clusterbildung erfolgt durch Minimierung der Summe der quadrierten Distanzen zwischen den Punkten und den entsprechenden Cluster-Schwerpunkten (sog. Centroide). Die Initialisierung der Centroide $\{\mu^{(1)}, \dots, \mu^{(K)}\}$ kann entweder zufallsbasiert, oder durch *K-Means++* (ARTHUR & VASSILVITSKII 2007) erfolgen. Bei K-Means++ wird nur der erste Centroid zufällig ausgewählt und die euklidischen Distanzen $d_{i,j}$ des Centroiden i zu allen Punkten j berechnet. Die berechneten Distanzen werden für eine Wahrscheinlichkeitsfunktion $w_i = d_{i,j} / \sum_{j=0}^{n} d_{i,j}$ für die Auswahl des nächsten Centroiden verwendet. Punkte mit größeren Abständen erhalten eine höhere Wahrscheinlichkeit, während Punkte in der Nähe eines Centroids eine geringere Wahrscheinlichkeit erhalten. Für weitere Iterationen wird die minimale Distanz für die Wahrscheinlichkeitsfunktion verwendet. Dieser Prozess wird so lange wiederholt, bis die gewählte Anzahl an Centroiden erreicht ist. Diese verbesserte Initialisierung wird die Wahrscheinlichkeit der Konvergenz zu suboptimalen Lösungen verringert (ATTA 2023).

Nach der Initialisierung (1. in Abbildung 7) wird eine Distanzmetrik (in der Regel die euklidische Distanz) zwischen jedem Datenpunkt und jedem Cluster-Centroid berechnet. In einem ersten Schritt wird jeder Datensatz dem Cluster i hinzugefügt, wobei i den Index des dichtesten Centroids $\mu^{(i)}$ darstellt (2. in Abbildung 7). Dann werden die Koordinaten jedes Centroids $\mu^{(i)}$ durch den Mittelwert aller Datenpunkte $x^{(j)}$ erneuert, die zum jeweiligen Cluster i gehören. (3. in Abbildung 7). Dieser Prozess wird solange wiederholt, bis die Summe der Quadrate innerhalb jedes Clusters ein Minimum erreicht oder bis sich das Ergebnis nicht mehr ändert (4. in Abbildung 7) (GOODFELLOW et al. 2016:144).

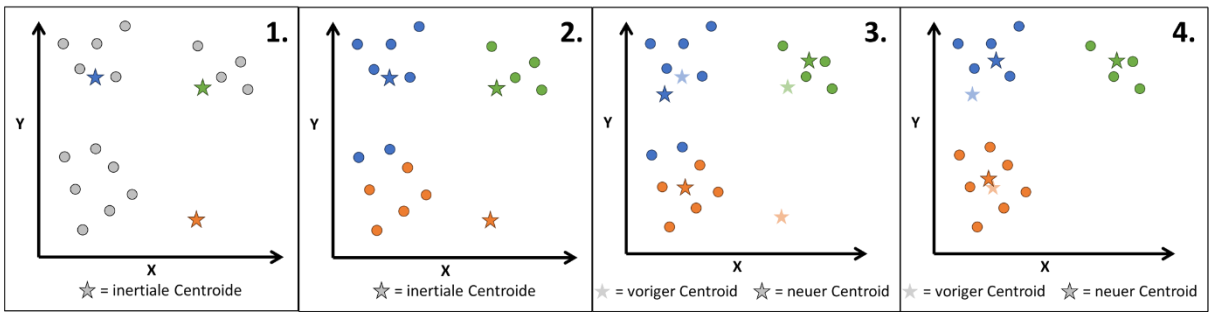


Abbildung 7: Visualisierung des K-Means Algorithmus

Ein weiterer entscheidender Schritt für das Resultat von K-Means, insbesondere bei mehrdimensionalen Daten, ist die Wahl der Anzahl der Cluster K. Eine Möglichkeit, diesen Parameter optimal zu wählen, ist die *Ellbow-Curve-Method*. Die Grundidee dieses heuristischen Verfahrens besteht darin, die Quadratsumme der Abstände für verschiedene K zu visualisieren, um den „Ellenbogen-Punkt“ zu identifizieren. Typischerweise nimmt die Quadratsumme, wie in Abbildung 8 dargestellt, bis zu diesem Punkt stark ab. In diesem Beispiel könnte $K = 6$ eine optimale Anzahl von Clustern sein.

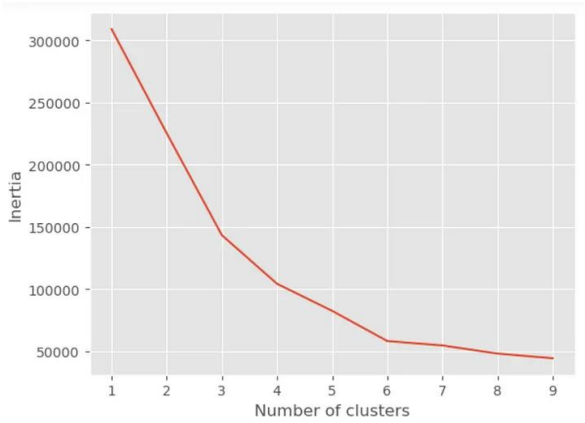


Abbildung 8: Visualisierung der Ellbow-Curve-Method. Anzahl der Cluster sind auf der X- und die zugehörige Quadratesumme auf der Y-Achse dargestellt. (WEI 2023)

2.3.6.2. Segmentierung basierend auf DBSCAN

Ein weiterer populärer Clustering-Algorithmus ist *DBSCAN* (*Density-Based Spatial Clustering of Applications with Noise*), der von ESTER et al. (1996) vorgestellt wurde und zu den dichtebasierten Verfahren gehört. Im Gegensatz zu K-Means kann DBSCAN Rauschen erkennen und herausfiltern. Der generelle Ablauf von DBSCAN ist in Abbildung 9 dargestellt.

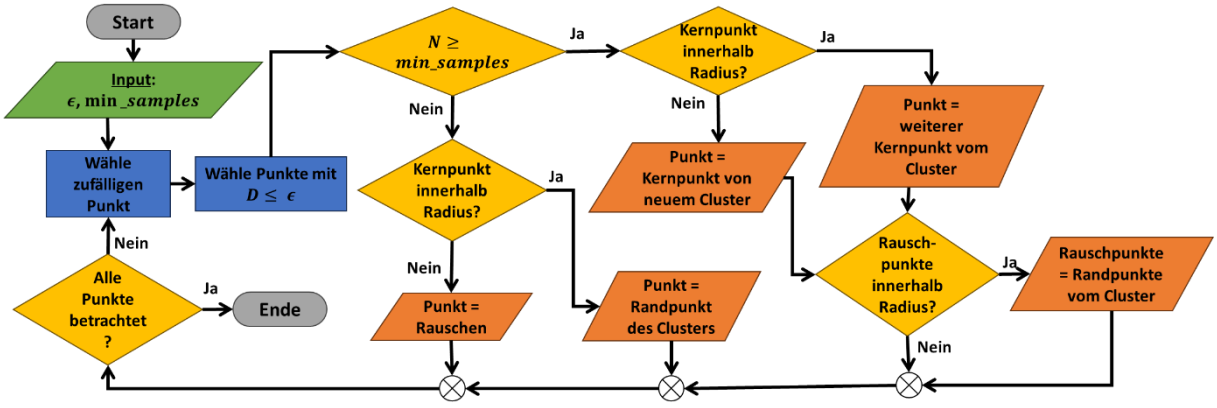


Abbildung 9: Ablaufdiagramm des DBSCAN Algorithmus

Zur Initialisierung werden zwei Parameter benötigt: der Radius (ϵ), der die maximale Distanz zwischen zwei Punkten innerhalb eines Clusters angibt, und die minimale Anzahl von Punkten pro Cluster ($min\_samples$). Der Algorithmus wählt iterativ zufällig einen Punkt aus und selektiert alle

Datenpunkte, deren Distanz D kleiner als der Radius ϵ ist. Wenn die Anzahl der selektierten Punkte N die Mindestanzahl der Punkte pro Cluster überschreitet, handelt es sich bei dem Punkt um einen Kernpunkt. Liegt ein anderer Kernpunkt einer vorhergehenden Iteration innerhalb des Radius, so wird der Punkt dem Cluster des anderen Kernpunktes hinzugefügt. Befindet sich kein Kernpunkt in der Nachbarschaft, wird ein neues Cluster gebildet. Befinden sich in der Nachbarschaft Punkte, die in früheren Iterationen als Rauschpunkte identifiziert wurden, so werden diese als Randpunkte dem jeweiligen Cluster hinzugefügt. Wird die Mindestanzahl von Clusterpunkten nicht erreicht, so ist dieser Punkt ein Rauschpunkt, wenn sich in der Nachbarschaft kein Kernpunkt eines Clusters befindet. Wenn ein Kernpunkt in der Nachbarschaft vorhanden ist, wird der Punkt als Randpunkt zu dem betreffenden Cluster hinzugefügt. Dieser Vorgang wird solange wiederholt, bis alle Punkte berücksichtigt wurden (DOBILAS 2021). Dieser Algorithmus hat den Vorteil, dass er mit beliebig geformten Clustern arbeiten kann und in der Lage ist, Ausreißer zu erkennen. Gleichzeitig ist die Wahl der Parameter nicht trivial und DBSCAN stößt an seine Grenzen, wenn die Dichte innerhalb eines Datensatzes variiert.

2.3.6.3. Cloth-Simulation-Filter

ZHANG, W. et al. (2016) stellten für die unüberwachte Klassifikation in Bodenpunkte und Nicht-Bodenpunkte den *Cloth Simulation Filter* (CSF) vor, der speziell für LiDAR-Punktwolken entwickelt wurde. Dieses Verfahren benötigt nur wenige intuitive Parameter zur Initialisierung, was seine Anwendung zugänglich macht. Bildlich gesprochen kann man sich ein Tuch vorstellen, das aufgrund der Gravitation auf ein Gelände fällt. Unter der Voraussetzung, dass das Tuch weich genug ist, um auf der Oberfläche zu haften, ist die finale Form des Tuches das digitale Oberflächenmodell. Wird die Punktwolke jedoch zunächst durch eine Spiegelung in der XY-Ebene invertiert und dem Tuch eine gewisse Starrheit verliehen, stellt die endgültige Form das digitale Geländemodell dar. Das Tuch wird durch ein Masse-Feder-Modell dargestellt, das aus einem Gitter von Partikeln und Verbindungen besteht. Jedes Partikel im Gitter hat keine physikalische Ausdehnung, aber eine konstante Masse. Vor der Anwendung von CSF ist in der Regel eine automatische oder manuelle Ausreißerbehandlung erforderlich. Nach der Invertierung der Punktwolke wird ein Tuchgitter mit einer benutzerdefinierten Auflösung initialisiert. Die inertiale Position des Gitters ist normalerweise der höchste Punkt. Anschließend werden die Tuchpartikel und LiDAR-Punkte auf eine horizontale Fläche projiziert, wobei für jedes Partikel der nächstgelegene LiDAR-Punkt (Korrespondenzpunkt, KP) identifiziert wird. Zusätzlich wird der Höhenwert vor der Projektion (HVP) bestimmt, der den tiefsten Punkt darstellt, den das Partikel erreichen kann. Anschließend wird die Position jedes Gitterpartikels berechnet, die von der Schwerkraft beeinflusst wird, wenn das Partikel beweglich ist (Formel 19).

Formel 19: Position der Gitterpunkte nach Einfluss der Gravitation pro Zeiteinheit (ZHANG, W. et al. 2016:5)

$$X(t+\Delta t) = 2X(t) - X(t-\Delta t) + \frac{G}{m}\Delta t^2$$

wobei X die Position $(X, Y, Z)^T$, m die Masse der Partikel (wird für gewöhnlich gleich 1 gesetzt) und Δt die Zeitdifferenz ist. Mit einer gegeben Zeitdifferenz ist diese Gleichung direkt lösbar, da G eine Konstante ist. Ist der berechnete Höhenwert $Z_i \leq HVP_i$, dann wird $Z_i = HVP_i$ gesetzt und der Punkt als unbeweglich festgelegt. Aufgrund der Starrheit des Tuches entsteht eine interne Kraft und die Partikel im Gitter versuchen in ihre horizontale Position zurückzukehren. Diese Disposition wird anschließend durch Formel 20 modelliert.

Formel 20: Berechnung der Disposition aufgrund der inneren Kräfte (ZHANG, W. et al. 2016:6)

$$\vec{d} = \frac{1}{2} b(\overrightarrow{p_i} - \overrightarrow{p_0}) \cdot \vec{n}$$

wobei $b = 0$, wenn der Punkt unbeweglich ist oder $b = 1$, wenn er beweglich ist. $\overrightarrow{p_0}$ beschreibt die aktuelle Position des Punktes und $\overrightarrow{p_i}$ die Position des benachbarten Partikels. $\vec{n}$ ist der Normalenvektor, welcher in vertikaler Richtung zeigt, $\vec{n} = (0, 0, 1)^T$. Dieser Bewegungsprozess kann wiederholt werden, wobei die Anzahl der Wiederholungen die Starrheit (engl. *Rigidness*, RI) des Tuches definiert. Bei $RI = 1$ wird der Prozess einmal wiederholt und die Disposition beträgt die Hälfte der vertikalen Distanz zwischen den Partikeln. Bei $RI = 2$ beträgt die Disposition $3/4$ der vertikalen Distanz und bei $RI = 3$ beträgt sie $7/8$ der vertikalen Distanz. Dieser Prozess der Modellierung der Disposition durch Gravitation und interner Kraft ist in Abbildung 10 dargestellt und wird wiederholt, bis die maximale Höhenvariation aller Partikel klein genug ist oder eine maximale Anzahl von Iterationen erreicht wurde.

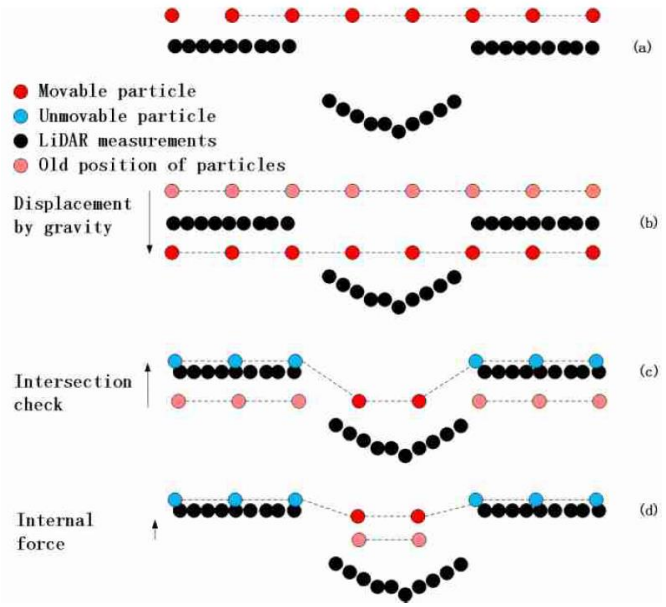


Abbildung 10: Hauptschritte von CSF. (a) inertialer Status, (b) Disposition durch den Einfluss der Gravitation, (c) Prüfung der Überschreitung des Grenzwertes HVP, (d) Berücksichtigung der internen Kräfte. (ZHANG, W. et al. 2016:5)

Abschließend wird die Cloud-to-Cloud Distanz zwischen den Gitter-Partikeln und der LiDAR-Punktwolke berechnet und zwischen Bodenpunkten und Nicht-Bodenpunkten unterschieden. Überschreitet die Distanz in Z-Richtung einen Schwellenwert, handelt es sich um einen Nicht-Bodenpunkt, alle Punkte innerhalb des Schwellenwertes sind Bodenpunkte.

Die vorgestellte Methode kann bei steilen Neigungen zu großen Fehlzuordnungen führen. Dieses Problem kann durch eine Nachbearbeitungsmethode gelöst werden, welche die Kanten steiler Hänge glättet. Bei dieser Methode wird in den vier adjazenten Nachbarschaften jedes beweglichen Partikels ein unbewegliches Partikel gesucht und mit den Höhenwerten der korrespondierenden Punkte verglichen. Wenn die Höhendifferenz innerhalb eines Schwellenwerts liegt, wird der bewegliche Punkt auf den Boden gesetzt und als unbeweglich definiert.

2.3.7. Herkömmliche überwachte Machine-Learning-Verfahren zur Segmentierung

Nach WEINMANN, M. et al. (2015b) können herkömmliche überwachte Verfahren zur punktweisen Klassifikation in zwei Gruppen eingeteilt werden: Die erste Gruppe verwendet Standard-ML-Algorithmen wie *Random Forest* (CHEHATA et al. 2009) oder *Support Vector Machines* (ZHANG, J. et al. 2013). Die andere Gruppe sind statistische kontextuelle Modelle wie *Assoziierte* oder *Nicht-assoziierte Markov Modelle* (SHAPOVALOV et al. 2010) oder *Conditional Random Fields* (NIEMEYER et al. 2012). Die statistisch kontextuellen Modelle sind probabilistische Verfahren, die die Einbeziehung kontextueller Informationen und das Erlernen spezifischer Beziehungen zwischen Objektklassen innerhalb eines Trainingsschritts ermöglicht. Da sie im Rahmen dieser Arbeit keine Anwendung finden, werden sie nicht weiter erläutert.

Support Vector Machines (SVM) (CORTES & VAPNIK 1995) zielen darauf ab, eine Hyperebene im Merkmalsraum zu finden, die die Klassen bestmöglich voneinander trennt. Die Datenpunkte, die der Trennebene am nächsten liegen, werden als „Support Vektoren“ bezeichnet. Das Hauptziel von SVM besteht darin, die optimale Hyperebene zu bestimmen, bei der die Distanz zwischen Support Vektoren und der Hyperebene maximal ist. Dieses Vorgehen wird als *Maximum-Margin-Prinzip* bezeichnet (AGGARWAL 2015:313f.). Die Hyperebenen können sowohl durch lineare Funktionen als auch durch die radiale Basisfunktion dargestellt werden, sodass eine nichtlineare Klassifikation möglich ist.

Zu den am häufigsten verwendeten Machine-Learning-Algorithmen für die semantische Segmentierung von Punktwolken ist Random Forest (RF) (BREIMAN 2001), das zu den *Ensemble*-Lernverfahren gehört. Dieses Verfahren basiert auf der Aggregation der Ergebnisse eines Ensembles einfacher Schätzer. Durch die Mehrheitsentscheidung einer Reihe von Schätzern kann eine bessere Performance erreicht werden als durch einzelne Schätzer. Random Forest verwendet dabei ein Ensemble von randomisierten Entscheidungsbäumen. Ursprünglich wurden Entscheidungsbäume von BREIMAN et al. (1984) entwickelt. Als Wurzelknoten wird das Attribut gewählt, das die Klassen aufgrund der Attributausprägungen am besten trennt. Von diesem Knoten gehen Kanten aus, die alle möglichen Ausprägungen des Attributs widerspiegeln. Der nächste Knoten wird wiederum durch die beste Trennbarkeit der Klassenzugehörigkeit der verbleibenden Trainingsdaten ausgewählt. Dieser Prozess wird wiederholt, bis der Entscheidungspfad im Baum alle Elemente einer Klasse zuordnet. Die Auswahl

des Attributs, das die Klassen am besten trennt, erfolgt nach dem ID3-Verfahren (QUINLAN 1986) über den Informationsgewinn (*Information Gain, IG*).

Jeder der Entscheidungsbäume eines Random Forest sagt eine Klassenzugehörigkeit für einen Datensatz voraus und eine Mehrheitsentscheidung bestimmt die endgültige Klassenzugehörigkeit *(YIU 2019)*. Der Grundgedanke dieses Ansatzes ist, dass sich die relativ unkorrelierten Entscheidungsbäume gegenseitig vor individuellen Fehlern schützen. Einzelne Bäume können zwar eine falsche Entscheidung treffen, aber die Mehrheit der Bäume liegt mit ihren Vorhersagen richtig. Die Unkorreliertheit eines Entscheidungsbaumes zu den anderen Modellen wird durch zwei Methoden sichergestellt: *Bagging* und *Random-Split Selection*.

Bagging, auch *Bootstrap-Aggregation* genannt, ist ein Ansatz zur Minimierung der Varianz der Vorhersage. Es basiert auf der Idee, dass, wenn die Varianz einer Vorhersage σ^2 beträgt, die Varianz auf σ^2/k reduziert wird, indem k unabhängige und identisch verteilte Vorhersagen gemittelt werden. Unabhängigkeit und Gleichverteilung werden erzielt, indem jeder Entscheidungsbaum eine zufällige Stichprobe mit Zurücklegen aus dem Trainingsdatensatz zieht (AGGARWAL 2015:379).

Durch die Verwendung von *Random-Split Selection* können die einzelnen Entscheidungsbäume nur aus einer Teilmenge der Attribute wählen, sodass nicht alle Attribute für die Wahl des Knotenkriteriums bzw. für die Berechnung des Informationsgewinns zur Verfügung stehen. Dies erzwingt eine größere Variation der Bäume und führt zu einer geringeren Korrelation zwischen den Bäumen (AGGARWAL 2015:380).

Dieser Klassifikator besitzt nach VANDERPLAS (2018:513) eine Reihe von Vorteilen:

- Training und Vorhersage sind aufgrund der Einfachheit schnell zu berechnen. Aufgaben können einfach parallelisiert werden, da die Entscheidungsbäume unabhängig voneinander sind.
- Mehrere Bäume ermöglichen eine probabilistische Klassifikation, indem die Mehrheitsentscheidung eine Abschätzung der Wahrscheinlichkeit liefert.
- Das nichtparametrisierte Modell ist flexibel und eignet sich besonders für kleine Trainingsdatenmengen.

In den erläuterten Ansätzen werden die Komponenten der Nachbarschaftsdefinition, Merkmalsableitung und überwachten Klassifikation zur semantischen Segmentierung von Punktwolken unabhängig voneinander durchgeführt. Allerdings wäre es von Vorteil diese Komponenten miteinander zu verknüpfen, indem die Ergebnisse über alle Komponenten hinweg geteilt werden (GRILLI et al. 2017).

2.3.8. Deep-Learning-Verfahren zur semantischen Segmentierung

Wie bereits beschrieben, finden Deep-Learning-Ansätze in vielen Bereichen Anwendung. Beispielsweise werden CNNs in der Computer Vision eingesetzt, um Bilder und Videos zu analysieren. Im Bereich der natürlichen Sprachverarbeitung können andere Verfahren menschliche Sprache verstehen und darstellen. Die hier verwendeten Datentypen teilen die Gemeinsamkeit, dass sie strukturiert sind, was ihre Indexierung ermöglicht, und Nachbarschaftssuchen begünstigt. Wie bereits in Abschnitt 2.3.2 beschrieben, sind Punktwolken nicht strukturiert und die Speicherung von Punktwolken ist inkonsistent. Punktwolken stellen Deep-Learning-Verfahren vor neue Herausforderungen – die größten Herausforderungen sind die Translationsinvarianz, die Rotationsinvarianz und die Permutationsinvarianz (DU 2021). Eine kategorische Einordnung bestehender Deep-Learning-Ansätze für Punktwolken wird beispielsweise in HE, Y. et al. (2021), GUO, Y. et al. (2021) und BELLO et al. (2020) vorgestellt, welche sich durch Abbildung 11 visualisieren lassen.

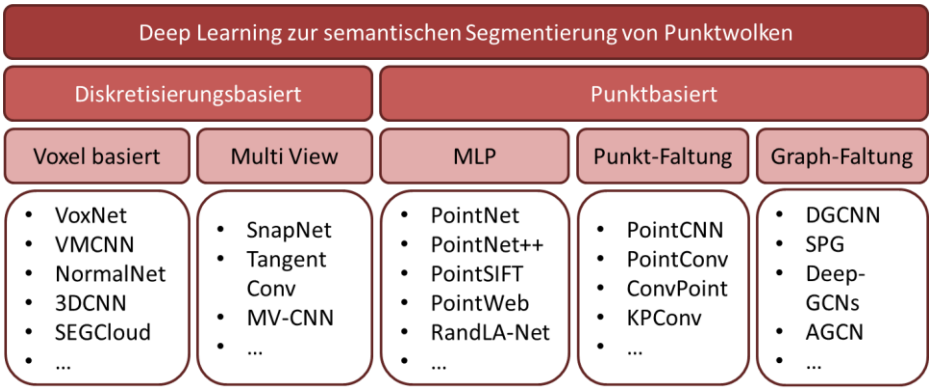


Abbildung 11: Überblick in Deep-Learning Verfahren zur Verarbeitung von Punktwolken (BELLO et al. 2020, HE, Y. et al. 2021, GUO, Y. et al. 2021)

Deep Learning für Punktwolken ist ein junges Forschungsgebiet. Die ersten Architekturen haben Punktwolken zunächst in ein strukturiertes Gitter überführt – z. B. in ein 3D-Voxelgitter, in Multi-View-Repräsentationen oder in höherdimensionale Gitter. Die Projektion in zweidimensionale Bilder ist ein nachvollziehbarer Ansatz, da Faltungsoperationen für Bilder weit entwickelt sind. Eine der ersten Architekturen dieser Art ist das von SU, H. et al. (2015) vorgestellte *Multi-View CNN* (MV-CNN). Wie in Abbildung 12 dargestellt, funktioniert dies insbesondere für isolierte Objekte gut.

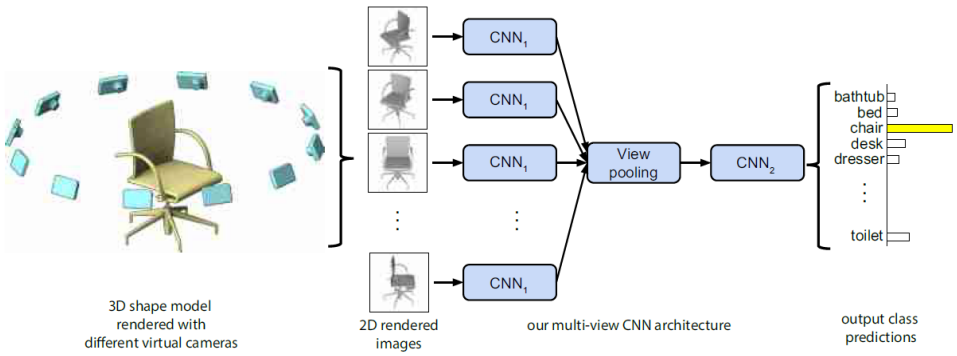


Abbildung 12: Semantische Classification von 3D Objekten mit Multi-View CNN. (SU, H. et al. 2015)

Diese Multi-View-basierten Methoden basieren stark auf den inertialen und finalen Schritten: Die Auswahl der Viewpoints und die Rückprojektion der Segmentierung. Diese Schritte sind nicht robust und die gesamte Pipeline leidet unter den auftretenden Oberflächen- und Dichtevariationen (THOMAS 2019:92f.). Um der Natur von Punktwolken besser gerecht zu werden, werden sie in 3D-Gitter (Voxelgitter) projiziert, wie in *VoxNet* (MATURANA & SCHERER 2015). Da Bildfaltungen nicht auf zwei Dimensionen beschränkt sind, können CNNs in einfacher Weise auf drei Dimensionen erweitert werden, um voxelbasierte Netzwerke zu designen. Ein generelles Problem dieser Architekturen ist jedoch, dass die Komplexität mit der Eingangsgröße kubisch statt quadratisch ansteigt. Dies wirkt sich vor allem auf den Speicherbedarf aus, der durch die Kapazität des Grafikprozessors limitiert ist. Die Diskretisierung führt zudem zu einem Informationsverlust. Aufgrund der Unstrukturiertheit der Daten ist die Anwendung von Standard-CNNs auf die direkte Punktwolke nicht möglich. Erst die Entwicklung von *PointNet* (QI, C. R. et al. 2016) ermöglichte erstmals eine direkte Verarbeitung ohne vorherige Diskretisierung. Die punktbasierten Verfahren lassen sich grob in punktweise MLP-Verfahren, Punktfaltungsverfahren und graphbasierte Verfahren unterteilen. Punktweise MLP-Verfahren verwenden in der Regel geteilte MLPs als Basiseinheit. Sie sind jedoch nicht in der Lage, lokale Geometrien oder Wechselwirkungen zwischen Punkten angemessen zu extrahieren. Um einen erweiterten Kontext zu erfassen, wurden verschiedene Netze vorgeschlagen. Beispielsweise aggregieren Methoden des *benachbarten Feature-Poolings* Informationen von lokalen Nachbarpunkten, um lokale geometrische Muster zu erfassen. Hierarchische MLP-Architekturen wie *PointNet++* (QI, C. R. et al. 2017) extrahieren und aggregieren Merkmale in mehreren Maßstäben.

Punktfaltungsmethoden basieren auf den Grundgedanken der klassischen Bildfaltung und bieten effiziente Faltungsoperationen für Punktwolken. Der Hauptunterschied zur Bildfaltung liegt in der Definition der Nachbarschaft für die Kernel-Funktion. Während bei Bildern benachbarte Pixel verwendet werden, kann die Lokalität von 3D-Punkten durch einen Radius oder kNN abgeleitet werden. Ein Beispiel hierfür liefern THOMAS et al. (2019) mit einem *Kernel Point Fully Convolutional Network* (KP-FCNN), basierend auf *Kernel Point Convolutions* (KPConv). Dabei werden die Faltungsgewichte von KPConv durch die euklidischen Abstände zu den Kernelpunkten bestimmt, wobei die Anzahl der Kernelpunkte nicht festgelegt ist. Die Position der Kernelpunkte wird durch ein Optimierungsproblem der besten Abdeckung im Kugelraum formuliert (GUO, Y. et al. 2021:18f.).

Graph-Faltungs-Methoden verwenden Graphen-Netzwerke, um die zugrunde liegenden Formen und geometrischen Strukturen von Punktwolken zu erfassen. Nachbarschaftsbeziehungen werden durch attributierte gerichtete Graphen simuliert. LANDRIEU & SIMONOVSKY (2017) stellten eine Punktwolke als eine Reihe miteinander verbundener einfacher Formen und Superpunkte dar und verwendeten einen attributierten gerichteten Graphen (bezeichnet als *Superpunktgraphen*, SPG), um die Struktur- und Kontextinformationen zu erfassen. Das Problem der großräumigen Punktwolkensegmentierung wird

anschließend in drei Teilprobleme aufgeteilt: geometrische homogene Partitionierung, Superpunkteinbettung und kontextbezogene Segmentierung (GUO, Y. et al. 2021:18f.).

2.3.8.1. PointNet

Der Einsatz von Deep-Learning-Verfahren zur direkten Verarbeitung von Punktwolken wurde durch die Pionierarbeit *PointNet* von QI, C. R. et al. (2016) ermöglicht. Der Ansatz von PointNet liegt in der Lösung der bestehenden Invarianzprobleme: Es lernt eine Rotationsmatrix, um Rotationsinvarianz herzustellen; es verwendet eine symmetrische Funktion für das globale Pooling, um Permutationsinvarianz herzustellen und es verwendet lokale Koordinaten, um Translationsinvarianz herzustellen (DU 2021). Das Netzwerk approximiert eine allgemeine Funktion f durch zwei Hilfsfunktionen g und h, wie in Formel 21 dargestellt.

Formel 21: Approximation einer Funktion durch PointNet (QI, C. R. et al. 2016)

$$f(\{p_1, p_2, \dots, p_n\}) \approx g(h(p_1), h(p_2), \dots, h(p_n))$$

Um unabhängig von der Eingabereihenfolge zu sein, muss die Funktion g symmetrisch zu ihren Argumenten sein – wie z. B. beim Max Pooling, bei dem nur die maximalen Ausprägungen ausgewählt werden. Die Funktion h wird approximiert durch MLP mit einer festen Anzahl an Layern und Neuronen. Dabei wird nicht für jeden Punkt eine Funktion einzeln gelernt, sondern eine Funktion h wird für alle Punkte verwendet. Dadurch wird Generalisierung und Unabhängigkeit von der Eingabereihenfolge erreicht. Um Rotationsinvarianz zu erreichen, implementiert PointNet eine Koordinatentransformation, welche mittels Matrixmultiplikation durchgeführt wird. Die Transformationsparameter werden ebenfalls durch ein MLP gelernt. Des Weiteren werden die Koordinaten auf den Mittelpunkt reduziert, um Translationsinvarianz zu erreichen (WINIWARTER 2018:20). Das gesamte Netzwerk ist in Abbildung 13 dargestellt und besteht aus einem Klassifikations- und einem Segmentierungsnetzwerk, die sich einen Großteil der Struktur teilen.

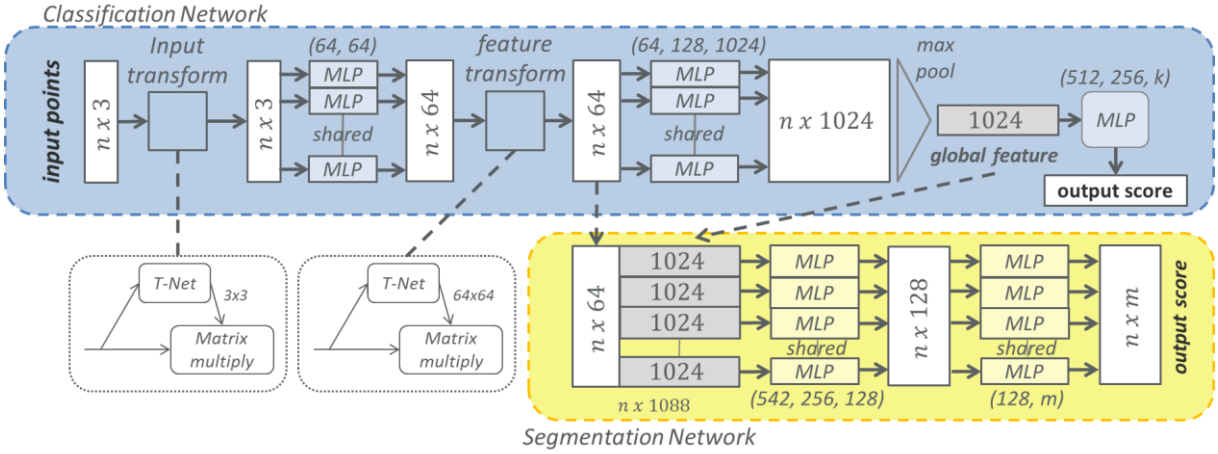


Abbildung 13: PointNet Architektur (QI, C. R. et al. 2016:3)

Das Netzwerk nimmt die n 3D Koordinaten mit möglichen zusätzlichen Attributen als Eingabe, wendet eine Eingabetransformation (*input transform*) an und übergibt die Matrix an das neuronale Netz h. Das neuronale Netz h gibt für jeden Punkt einen Vektor mit 64 Merkmalen zurück, die ebenfalls einer Transformation (*feature transform*) unterzogen werden, welche durch eine 64×64 Matrix repräsentiert wird. Diese Merkmalsmatrix wird an ein weiteres neuronales Netz übergeben, das den Ausgabevektor mit 1024 Merkmalen durch das Max Pooling g aggregiert. Dieser Vektor beschreibt die globalen Merkmale, die aus der gesamten Punktwolke abgeleitet werden. Für Klassifikationszwecke wird der globale Merkmalsvektor einem weiteren MLP mit k Ausgabekanälen übergeben. Mit Hilfe einer Sigmoidfunktion kann so die Wahrscheinlichkeit der Klassenzugehörigkeit der k Klassen abgeleitet werden.

Das priorisierte Ziel von PointNet ist die Klassifikation der gesamten Punktwolke. Da ebenfalls Merkmale für jeden Punkt berechnet werden, können diese Informationen mit dem globalen Merkmalsvektor verknüpft werden. So entsteht ein Merkmalsvektor, der sowohl lokale als auch globale Informationen über jeden Punkt enthält. Diese Matrix wird ebenfalls mehreren MLPs übergeben, sodass eine Wahrscheinlichkeit über die Klassenzuordnung jedes Punktes getroffen wird (QI, C. R. et al. 2016).

PointNet lernt einen Merkmalsvektor für jeden Punkt und aggregiert sie zu einer globalen Signatur. Dieser Ansatz funktioniert in der Regel gut für einfache Formen wie Flugzeuge oder Tassen. Da die Punktmerkmale auf der gesamten Punktwolke gelernt werden, ist PointNet nicht in der Lage lokale Strukturen zu erfassen, was die Fähigkeit limitiert feine Strukturen zu erkennen und komplexe Szenen zu generalisieren. Seit der Veröffentlichung von PointNet wird der direkten Verarbeitung von Punktwolken mehr Aufmerksamkeit geschenkt, wobei viele aktuelle Architekturen auf PointNet basieren.

2.3.8.2. PointNet++

Um den Problemen von PointNet zu entgegnen, entwickelten dieselben Autoren eine erweiterte Version namens *PointNet++* (QI, C. R. et al. 2017). Dabei handelt es sich um ein hierarchisches neuronales Netz, welches PointNet rekursiv auf überlappende Partitionen von Eingangspunkten anwendet. Durch die Verwendung metrischer Raumabstände ist das Netz in der Lage, lokale Merkmale mit zunehmenden Kontextskalen zu lernen. Die Grundidee von PointNet++ basiert auf der Architektur von Convolutional Neural Networks. CNNs verwenden Daten in einem regelmäßigen Gitter und sind in der Lage, Merkmale schrittweise in immer größeren Maßstäben entlang einer Hierarchie mit mehreren Auflösungen, zu erfassen. Neuronen auf niedrigeren Ebenen haben kleinere rezeptive Felder, während Neuronen auf höheren Ebenen größere rezeptive Felder haben. Um dies zu erreichen, teilt PointNet++ die Punktwolke auf der Basis der Distanzmetrik in sich überlappende lokale Regionen auf und extrahiert lokale Merkmale, die feine geometrische Strukturen erfassen. Diese Merkmale werden weiter zu größeren Einheiten gruppiert und prozessiert, um Merkmale auf höherer Ebene zu erzeugen.

Die Architektur von PointNet++ ist in Abbildung 14 dargestellt und besteht aus einer Reihe von *Set-Abstraction* (SA)-Layern, in denen die Merkmale hierarchisch gelernt werden, gefolgt von einer gleichen Anzahl von *Feature-Propagation* (FP)-Layern, die die gelernten Merkmale auf die nächst höhere Auflösung zurückinterpolieren.

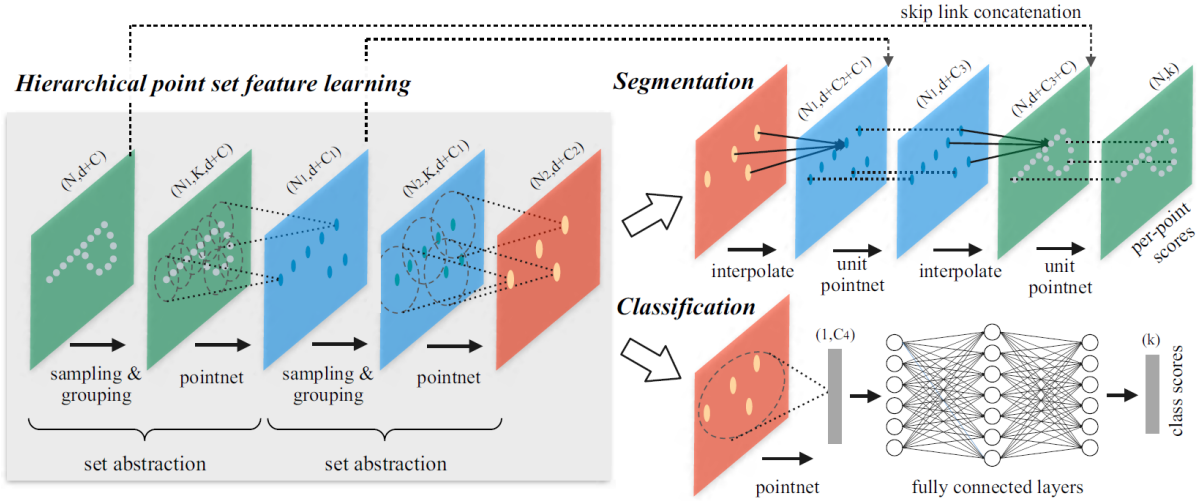


Abbildung 14: Visualisierung der Architektur von PointNet++ (QI, C. R. et al. 2017:3)

Ein SA-Layer erhält die Punktwolke in Form einer $N \times (d + C)$ großen Matrix, bestehend aus N Punkten mit d-dimensionalen Koordinaten und C-dimensionalen Punktmerkmalen. In einem SA-Layer werden drei Schlüssel-Layer ausgeführt: ein Sampling-Layer, ein Gruppierungs-Layer und ein PointNet-Layer.

Der Sampling-Layer selektiert aus einer Punktmenge $\{x_1, \ldots, x_n\}$ mittels *Farthest-Point-Sampling* (FPS) (ELDAR et al. 1997) eine Untermenge von Punkten $\{x_{i_1}, x_{i_2}, \ldots, x_{i_n}\}$ mit N' Punkten, sodass x_{i_j} der am weitesten entfernte Punkt von der Menge $\{x_{i_1}, x_{i_2}, \ldots, x_{i_{j-1}}\}$ bezüglich der übrigen Punkte ist.

Der Gruppierungslayer verwendet die $N \times (d + C)$ Eingabepunktwolke und die Koordinaten der Centroide aus FPS mit der Größe $N' \times d$ und konstruiert lokale Regionen, indem die den Centroiden nächstgelegenen benachbarten Punkte gesucht werden. Die Ausgabe dieses Layers sind Gruppen von Punktmengen, dargestellt durch eine Matrix mit einer Größe von $N' \times K \times (d + C)$, wobei jede der N' Gruppen eine lokale Region darstellt und K Punkte enthält. Benachbarte Punkte werden dabei durch den metrischen Raum bestimmt, indem sogenannte Ball-Abfragen alle Punkte innerhalb eines festen Radius zum Abfragepunkt finden.

Im PointNet-Layer werden die K Punkte jeder der N' lokalen Regionen auf den jeweiligen Centroiden reduziert und an ein Mini-PointNet übergeben, um lokale Regionsmuster zu Vektoren mit C' Merkmalen zu decodieren. Die Merkmalsvektoren jeder Region und die Koordinaten der Centroide werden in Form einer Matrix der Größe $N' \times (d + C')$ ausgegeben (QI, C. R. et al. 2017:3f.).

Um Merkmale in unterschiedlichen Maßstäben zu erhalten und die Informationsdichte zu erhöhen, wird die ausgegebene Punktwolke eines SA-Layers an einen weiteren SA-Layer übergeben. Dabei wird mittels FPS eine Teilmenge der vorherigen Teilmenge selektiert und eine lokale Nachbarschaft gebildet, die an einen weiteren PointNet-Layer übergeben wird. Während die Anzahl der Punkte in jedem SA-Layer reduziert wird, entsteht gleichzeitig eine höhere Informationsdichte. Da für die semantische Segmentierung Merkmale für jeden einzelnen Punkt benötigt werden, müssen die Merkmale der gefilterten Punkte auf die ursprünglichen Punkte propagiert werden. Dazu werden im FP-Layer die abgeleiteten Merkmale der höheren Ebenen auf die ursprüngliche Auflösung der Eingabepunktwolke interpoliert und mit den Merkmalen der jeweiligen Ebene verkettet.

Im Detail adaptiert die Propagationsstrategie eine distanzbasierte Interpolation der drei nächstgelegenen Punkte und layerübergreifende "Skip Links" zur Verkettung der Merkmale (siehe Abbildung 14). In einem FP-Layer wird die $N_l \times (d + C)$ große Merkmalsmatrix eines SA-Layers l zu N_{l-1} Punkten propagiert, wobei N_{l-1} und N_l (mit $N_l \leq N_{l-1}$) Punktmengen der Eingabe und Ausgabe des SA-Layers l sind. Die Propagation erfolgt, wie in Formel 22 dargestellt, durch inverse abstandsgewichtete Interpolation der Merkmalswerte f von N_l Punkten zu den Koordinaten den N_{l-1} Punkten. Standardmäßig werden die drei nächsten Nachbarn ($k = 3$) und $p = 2$ für die Interpolation verwendet.

Formel 22: Inverse distanzgewichtete Merkmalsinterpolation (QI, C. R. et al. 2017:5)

$$f^{(j)}(x) = \frac{\sum_{i=1}^{k} w_i(x) f_i^{(j)}}{\sum_{i=1}^{k} w_i(x)} \quad mit\ w_i(x) = \frac{1}{d(x, x_i)^p},\ j = 1, \dots, C$$

Die interpolierten Merkmale werden dann mit den Punktmerkmalen des entsprechenden SA-Layers verkettet ("skip-linked"). Die verketteten Merkmale werden dann an ein PointNet übergeben, um jeden Merkmalsvektor zu aktualisieren. Dieser Prozess wird wiederholt, bis die Merkmale für die ursprüngliche Eingangspunktmenge propagiert wurden. Ein abschließendes PointNet berechnet die Klassenwahrscheinlichkeiten für jeden Punkt (QI, C. R. et al. 2017:5).

3. Methodik

Vorausgehend wurden die theoretischen Grundlagen erörtert, die für die Erstellung eins Modells zur semantischen Segmentierung von Punktwolken notwendig sind. Zur Generierung eines solchen Modells sind zwei Schritte notwendig: (1) die Generierung einer gelabelten Punktwolke mit Informationen über die semantische Klasse jedes Punktes für das Training des Modells und (2) die Implementierung und das Training des Modells. Im folgenden Abschnitt werden diese Schritte im Detail beschrieben, eine Deep Learning Architektur ausgewählt und eine Implementierung vorgestellt.

3.1. Generierung von Trainingsdaten

Die in Kapitel 2.2.4 genannten Anforderungen gelten auch für Deep-Learning-Modelle zur semantischen Segmentierung von Punktwolken. Daraus folgt, dass zum Training des Modells eine gelabelte Punktwolke mit einer ausreichenden Anzahl von Punkten benötigt wird. Allerdings ist nicht nur die Anzahl der Punkte in der Punktwolke entscheidend, sondern auch die Heterogenität der Punktwolke ist von Bedeutung. Um eine möglichst hohe Generalisierung des Modells zu erreichen, ist es notwendig, eine große Vielfalt von Objekten in den unterschiedlichsten Ausprägungen zu berücksichtigen. Dazu gehören unterschiedliche Gebäudestrukturen (Hochhäuser, Reihenhäuser, Einfamilienhäuser etc.), um die Gebäude korrekt zu segmentieren. Darüber hinaus stellt die korrekte und konsistente Zuordnung jeder Punktklasse eine Herausforderung dar (ZHANG, Q. et al. 2018). Zur Annotation von Punktwolken und damit zur Generierung geeigneter Trainingsdaten stehen verschiedene Methoden zur Verfügung: (1) manuelle Annotation, (2) semiautomatische Annotation durch unüberwachte Verfahren oder (3) Generierung synthetischer Punktwolken.

Bei der manuellen Annotation werden die Punkte manuell ausgewählt und einer Klasse zugeordnet. Rudimentär ist dies mit der Open-Source-Software *CloudCompare* möglich, es gibt aber auch spezielle Annotations-Tools, um den Zeitaufwand zu minimieren. Dazu gehören beispielsweise die kommerzielle *3D Labeling Toolbox* von SUPERVISE.LY (2019) oder auch freie Software wie *PC-Annotate* von IBRAHIM et al. (2021) oder der *Semantic Segmentation Editor* von HITACHI AUTOMOTIVE AND INDUSTRY LABORATORY (2021). Die manuelle Annotation von Punktwolken ist jedoch auch mit spezieller Software zeitaufwändig, sodass die Generierung einer geeigneten Menge an Trainingsdaten für ein Deep Learning Modell nur schwer wirtschaftlich umsetzbar ist (HILDEBRAND 2023). Für die Generierung eines geeigneten Trainingsdatensatzes für Punktwolken aus Luftbildern stellt dieses Verfahren keinen geeigneten Ansatz dar, da die manuellen Arbeitsschritte zu zeitaufwendig sind. Zudem ist im Rahmen dieser Arbeit keine kommerzielle Annotationssoftware verfügbar, die den manuellen Annotationsprozess beschleunigen könnte.

Bei der semiautomatischen Annotation von Punktwolken werden unüberwachte Lernverfahren wie K-Means (siehe Kap. 2.3.6.1) oder DBSCAN (siehe Kap. 2.3.6.2) eingesetzt, um Punkte mit ähnlichen Eigenschaften zu gruppieren. Als Eigenschaften können neben den XYZ-Koordinaten auch Farbinformationen (siehe 2.3.5) oder die in Kapitel 2.3.4 vorgestellten geometrischen Merkmale verwendet werden. Die aus den unüberwachten Algorithmen resultierenden Cluster müssen anschließend manuell der richtigen semantischen Klasse zugeordnet werden. Diese Vorgehensweise kann den Zeitaufwand gegenüber der manuellen Annotation deutlich reduzieren – auch wenn die Wahl des Algorithmus und der optimalen Merkmalskombination oft schwierig und vom jeweiligen Anwendungsfall abhängig ist. Im Vergleich zur manuellen Annotation bietet dieses Verfahren das Potenzial einer deutlichen Effizienzsteigerung bei der Annotation von Punktwolken aus Luftbildern.

Bei der Generierung synthetischer Punktwolken wird ein 3D-Modell der Umgebung verwendet, um mit Hilfe virtueller Laserscanner Punktwolken mit semantischen Informationen zu genieren. Dabei können verschiedene virtuelle Laserscanner implementiert werden, um die verschiedenen Eigenschaften von Laserscannern zu imitieren. Ein Beispiel für ein solches System ist die Implementierung von HILDEBRAND et al. (2022). Ein Großteil dieser Prototypen basiert auf der Simulation von Laserscannern. Eine Pipeline zur synthetischen Generierung von annotierten photogrammetrischen Punktwolken aus Luftbildern wurde beispielsweise von CHEN, M. et al. (2022) vorgestellt. Dieser Workflow verwendet eine GIS-Datenbank als Grundlage für die Generierung einer realitätsnahen virtuellen Umgebung. Aus diesem 3D-Modell werden Bilder entlang eines vordefinierten UAV-Flugmusters gerendert und anschließend mit kommerzieller Photogrammetrie-Software zu einer Punktwolke verarbeitet. Da der von CHEN, M. et al. (2022) vorgestellte Datensatz auf der Simulation einer UAV-Befliegung basiert, ist er für diese Arbeit nicht geeignet. Die Erstellung einer ähnlichen Pipeline für die Simulation einer bemannten Luftbildbefliegung ist denkbar, würde aber den Rahmen dieser Arbeit übersteigen.

3.2. Wahl einer geeigneten Netzarchitektur

Die Wahl einer optimalen Deep-Learning-Architektur für die semantische Segmentierung von Punktwolken variiert je nach Anwendungsfall. Viele punktweise Deep-Learning-Architekturen erzielen zufriedenstellende Ergebnisse für Indoor-Datensätze. Insbesondere für 2,5D-Punktwolken, wie sie durch Airborne Laserscanning oder DIM aus Luftbildern erhoben werden, fehlt es bisher an umfassender Forschung (WIDYANINGRUM et al. 2021:2). Im Kontext der semantischen Segmentierung von photogrammetrischen Punktwolken aus Luftbildern mittels Deep Learning Ansätzen fehlen bisher spezifische Untersuchungen. Es existieren jedoch einige Publikationen, die unterschiedliche Architekturen zur semantischen Segmentierung von ALS-Punktwolken verwenden, die näherungsweise ähnliche Eigenschaften wie DIM-Punktwolken aufweisen.

WIDYANINGRUM et al. (2021) adaptieren das DGCNN (*Dynamic Graph Convolutional Neural Network*) von WANG, Y. et al. (2018) für die semantische Segmentierung von ALS-Punktwolken und erreichen eine Gesamtgenauigkeit von 93,3% für den niederländischen AHN3-Datensatz (*Actueel Hoogtebestand Nederland*) mit vier Klassen. WINIWARTER & MANDLBURGER, G. (2019) untersuchten die Eignung von PointNet++ für die punktweise Klassifikation von ALS-Punktwolken und erreichen eine Gesamtgenauigkeit von 95,8% für urbane Gebiete im Voralberg-Datensatz. Weiterhin evaluierten KADA & KURAMIN (2021) die Anwendung von PointNet++ und KPConv auf ALS-Punktwolken und erreichten Gesamtgenauigkeiten von 93,1% mit PointNet++ und 95,5% mit KPConv erreichten. Ihre Ergebnisse deuten zudem auf eine Verbesserung der Modellleistung durch die Verwendung von zylindrischen Nachbarschaften und Vorinformationen hin.

In dieser Arbeit wird *PointNet++* verwendet, um ein Modell für die semantische Segmentierung von Punktwolken aus photogrammetrischen Luftbildern zu erstellen. Auch wenn PointNet++ mittlerweile in vielen Experimenten von aktuelleren Netzwerken wie *KPConv (THOMAS 2019)*, *ConvPoint* (BOULCH 2019) oder *PointCNN* (LI, Y. et al. 2018) übertroffen wird, bleibt das Modell relevant, da sein Mechanismus zur Merkmalsextraktion und seine Architektur oft ein Baustein für andere punktwolken-basierte Netzwerke sind, z. B. zur Lösung von Objekterkennungsaufgaben (QI, C. R. et al. 2019).

4. Implementierung

Mit den erörterten Grundlagen und der angestrebten Methodik im Hintergrund kann nun mit der Umsetzung des Workflows begonnen werden. Für das Training eines Modells wird zunächst eine geeignete annotierte Punktwolke benötigt. Hier steht ein Datensatz zur Verfügung, der bereits im Rahmen einer Dissertation semantisch segmentiert wurde. Die bestehende Segmentierung des Datensatzes muss zunächst optimiert werden, um dem Modell optimale Trainingsdaten zur Verfügung zu stellen. Anschließend kann ein geeignetes Deep-Learning-Modell ausgewählt und angewendet werden. Im Folgenden wird zum einen der vorhandene Trainingsdatensatz beschrieben und zum anderen der Workflow zur Optimierung der Punktwolke des Testgebietes erläutert. Des Weiteren werden die Implementierung und das Training des Modells beschrieben und die verwendeten Hard- und Softwarekomponenten vorgestellt.

4.1. Annotation der Trainingsdaten

Für interne Testzwecke hält das Landesamt für Geoinformation und Landesvermessung Niedersachsen (LGLN) ein Testgebiet im Raum Hannover-Bemerode vor. Für dieses Gebiet sind diverse Daten verfügbar – darunter Luftbilder, ALS-Punktwolken aus dem Jahr 2016, sowie DIM-Punktwolken aus den Jahren 2016, 2019 und 2022. In Tabelle 1 sind die relevanten Eckdaten der Befliegung aus dem Jahr 2022 aufgeführt: der verwendete Sensor, der Aufnahmezeitpunkt, die erzielte Längs- und Querüberlappung, sowie die erreichte Ground Sample Distance (GSD).

Tabelle 1: Eckdachten der Luftbildaufnahmen im Jahr 2022

	Sensor	**Aufnahme-zeitpunkt**	**Längs- & Querüberlappung**	**Tatsächliche GSD**
2022	UltraCam Eagle M3 431S72402X218442	08.05.2022 (8:58 bis 11:22 UTC)	80,1 – 81,9% / 51,2 – 55,5%	15,1– 15,5 cm

Die ALS-Punktwolken wurden direkt vom ausführenden Unternehmen klassifiziert und durch das LGLN validiert. Die DIM-Punktwolken wurden mit der Software *SURE* von *nFrames* aus den Luftbildern prozessiert. Neben den räumlichen Koordinaten (X, Y, Z) und der Farbinformation (R, G, B)

jedes einzelnen Punktes sind auch weitere Attribute verfügbar: die Punktpräzision ($\sigma \cdot 10.000/GSD$), die Anzahl der Stereomodelle (maximaler Wert ist sieben), die Genauigkeit (keine detaillierte Beschreibung) und der Index des Basisbildes aus dem Matching (NFRAMES o.J.). Die Daten dieses Testgebietes wurden bereits für verschiedene interne Tests sowie für wissenschaftliche Arbeiten wie Masterarbeiten oder Dissertationen verwendet. So wurden die DIM-Punktwolken im Rahmen einer noch unveröffentlichten Dissertation bereits semantisch segmentiert. Die Qualität der Annotation ist jedoch für das Training eines Deep Learning Modells nicht ausreichend, sodass eine weitere Optimierung notwendig ist.

4.1.1. Klassifikation der Punktwolke aufgrund geometrischer Eigenschaften

In der genannten Dissertation wurde die Punktwolke anhand verschiedener geometrischer Eigenschaften semantisch segmentiert. Dabei wurde unterschieden in die Klassen: *Gelände*, *Gebäude*, *Nicht-Gelände* und *Brücken*. In der Arbeit wurden die westlichen 12 km² des Gebietes verwendet, dessen Ausdehnung in Abbildung 15 ersichtlich ist. Das Gebiet zeichnet sich durch seine Heterogenität aus und umfasst sowohl urbane Bereiche als auch Waldgebiete (Seelhorst). Darüber hinaus weist das Gebiet unterschiedliche Bauweisen auf: Hütten in Kleingärten, Ein- und Mehrfamilienhäuser, Hochhäuser, Schulen und auch große Hallen. Ebenso sind Sportanlagen, Straßenbahnen, Baustellen und Autobahnen im Trainingsgebiet vertreten.

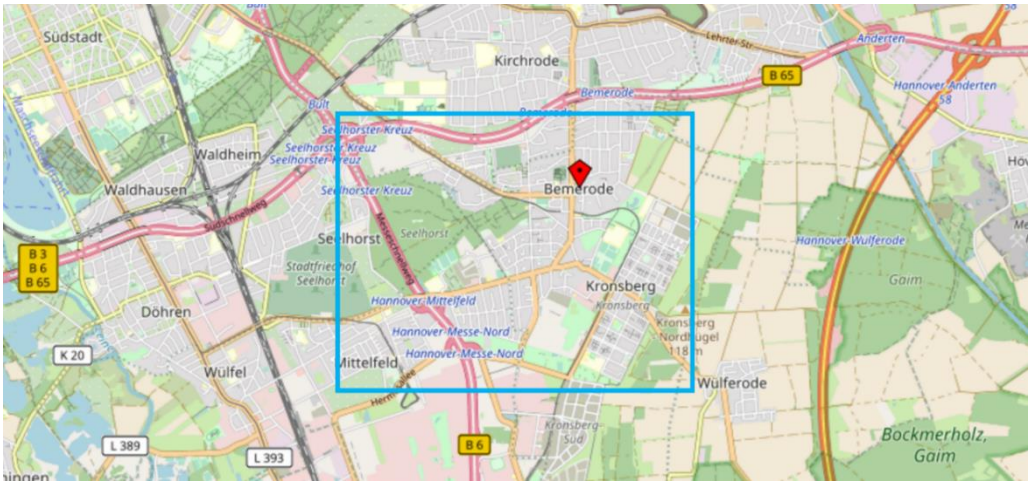


Abbildung 15: Übersichtskarte des Trainingsgebietes (blau umrandet) – Kartengrundlage: © Open Street Map.

Die Annotation der Punktwolken basiert auf geometrischen Eigenschaften und anhand verschiedenen Daten des ALKIS (Amtliches Liegenschaftskataster-Informationssystem). Zur Identifikation der Bodenpunkte wurde aus der vorklassifizierten ALS-Punktwolke aus dem Jahr 2016 ein digitales Geländemodell (DGM) erstellt und die Höhendifferenz Δh zwischen den DIM-Punktwolken und dem DGM berechnet. Alle Punkte mit $-0.2\ m < \Delta h \leq 0.2\ m$ wurden der Klasse *Gelände* zugeordnet. Die Klassen *Gebäude* und *Brücke* wurden durch Überlagerung mit den 2D ALKIS-Objekten *AX_Gebaeude* und *AX_Bruecke* des LGLN erzeugt. Alle übrigen Punkte wurden der Klasse *Nicht-Gelände* zugeordnet.

Obwohl die annotierten Punktwolken einen guten Ausgangspunkt für das Training eines Klassifikators zur semantischen Segmentierung darstellen, sind weitere Optimierungen notwendig. Sowohl die Qualität der Annotation als auch die Richtigkeit der Klassenzuordnung sind verbesserungswürdig. Bereiche, in denen sich das Gelände zwischen 2016 und 2022 durch bauliche Veränderungen stark verändert hat, werden fälschlicherweise als *Nicht-Gelände* klassifiziert. Wurde z. B. Erde für ein Baugebiet aufgeschüttet, wird der Untergrund als *Nicht-Gelände* klassifiziert. Vegetationspunkte oberhalb von Gebäuden und Bodenpunkte unterhalb von Brücken erhalten durch die Verschneidung mit den zweidimensionalen ALKIS-Objekten ebenfalls falsche Klassenzuordnungen. Zusätzlich werden Dachüberstände aufgrund der Gebäudeumringe in ALKIS als *Nicht-Gelände* klassifiziert, da die Gebäudeumringe in ALKIS den Außenwänden der Gebäude entsprechen. Generell ist die Klasse *Nicht-Gelände* für diese Arbeit ungeeignet, da diese Klasse sehr unterschiedliche Objekte enthält, die sich in ihren geometrischen und radiometrischen Eigenschaften stark unterscheiden. Beispielsweise unterscheiden sich Fahrzeuge und Vegetation erheblich in ihrer Ausdehnung und Oberflächenrauhigkeit. Diese Heterogenität innerhalb der gleichen semantischen Klasse wirkt sich negativ auf die Leistungsfähigkeit des DL-Modells aus. Abbildung 16 zeigt einige problematische Bereiche der annotierten Punktwolken. In der oberen Hälfte ist zu sehen, dass Dachüberstände in der Punktwolke fälschlicherweise als Nicht-Gelände klassifiziert werden. In der unteren Hälfte ist zu erkennen, dass sowohl Fahrzeuge auf der Straße als auch Vegetation als *Nicht-Gelände* klassifiziert werden, obwohl sie sehr unterschiedliche geometrische Eigenschaften aufweisen.

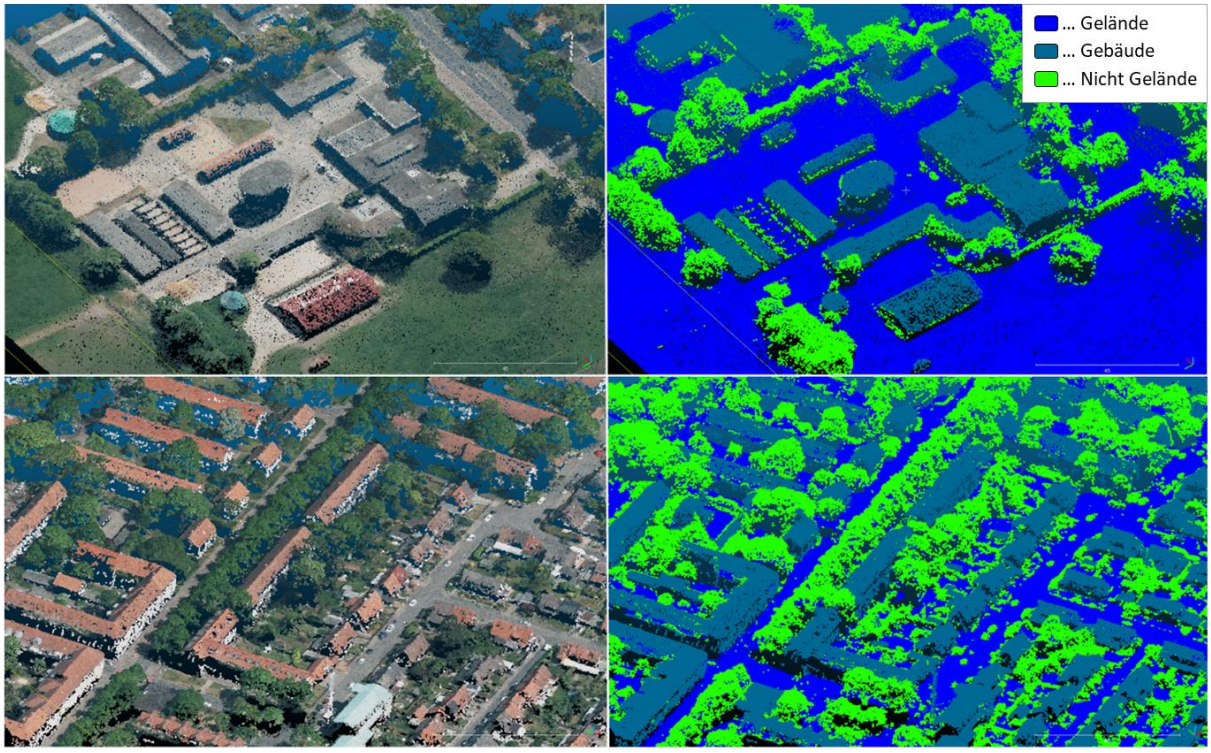


Abbildung 16: Klassifikation durch geometrische Eigenschaften. Links: RGB-Darstellung - rechts: Klassifizierte Punktwolke

Aus den genannten Gründen wird die bestehende klassifizierte DIM-Punktwolke aus 2022 optimiert, um dem Deep-Learning-Modell Trainingsdaten mit dem bestmöglichen Kompromiss aus Qualität und Quantität zur Verfügung zu stellen. Die DIM-Punktwolke aus 2022 enthält ca. 66.700.000 Punkte mit einer Punktdichte von ca. 5 Punkten pro m². Im Rahmen der Optimierung soll zudem die Klasse *Nicht-Gelände* in die Klassen *Vegetation* und *Menschgemacht* unterteil werden. Aus diesem Grund wird die Klasse *Nicht-Gelände* im Folgenden als unklassifiziert angesehen. In der Klasse *Vegetation* sollen sowohl große Bäume und Wälder als auch Hecken und Sträucher zusammengefasst werden. Die Klasse *Menschgemacht* umfasst hauptsächlich Fahrzeuge, aber auch Zäune und Kräne. Um dieses Ziel zu erreichen wurde ein Workflow entwickelt, um eine semiautomatische Optimierung der Punktwolke durchzuführen.

4.1.2. *Workflow zur semiautomatischen Optimierung der klassifizierten Punktwolken*

Um eine zeitaufwändige manuelle Bereinigung der Klassifikation der Punktwolke zu vermeiden, wird ein semiautomatischer Workflow entwickelt, der die Klassenzuordnung optimiert. Der Workflow integriert sowohl unüberwachte als auch überwachte Lernverfahren und ist in Abbildung 17 visualisiert. Die insgesamt 12 km² umfassende Punktwolke wird zunächst in zwölf Patches von je $1 \times 1\ km^2$ Größe unterteilt. Die Bezeichnung der einzelnen Patches richtet sich im Folgenden nach der UTM-Koordinate der nordwestlichen Ecke. Die gewählten Parameter und Schwellwerte der Verfahren werden aufgrund von Erfahrungswerten auf Basis einzelner Patches gewählt.

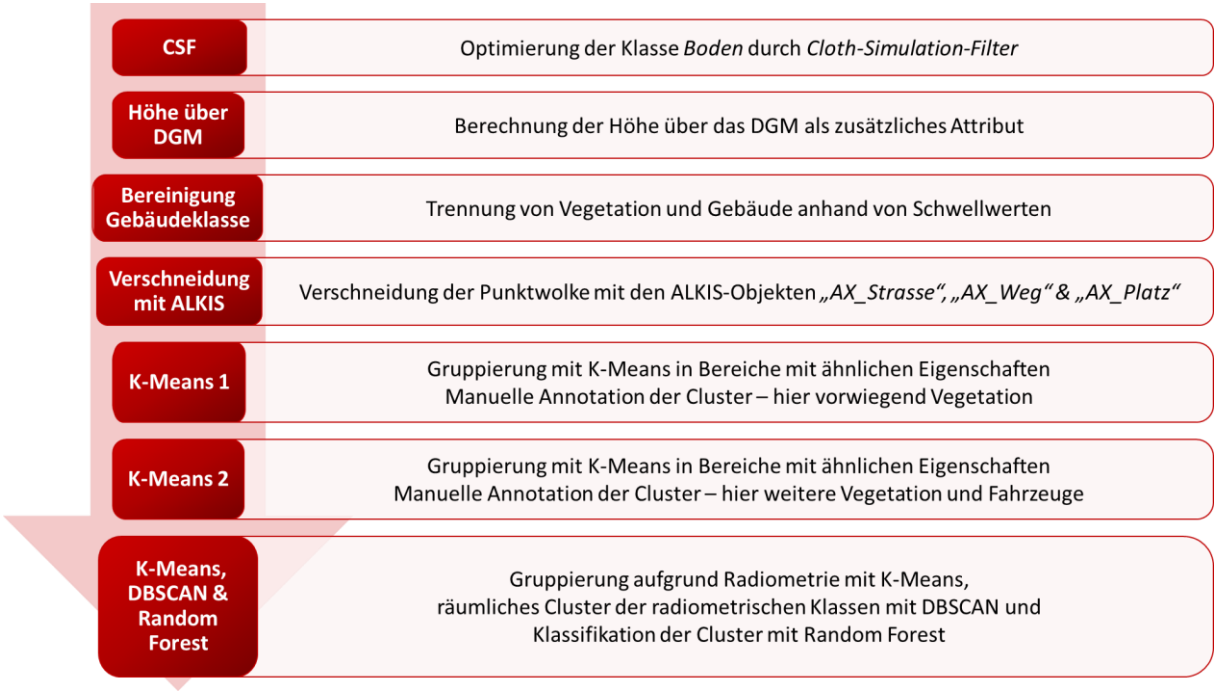


Abbildung 17: Workflow zur Optimierung der Punktwolkenklassifikation

Der erste Schritt besteht in der Anwendung des *Cloth-Simulation-Filter* zur Optimierung der Bodenklasse, welcher in der Software *CloudCompare* implementiert ist. Anschließend wird die relative Höhe jedes Punktes über ein DGM berechnet und der Punktwolke als skalares Feld hinzugefügt. Die Bereinigung der Gebäudeklasse erfolgt durch die Anwendung von Schwellenwerten, welche die Vegetationspunkte von Gebäudepunkten trennen. Da sich Fahrzeuge hauptsächlich auf der Straße befinden, wird die Punktwolke mit den ALKIS-Objekten *AX_Strasse*, *AX_Weg* und *AX_Platz* verschnitten, um für jeden Punkt zu bestimmen, ob er sich innerhalb oder außerhalb der Straße befindet ($inside\_road = 1$ oder $inside\_road = 0$). Anschließend werden verschiedene unüberwachte und überwachte Segmentierungsverfahren verwendet, um die unklassifizierten Punkte der richtigen Klasse zuzuordnen. Dazu kommen sowohl die unüberwachten Verfahren K-Means und DBSCAN (siehe 2.3.6.1 und 2.3.6.2) zum Einsatz als auch das überwachte Verfahren Random Forest (siehe 2.3.7).

4.1.2.1. Cloth-Simulation-Filter

Der erste Schritt der Optimierung besteht in der Verbesserung der Bodenklasse mit Hilfe des CSF, da durch bauliche Veränderungen einige Bereiche fälschlicherweise nicht der Bodenklasse zugeordnet wurden. Eine solche fehlerhafte Zuordnung ist in Abbildung 18 veranschaulicht. Auf der linken Seite der Abbildung ist die RGB-Punktwolke dargestellt, während in der Mitte die Klassifikation vor der Optimierung durch den CSF zu sehen ist.

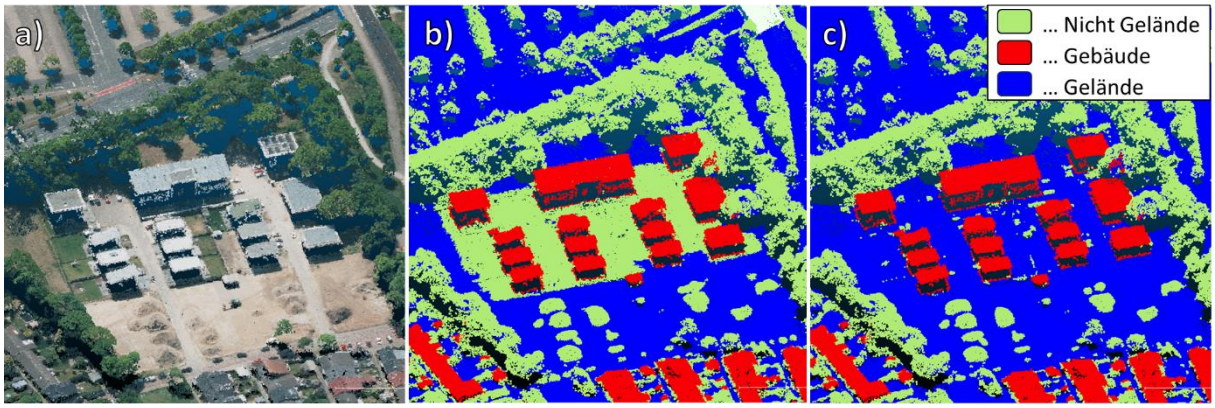


Abbildung 18: Fehlerhafte Geländeklassifikation aufgrund baulicher Änderungen.
a): RGB. b): Klassifikation vor CSF. c): Klassifikation nach CSF.

Zur Durchführung der Optimierung werden die zwölf Patches einzeln in CloudCompare importiert und anschließend mit Hilfe des CSF in Boden- und Nichtbodenpunkte unterteilt. Bei der Implementierung in CloudCompare wird zwischen generellen und erweiterten Parametern unterschieden. Zu den generellen Parametern gehören die Szeneneinstellungen, bei denen der Benutzer den Szenentyp der Punktwolke ("Steile Neigung", "Relief" oder "Flach") festlegt, sowie das *Slope Postprocessing*, bei dem festgelegt wird, ob ein Postprocessing für steile Neigungen durchgeführt werden soll. Zu den erweiterten Parametern gehören die Rasterauflösung, die maximale Anzahl der Iterationen und der

Klassifizierungsschwellenwert. Die Tuchauflösung wird für die Prozessierung der zwölf Patches auf 2 m festgelegt. Die maximale Anzahl der Iterationen und der Klassifizierungsschwellenwert werden auf die Standardwerte von 300 Iterationen und 0,5 m gesetzt. Die anderen Parameter werden für jedes Patch individuell festgelegt, abhängig von den Gegebenheiten in der Punktwolke. Für Gebiete mit Böschungen an Autobahnen oder Brücken erzielt der Szenentyp *Steile Neigung* die besseren Ergebnisse – flache Gebiete werden als ebensolche definiert. Können auch mit dem Szenentyp *Steile Neigung* nicht alle Bereiche als Boden identifiziert werden, wird zusätzlich die Nachbearbeitung aktiviert. In stark bewaldeten Gebieten versagt der Filter, da hier nur wenige oder gar keine Bodenpunkte vorhanden sind. In solchen Fällen ist eine manuelle Extraktion der Vegetationspunkte erforderlich, um anschließend CSF auf die verbleibende Punktwolke anzuwenden. Bei der Prozessierung mit dem Szenentyp *steiles Gelände* werden auch Brücken als Boden klassifiziert, sodass diese manuell aus der Klasse entfernt werden müssen. Gelegentlich werden auch Gebäudepunkte irrtümlich als Boden klassifiziert – z. B., wenn die Größe des Daches zwei- bis dreimal größer als die Tuchauflösung ist, oder wenn Punkte am Sockel des Gebäudes liegen. Aus diesem Grund werden Gebäudepunkte als ebensolche beibehalten und keine Gebäudepunkte in Bodenpunkte umklassifiziert. Die verbleibenden Punkte, die durch den CSF als Boden klassifiziert werden, werden als ebensolche umklassifiziert. Ein beispielhaftes Ergebnis dieses Vorgangs ist in Abbildung 18 c) dargestellt. Auf diese Weise werden falsch klassifizierte Punkte effizient detektiert und korrigiert.

4.1.2.2. Berechnung der Höhe über DGM

Im Vergleich zu den X- und Y-Koordinaten erweist sich die Höheninformation, d. h. die Z-Koordinate eines Punktes, als ein entscheidendes Attribut für die Klassifikation. Insbesondere für die Klassifikation von Bodenpunkten, aber auch für die Klassifikation von Gebäuden und Vegetation spielt die Z-Koordinate eine entscheidende Rolle. So kann die absolute Höhe ein geeigneter Schwellenwert sein, um Gebäude von Bäumen zu trennen (wie in 4.1.2.3). Die alleinige Verwendung der absoluten Höheninformation ist jedoch problematisch, da ein Gebäude auf einem Hügel höhere Z-Koordinaten aufweist als ein gleich hohes Gebäude in einer Senke. Zur Lösung dieses Problems wird die Berechnung der relativen Höhe über dem Boden vorgeschlagen. Dazu wird aus den bereits klassifizierten ALS-Punktwolken ein DGM generiert und anschließend die Höhe jedes Punktes über dem DGM berechnet.

Zur Generierung des DGM werden zunächst alle Bodenpunkte aus der ALS-Punktwolke aus dem Jahr 2016 importiert. Anschließend wird ein Raster mit einer vordefinierten Rastergröße von 1 m initialisiert. Die Z-Koordinaten aller Punkte innerhalb einer Rasterzelle werden gemittelt. Das resultierende Raster enthält somit für jede Rasterzelle gemittelte Höhenwerte, aber auch einige Nullwerte, da nicht alle Rasterzellen zugehörige Bodenpunkte enthalten, wie in Abbildung 19 links beispielhaft dargestellt. Diese fehlenden Werte werden durch lineare Interpolation eliminiert. Ein vollständiges DGM ist rechts

in Abbildung 19 dargestellt. Das vollständige Jupyter-Notebook für die DGM-Berechnung ist in Anlage A: *Jupyter-Notebooks* zu finden.

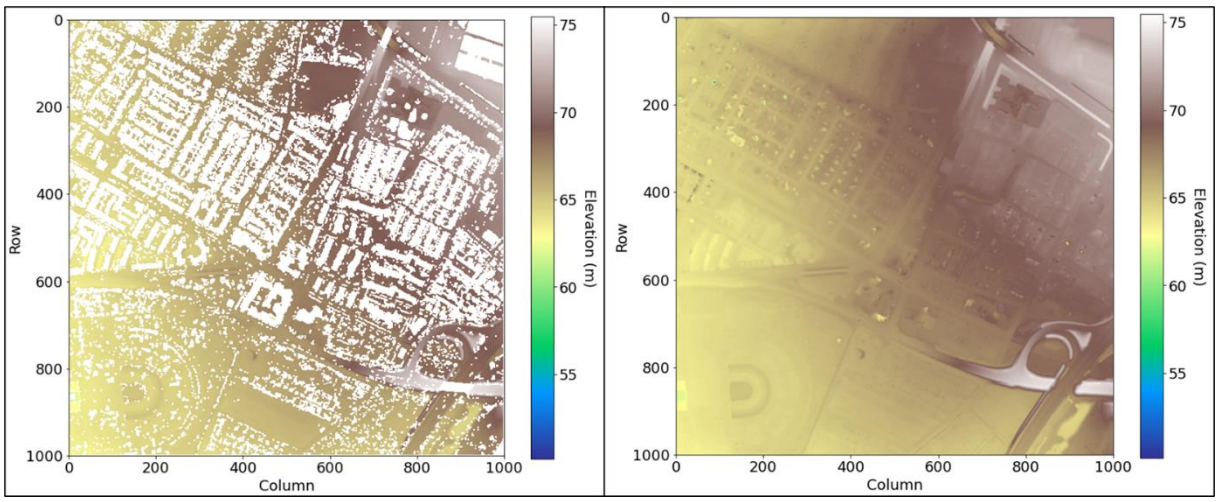


Abbildung 19: Berechnung des DGM.
Links: mittlere Z-Koordinate jedes Pixels – rechts: DGM mit linearer Interpolation der fehlenden Werte.

Die Berechnung der relativen Höhen für jeden einzelnen Punkt erfolgt in einem separaten Jupyter-Notebook, welches ebenfalls in Anlage A: *Jupyter-Notebooks* zu finden ist. In einem ersten Schritt wird das jeweilige berechnete DGM importiert und anschließend eine *Delaunay-Triangulation* (LAWSON 1972) angewendet, um die DGM-Höhe für jeden Punkt zu interpolieren. Anschließend wird die zugehörige Punktwolke importiert und die Differenz zwischen der Z-Koordinate und der interpolierten DGM-Höhe berechnet.

Diese berechnete Höhendifferenz wird als zusätzliches skalares Feld zu jeder Punktwolke hinzugefügt und gespeichert. Eine Veranschaulichung des skalaren Feldes, das die Höhe über dem DGM repräsentiert, ist in Abbildung 20 gezeigt.

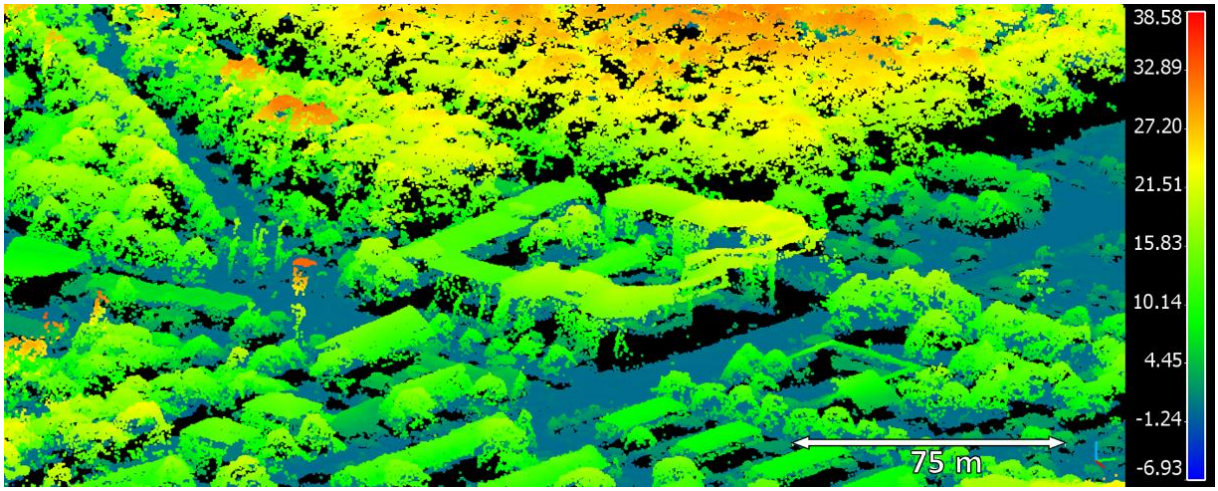


Abbildung 20: Visualisierung des skalaren Feldes der relativen Höhe über DGM. Höhenangabe in Meter.

4.1.2.3. Bereinigung der Gebäudeklassen

Die klassifizierte Gebäudeklasse enthält aufgrund der Verschneidung mit den 2D-Grundrissen der Gebäude diverse Fehlklassifizierungen, insbesondere wenn die Vegetation über die Gebäude hinausragt. In Abbildung 21 sind beispielhaft einige dieser Fehlklassifizierungen türkis umrandet dargestellt.

Es stehen verschiedene Ansätze zur Verfügung, diese Fehler automatisiert zu detektieren und den richtigen Klassen zuzuordnen. Im Rahmen dieser Arbeit werden zunächst die in Kapitel 2.3.4 und 2.3.5 beschriebenen geometrischen und radiometrischen Merkmale berechnet, da anzunehmen ist, dass sich die Merkmale für Vegetations- und Gebäudepunkte signifikant voneinander unterscheiden.

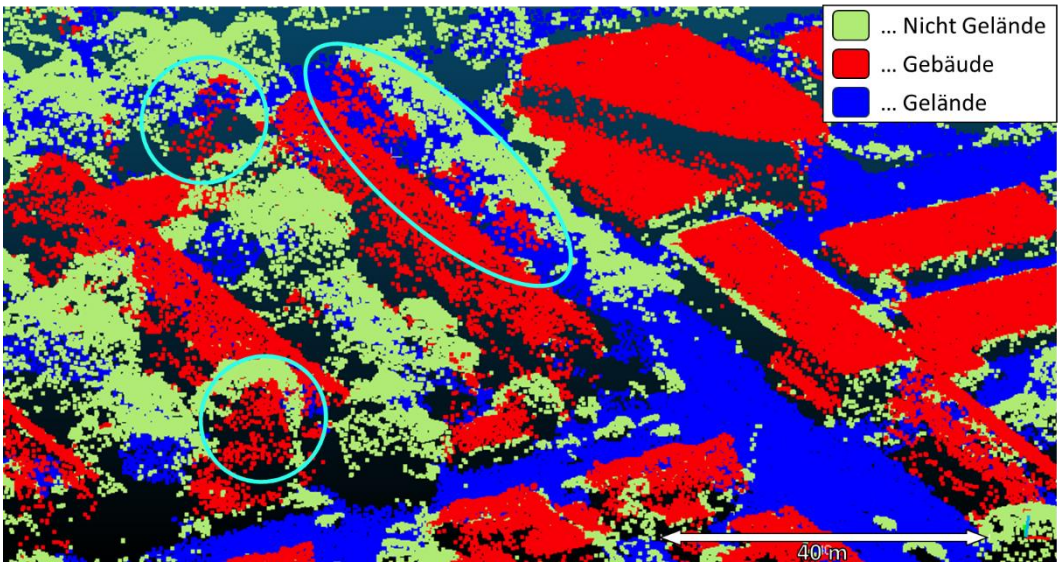


Abbildung 21: Fehlerhafte Klassifikation von Vegetationspunkten oberhalb von Gebäuden

Ein erster Versuch bestand in der Trennung der Klassen mittels K-Means unter Verwendung verschiedener Merkmalskombinationen. Trotz Variation der Clusteranzahl und verschiedener Merkmalskombinationen führte dieser Ansatz nicht zu zufriedenstellenden Ergebnissen. Eine alternative Methode zur Bereinigung der Gebäudeklasse besteht in der Festlegung bestimmter Schwellenwerte, die die Gebäudepunkte von den Vegetationspunkten trennen. Um geeignete Schwellwerte zu bestimmen, werden für einen einzelnen Patch verschiedene Merkmalsintervalle bestimmt, die die Gebäudepunkte herausfiltern. Durch Kombination verschiedener Intervalle bleiben schließlich nur die Vegetationspunkte übrig. Die beste Trennung wird durch die Verwendung der folgenden Merkmale erreicht: (1.) Höhe über DGM, (2.) Farbwert (Hue), (3.) Sättigung (Saturation) und (4.) Standardabweichung der Z-Koordinate innerhalb der Nachbarschaft. Da Vegetationspunkte ausschließlich über Gebäuden auftreten, liegen sie immer mindestens $4\,m$ über dem DGM ($\Delta Z > 4{,}0\,m$). Abbildung 22 a) zeigt eine Filterung anhand dieses Schwellwertes: Punkte, die diese Bedingung erfüllen, sind blau, herausgefilterte Punkte grau dargestellt. Die Farbwerte des HSV-Farbraums werden ebenfalls zur Bestimmung geeigneter Schwellwerte verwendet. Für den Farbwert (Hue) wurde ein Intervall von $[0.17; 0.49[$ und für die Sättigung (Saturation) ein Intervall von $[0.16; 0.60[$ festgelegt.

Diese Filterung ist ebenfalls in Abbildung 22 b) und c) dargestellt. Zusätzlich weisen Vegetationspunkte keine glatte Oberfläche auf, weshalb die Standardabweichung der Z-Koordinate (σ_Z) als geeigneter Schwellwert fungiert. Die lokale Nachbarschaft zur Berechnung der Standardabweichung in Z umfasst die 80 nächsten Nachbarn. Hier wurde der Schwellenwert auf $\sigma_Z > 0.2\ m$ festgelegt. Abbildung 22 d) zeigt die Filterung mit diesem Schwellwert. Durch die Kombination der beschriebenen Schwellenwerte verbleiben lediglich die gesuchten Vegetationspunkte, die dann entsprechend umklassifiziert werden.

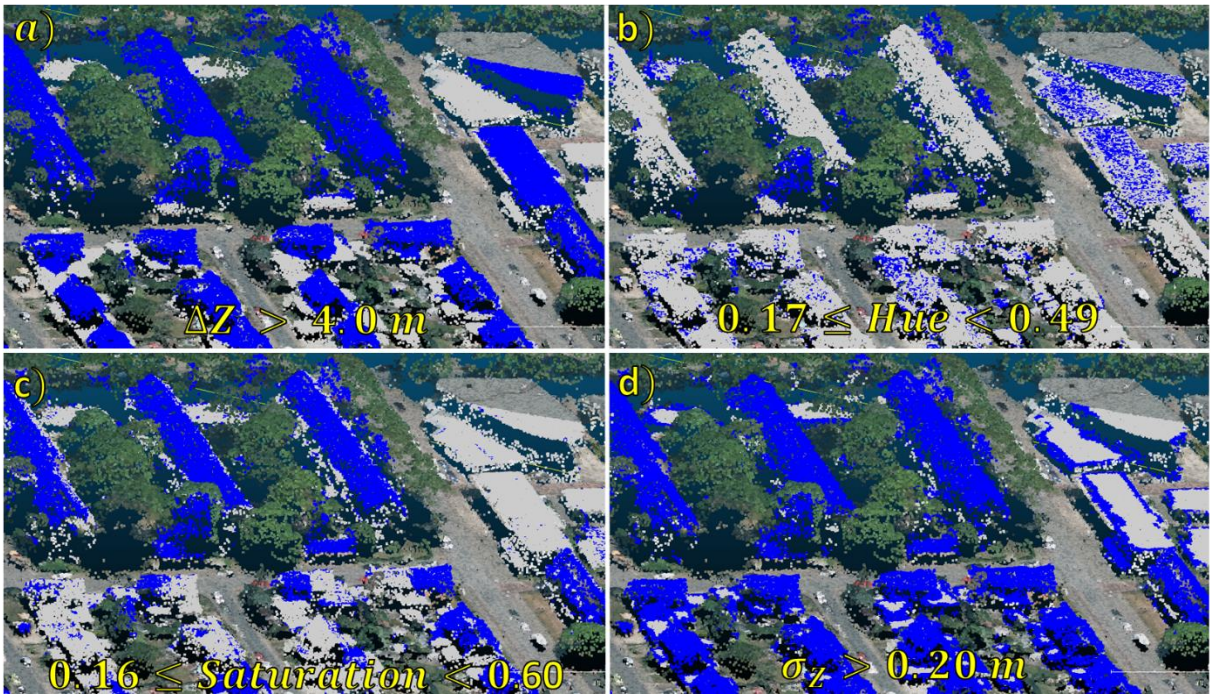


Abbildung 22: Filterung der Gebäudeklasse anhand verschiedener Schwellwerte.
Blau: verbleibende Punkte – grau: herausgefilterte Punkte
a): Höhe über DGM $> 4\ m$*; b):* $0.17 \leq Hue < 0.49$*; c):* $0.16 \leq Saturation < 0.6$*; d):* $\sigma_Z > 0.2$

In Abbildung 23 sind die umklassifizierten Vegetationspunkte in rot und die verbleibenden Gebäudepunkte in blau dargestellt. Diese Vorgehensweise eliminiert die meisten Fehlklassifikationen. Dennoch können vereinzelt Punkte verbleiben, die nach wie vor fehlerhaft klassifiziert sind – wie beispielhaft in Abbildung 23 b) zu sehen. Die geringe Anzahl der verbleibenden Fehlklassifizierungen ist für das weitere Vorgehen vernachlässigbar.

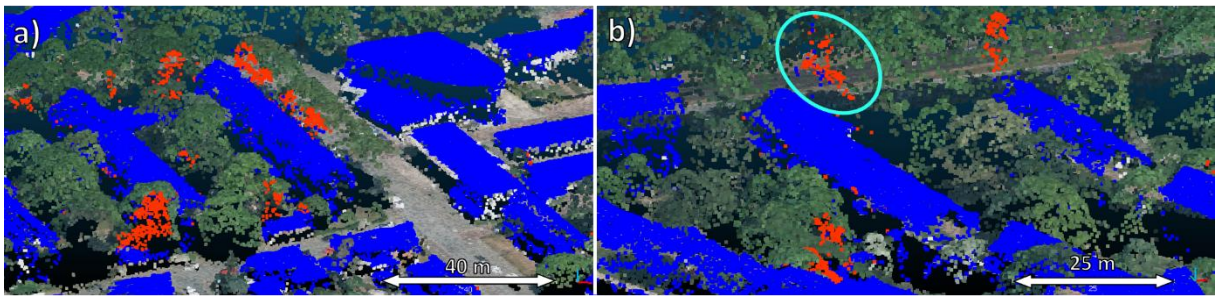


Abbildung 23: Bereinigung der Gebäudeklasse anhand von Schwellwerten.

Die beschriebene Methodik zur Bereinigung der Gebäudeklasse wird für sämtliche Patches durchgeführt. Nach Abschluss der Bereinigung erfolgt eine visuelle Kontrolle in der Software CloudCompare. Die ermittelten Schwellwerte liefern auch für die übrigen Punktwolken zufriedenstellende Ergebnisse. Einzelne fehlerhafte Zuordnungen werden im Rahmen der visuellen Kontrolle manuell bereinigt, sodass die Gebäudeklasse eine ausreichende Qualität für die weitere Verwendung aufweist. Das gesamte Jupyter-Notebook zur Bereinigung der Gebäudeklasse ist in Anlage A: *Jupyter-Notebooks* zu finden.

4.1.2.4. Verschneidung mit Straßenobjekten aus ALKIS

In Vorbereitung auf das anschließende K-Means-Clustering ist eine Unterteilung der Punktwolke in *Straße* und *Nicht-Straße* sinnvoll, da Fahrzeuge überwiegend auf der Straße anzutreffen sind. Zur Abgrenzung der Straße werden die ALKIS-Objekte *AX_Strassenverkehr*, *AX_Weg* und *AX_Platz* verwendet. Die Definitionen der Objekte nach dem ALKIS-Objektartenkatalog (ARBEITSGEMEINSCHAFT DER VERMESSUNGSVERWALTUNGEN DER LÄNDER DER BUNDEREPUBLIK DEUTSCHLAND 2018) sind in Tabelle 2 dargestellt.

Die beschriebenen ALKIS-Objekte werden im *geojson*-Format heruntergeladen und in ein Jupyter-Notebook importiert. Daraufhin erfolgt eine Verschneidung der zweidimensionalen Polygone mit der Punktwolke, wobei nur Punkte selektiert werden, deren XY-Koordinaten innerhalb der Polygone liegen. Diese Punkte werden mit dem Attributwert $inside\_road = 1$ gekennzeichnet, während verbleibenden Punkte den Attributwert $inside\_road = 0$ erhalten. Das so bearbeitete zusätzliche Attribut wird den Punktwolken als zusätzliches skalares Feld hinzugefügt und gespeichert, sodass es im nächsten Schritt in dem K-Means-Clustering verwendet werden kann. Das Jupyter-Notebook zur Verschneidung mit den ALKIS-Objekten befindet sich in Anlage A: *Jupyter-Notebooks*.

Tabelle 2: Zur Verschneidung verwendete ALKIS-Objekte und deren Definition (ARBEITSGEMEINSCHAFT DER VERMESSUNGSVERWALTUNGEN DER LÄNDER DER BUNDEREPUBLIK DEUTSCHLAND 2018)

ALKIS-Objekt:	**Definition:**
AX_ Strassenverkehr	umfasst alle für die bauliche Anlage Straße erforderlichen Flächen, sowie die dem Straßenverkehr dienenden bebauten und unbebauten Flächen.
AX_Weg	umfasst alle Flächen, die zum Befahren und/oder Begehen bestimmt sind. Zur Wegfläche gehören auch Seitenstreifen und Gräben zur Wegentwässerung.
AX_Platz	ist eine innerörtliche Verkehrsfläche oder eine ebene, befestigte oder unbefestigte Fläche, die bestimmten Zwecken dient (z. B. Verkehr, Parkplätze, Märkte oder Festveranstaltungen)

4.1.2.5. K-Means 1

Nach der Optimierung der Bodenklasse mittels CSF und der Gebäudeklasse mittels Schwellwertverfahrens liegt der Fokus auf der Unterteilung der nicht klassifizierten Punkte in die Klassen *Vegetation* und *Menschgemacht*. Dazu wurden bereits zusätzliche Attribute wie die Höhe über dem DGM berechnet und eine Unterteilung der Punktwolken in innerhalb und außerhalb von Straßenflächen vorgenommen. Die Klasse Vegetation soll Bäume, Sträucher und Hecken enthalten, während die Klasse Menschgemacht hauptsächlich Fahrzeuge, aber auch andere menschgemachte Objekte wie Zäune und Kräne enthalten soll. Die nicht klassifizierten Punkte enthalten jedoch zusätzlich weitere falsch zugeordnete Boden- und Gebäudepunkte, die separiert werden müssen. Als Ansatz wird hier das K-Means-Clustering gewählt, da es in der Lage ist, Bereiche mit ähnlichen Eigenschaften zu gruppieren. Dieser Ansatz ist in der Punktwolkenverarbeitung weit verbreitet – ein praktisches Implementierungsbeispiel wird von POUX (2022) erläutert. Für die erste semiautomatische Annotation, der nicht klassifizierten Punkte, wird ein Workflow entwickelt, der in Abbildung 24 dargestellt ist und im Folgenden näher beschrieben wird.

Als Merkmale für K-Means kommen neben den X-, Y- und Z-Koordinaten der Punkte auch andere Attribute der Punktwolke wie die Punktgenauigkeit oder die Anzahl der Stereomodelle in Frage. Um dem Algorithmus weitere geeignete Attribute zur Verfügung zu stellen, wird zunächst der Farbraum von RGB nach HSV transformiert, um eine intuitivere Farbbeschreibung zu erhalten. Die HSV-Farbwerte sind die ersten Merkmale, die für ein Clustering in Frage kommen.

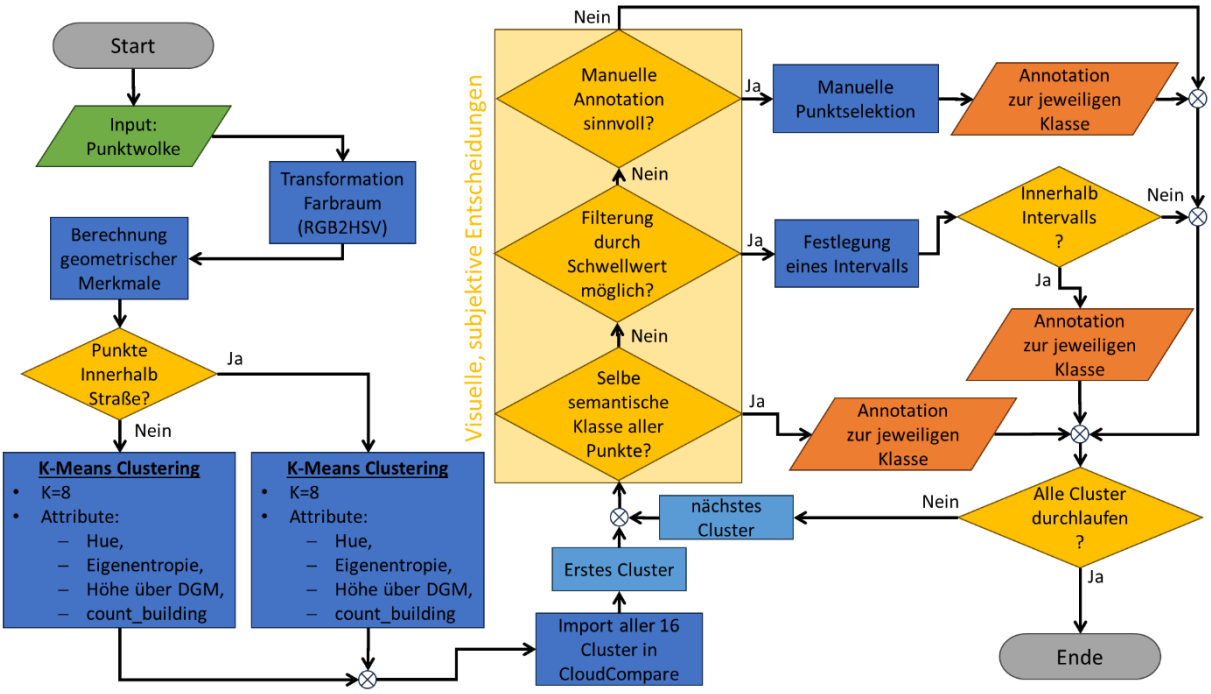


Abbildung 24: Prozessablauf des ersten K-Means Clusterings zur semiautomatischen Annotation der Nicht-Gelände-Klasse

Zusätzlich sollen die in 2.3.4 beschriebenen geometrischen Merkmale berechnet werden. Dazu wird die lokale Nachbarschaft mittels kNN definiert. Zur Steigerung der Performance wird zunächst ein Kd-Tree aus der Punktwolke erzeugt, um schnelle Abfragen zu ermöglichen. Die Auswahl der Anzahl der Punktnachbarn ist jedoch nicht trivial. Obwohl adaptive Methoden zur optimalen Nachbarschaftsberechnung existieren (siehe 2.3.3), wird hier aufgrund des hohen Rechenaufwandes auf solche Methoden verzichtet und eine feste Anzahl von Punktnachbarn definiert. Um befriedigende Ergebnisse zu erzielen, werden die Berechnungen der geometrischen Merkmale mit verschiedenen Nachbarschaftsgrößen durchgeführt und visuell dargestellt. In Abbildung 25 ist beispielhaft die berechnete Eigenentropie mit verschiedenen Nachbarschaftsgrößen dargestellt: links mit 50 Punktnachbarn, in der Mitte mit 100 Punktnachbarn und rechts mit 170 Punktnachbarn. Die unterschiedlichen Farbskalen in den Abbildungen weisen darauf hin, dass gleiche Farben nicht gleiche Werte bedeuten. Es ist zu erkennen, dass die Eigenentropie bei der Wahl von 50 Punktnachbarn kleinteiliger und damit verrauschter erscheint. Bei 100 und 170 Punktnachbarn ist das Ergebnis glatter und plausibler. Ähnliches ist auch bei anderen geometrischen Merkmalen zu beobachten. Da eine höhere Anzahl von Punktnachbarn mit einer höheren Rechenzeit einhergeht, wird die Anzahl der Punktnachbarn für die weiteren Berechnungen auf $K = 100$ festgelegt.

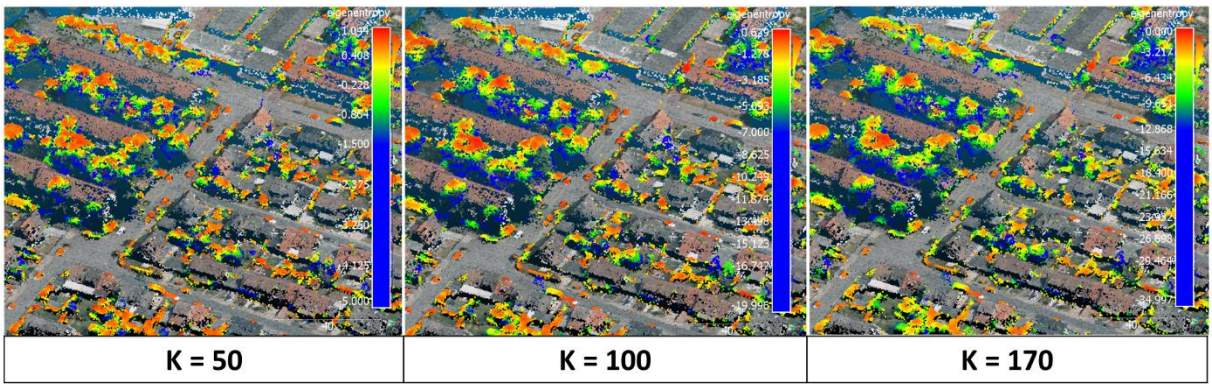


Abbildung 25: Berechnung der Eigenentropie anhand verschiedener Nachbarschaftsgrößen. K definiert die absolute Anzahl an Punktnachbarn. In den Abbildungen wurden verschiedene farbliche Skalierungen verwendet.

Unter Verwendung der definierten Nachbarschaften werden verschiedene geometrische Merkmale berechnet, darunter Rauheit, lokale Punktdichte und Standardabweichung der Z-Koordinate, aber auch eigenwertbasierte Formmerkmale: Linearität, Planarität, Streuung, Omnivarianz, Anisotropie, Eigenentropie, Summe der Eigenwerte und Änderung der Krümmung. Da die Optimierung der Gebäudeklasse bereits durchgeführt wurde, wird zusätzlich die Häufigkeit von Gebäudepunkten in der lokalen Nachbarschaft (Attribut *count_building*) berechnet. Mit Hilfe dieses Attributs können Punkte ausgewählt werden, die sich in unmittelbarer Nähe von bereits klassifizierten Gebäuden befinden. Die berechneten Attribute bilden eine geeignete Grundlage für den K-Means-Algorithmus. Die Auswahl der Merkmale für den K-Means-Algorithmus ist jedoch keine triviale Aufgabe. Es ist wichtig, Merkmale auszuwählen, die tatsächlich relevante und informative Informationen über die Klassenzugehörigkeit

enthalten. Gleichzeitig nimmt die Leistungsfähigkeit von K-Means ab, wenn die Dimensionalität der Merkmale zu groß wird. Dies wird als sogenannter *„Fluch der Dimensionalität“* bezeichnet. Denn mit zunehmender Dimensionalität werden die Abstände zwischen den Datenpunkten tendenziell kleiner. Dies stellt insbesondere für das K-Means-Clustering ein Problem dar, da dieses abstandsbasierte Metriken verwendet, um ähnliche Punkte für die Clusterbildung zu identifizieren (SINGH 2021). Daher ist es wichtig eine angemessene Kombination von Merkmalen für das Clustering zu finden. Hierbei wird zunächst die Korrelation zwischen den einzelnen Attributen betrachtet, welche in Abbildung 26 dargestellt ist. Dies hilft bei der Identifikation von Attributen, die möglicherweise eine hohe Ähnlichkeit aufweisen und womit zur Reduktion der Dimensionalität beitragen können.

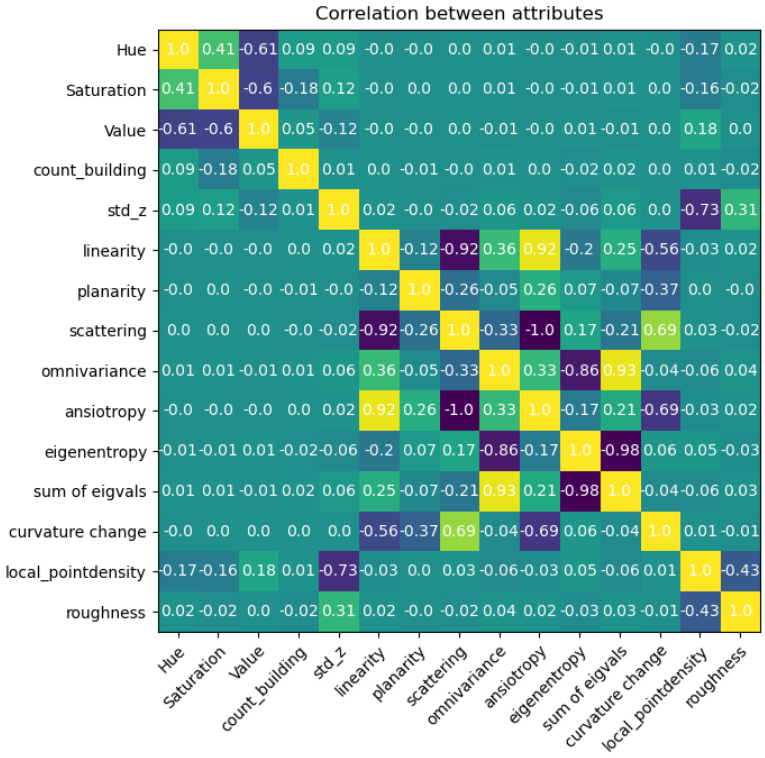


Abbildung 26: Visualisierung der Korrelation zwischen den berechneten Attributen

Es wird deutlich, dass einige Attribute stark miteinander korrelieren, was darauf hindeutet, dass sie keinen zusätzlichen Informationsgehalt zur Beschreibung der Daten beitragen. Zum Beispiel korreliert die Linearität stark mit der Streuung und der Anisotropie. Ebenso korreliert die Eigenentropie stark mit der Omnivarianz und der Summe der Eigenwerte. Die Standardabweichung der Z-Koordinate korreliert stark mit der lokalen Punktdichte. Da die Berechnung der lokalen Punktdichte im Gegensatz zur Standardabweichung der Z-Koordinate rechenintensiv ist, kann auf die Berechnung der lokalen Punktdichte in den weiteren Patches verzichtet werden. Im Gegensatz dazu haben Attribute, die schwach miteinander korrelieren, eine höhere Aussagekraft über die Klassenzugehörigkeit. Weiterhin ist zu erkennen, dass zwischen den Farbattributen eine hohe Korrelation auftritt. Dies ist jedoch zu erwarten, da diese direkt voneinander abhängig sind. Trotz der Korrelationen bilden diese Attribute eine wichtige Grundlage für das anschließende Clustering.

Um zu überprüfen, ob die Merkmale tatsächlich geeignet sind, die Punkte in die semantischen Klassen einzuteilen, werden sie visuell dargestellt und anschließend bewertet. Ein Beispiel hierfür ist der Farbwert, wie in Abbildung 27 dargestellt. Die Darstellung verdeutlicht, dass Gebäudeteile niedrige Farbwerte aufweisen (blau) aufweisen, während Vegetation eher mittlere Farbwerte (grün) hat. Dies ermöglicht eine deutliche Unterscheidung der beiden Klassen.

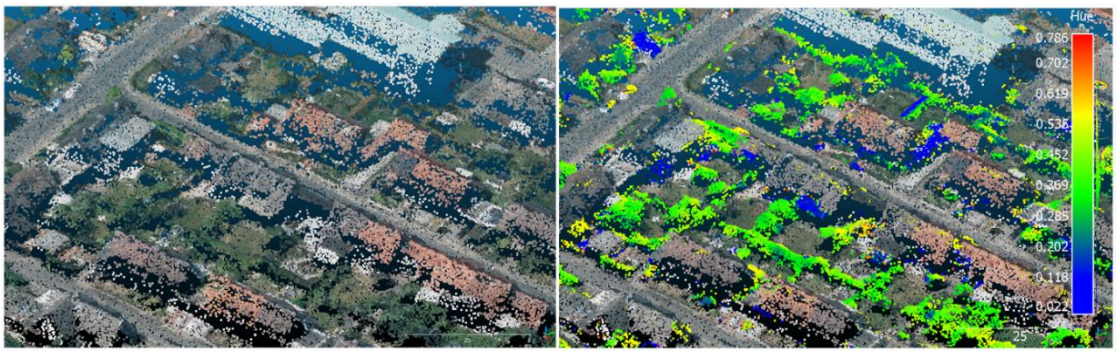


Abbildung 27: links: RGB-Punktwolke; rechts: RGB + Farbliche Darstellung des Farbwerts der Nicht-Gelände Punkte

Vor der Übergabe der Merkmale an den Clustering-Algorithmus ist eine Vorprozessierung der Daten erforderlich. Aufgrund der unterschiedlichen Einheiten der Attribute ist eine Normalisierung der Daten notwendig. Dazu werden alle Werte eines Merkmals in eine Intervall von [0,1] transformiert (HANEKE et al. 2021:125). Vor der Skalierung müssen jedoch Ausreißer aus Daten entfernt werden. Dazu werden, wie in Formel 23 dargestellt, alle Werte x eines Attributs i mit ihrem Mittelwert $\bar{x}_i$ und der zugehörigen Standardabweichung s_{x_i} standardisiert. Anschließend werden alle Merkmale deren standardisierte Zufallsvariable größer als vier ist, als Ausreißer O gekennzeichnet.

Formel 23: Definition aller Ausreißer durch Standardisierung

$$O_i = \left\{ x_i \mid \frac{x_i - \bar{x}_i}{s_{x_i}} > 4 \right\}$$

Nachdem Ausreißer aus den Daten entfernt und die Attributwerte normalisiert sind, können die Daten dem K-Means Algorithmus übergeben werden. Die Anzahl der Cluster K, die zur Initialisierung benötigt wird, wird anhand der Ellenbogen-Methode auf $K = 8$ festgelegt (siehe Abbildung 28).

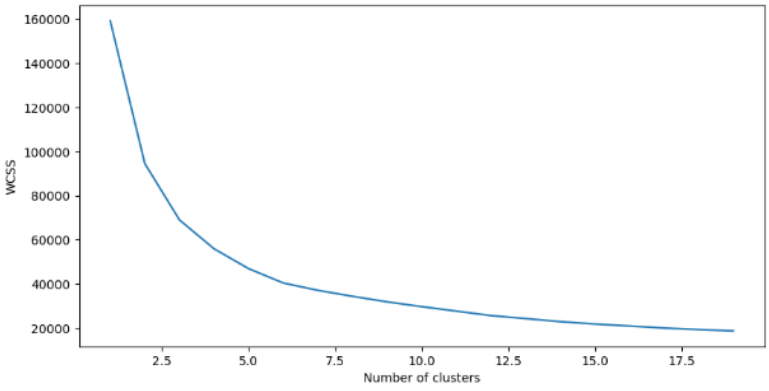


Abbildung 28: Bestimmung der Clusteranzahl durch die Ellenbogenmethode. (WCSS = Within Cluster Sum of Square)

Durch verschiedene Tests mit unterschiedlichen Merkmalskombinationen wird die folgende Merkmalskombination als optimal ermittelt: *Farbwert*, *Eigenentropie*, *Höhe über DGM* und *Anzahl der Gebäudepunkte in der Nachbarschaft*. Wie bereits in Abbildung 27 zu sehen ist, ermöglicht der Farbwert eine effektive Trennung von Gebäuden, Fahrzeugen und Vegetation. Die Hinzunahme der Höhe über dem DGM erleichtert insbesondere die Trennung hoher Vegetation von anderen Bereichen. Die Einbeziehung der Anzahl der Gebäudepunkte führt zu Clustern, die sich entweder in der Nähe von Gebäuden oder weiter entfernt befinden, sodass kritische Punkte zunächst nicht weiter betrachtet werden müssen. Die Berücksichtigung der Eigenentropie verbessert die Ergebnisse des Clusterings weiter. Die Integration weiterer Attribute verschlechtert jedoch das Ergebnis, sodass die Cluster visuell heterogener erscheinen. Eine zusätzliche Optimierung wird erreicht, indem der K-Means-Algorithmus separat für Punkte innerhalb und außerhalb von ALKIS-Straßenobjekte (Siehe Tabelle 2) durchgeführt wird. Dies ermöglicht eine bessere Trennung von Fahrzeugen, Vegetation und Gebäuden. Sowohl für Punkte innerhalb als auch außerhalb der Straße wurde $K = 8$ für K-Means festgelegt, sodass sich insgesamt 16 Cluster ergeben. Diese Cluster werden einzeln gespeichert und in CloudCompare importiert. Hier werden die Cluster der visuell richtigen Klasse zugeordnet, wie beispielsweise in Abbildung 29 dargestellt wird. In der Abbildung sind die jeweiligen Cluster in blau dargestellt. Die hier gezeigten Cluster 0, 1, 5 und 7 können der Vegetationsklasse zugeordnet werden.

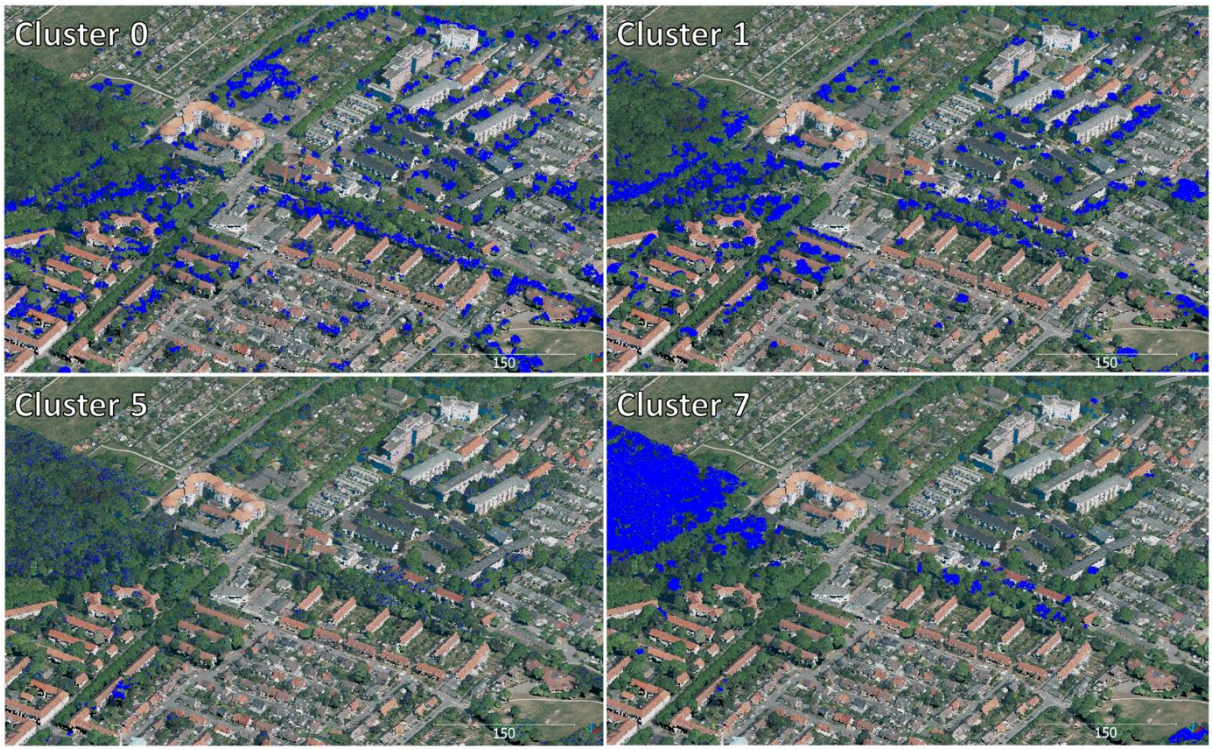


Abbildung 29: Ergebnisse des K-Means-Clusterings. RGB-Punktwolke, sowie Clusterpunkte in blau.

Es ist jedoch nicht immer möglich, alle Cluster bedingungslos einer Klasse zuzuordnen. In einigen Fällen können verschiedene Schwellenwerte festgelegt werden, um Punkte zu identifizieren, die nicht zu einer Vegetationsklasse gehören. Insbesondere die Sättigung und der Helligkeitswert haben sich als

geeignete Merkmale erwiesen, um solche Schwellenwerte festzulegen. Die gewählten Schwellwerte variieren individuell für jedes Cluster und werden durch die visuelle Darstellung bestimmt. Beispielsweise wird für einen Cluster, der sowohl Gebäude- als auch Vegetationspunkte enthält, ein Schwellwert für den Sättigungswert $Saturation > 0{,}07$ festgelegt, um die Gebäudepunkte herauszufiltern. Abbildung 30 veranschaulicht diesen Schwellenwert: Punkte mit einer Sättigung $> 0{,}07$ sind rot dargestellt – die übrigen Punkte blau. Es ist zu beachten, dass mit diesem Schwellenwert nicht alle Punkte, die unter dem Schwellenwert liegen, automatisch der Gebäudeklasse zugeordnet werden können. Für dieses Beispiel kann nur die Aussage getroffen werden, dass alle Punkte, die den Schwellenwert überschreiten, Vegetationspunkte sind und als solche klassifiziert werden können.

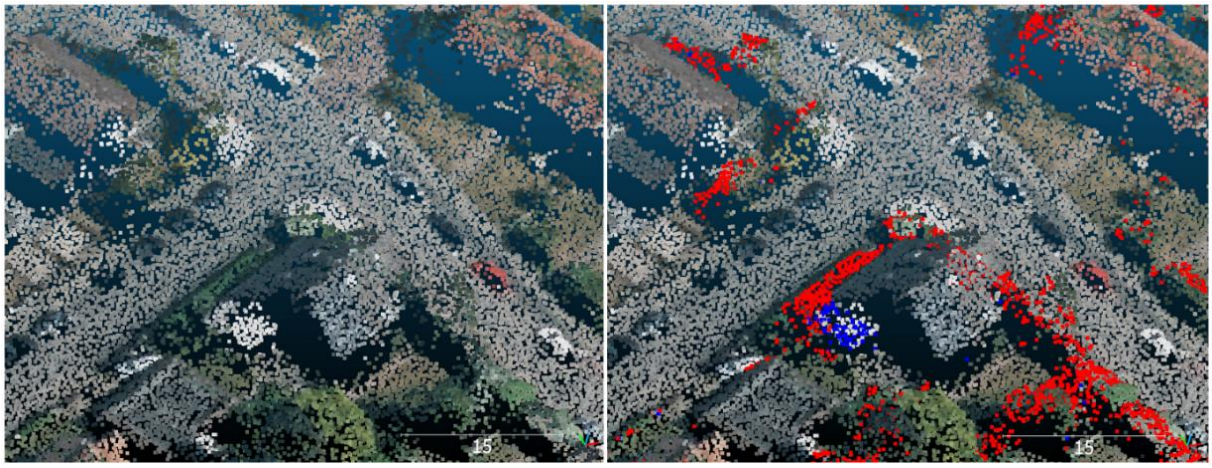


Abbildung 30: Filterung von des Clusters durch Schwellwert. Links: RGB; Rechts: RGB + Clusterpunkte in rot und blau

Durch diese Vorgehensweise werden der Vegetationsklasse weitere Punkte hinzugefügt. Gleichzeitig kann keines der resultierenden Cluster – auch nicht mit Hilfe von Schwellenwerten – einer anderen Klasse als der Vegetation zugeordnet werden. Insbesondere Cluster in unmittelbarer Nähe von Gebäuden (hoher Wert für *count_building*) lassen sich nur schwer in semantische Klassen einteilen. Einige Bereiche werden durch manuelle Segmentierung in CloudCompare der richtigen Klasse zugeordnet, sofern der Aufwand dafür vertretbar ist. Beispielsweise werden Erdhügel auf Baustellen durch manuelle Bearbeitung der Bodenklasse hinzugefügt.

Viele der dargestellten Schritte des Workflows basieren auf visuellen und subjektiven Entscheidungen, wodurch Fehler bei der Zuordnung unvermeidlich sind. Gleichzeitig bleibt das Verhältnis zwischen Aufwand und Nutzen in einem vertretbaren Rahmen. Die verbleibenden Punkte umfassen immer noch sowohl Vegetationspunkte als auch Punkte der Klassen *Gebäude* und *Menschgemacht*. Um die verbleibenden Punkte der richtigen semantischen Klasse zuzuordnen, sind weitere Verarbeitungsschritte notwendig.

4.1.2.6. K-Means 2

Der Großteil der vorhandenen Vegetationspunkte ist bereits durch das erste K-Means-Clustering, die Schwellwerte und die manuelle Annotation annotiert. Mit Hilfe dieser annotierten Vegetationspunkte kann die Anzahl der Vegetationspunkte in der Nachbarschaft (hier als *count_veg* bezeichnet) für weitere Prozessierungen – analog zur Anzahl der Gebäudepunkte (*count_building*) – berechnet werden. Dieses zusätzliche Attribut stellt ein weiteres geeignetes Merkmal für weitere unüberwachte Lernverfahren dar. Aus diesem Grund wird ein weiteres Clustering mit K-Means durchgeführt. Das Vorgehen bei diesem Clustering ist nahezu identisch mit dem ersten K-Means (siehe Abbildung 24). Der einzige Unterschied besteht darin, dass andere Attribute für das Clustering verwendet werden. Neben den Farbinformationen Hue und Value werden in diesem Fall noch die Standardabweichung der Z-Koordinate innerhalb der Nachbarschaft σ_Z sowie die Anzahl der Gebäude und Vegetationspunkte innerhalb der Nachbarschaft verwendet. Die Standardabweichung der Z-Koordinate korreliert mit der berechneten Rauheit und ist insbesondere für Vegetationspunkte relativ hoch. Da die Klassenzugehörigkeit stark von der Nachbarschaft des jeweiligen Punktes abhängt, werden zusätzlich *count_veg* und *count_building* für die Clusterbildung verwendet. Anschließend werden die berechneten Cluster wieder in CloudCompare importiert, visuell der richtigen Klasse zugeordnet und ggf. ein Schwellwertverfahren angewendet.

Abbildung 31 zeigt die Ergebnisse nach den beiden K-Means-Clusterings: In der Mitte sind die verbleibenden unklassifizierten Punkte nach dem ersten K-Means-Prozess in blau dargestellt – rechts die verbleibenden unklassifizierten Punkte nach dem zweiten K-Means-Prozess. Es ist zu erkennen, dass durch den zweiten K-Means-Prozess eine große Anzahl weiterer Vegetationspunkte annotiert werden konnte. Bei den verbleibenden unklassifizierten Punkten handelt es sich hauptsächlich um menschgemachte Objekte und Gebäude, sowie Vegetation in unmittelbarer Nähe von Gebäuden. Aber auch Bodenpunkte, die im Schatten von Gebäuden liegen und daher ein höheres Rauschen aufweisen, sind noch enthalten.

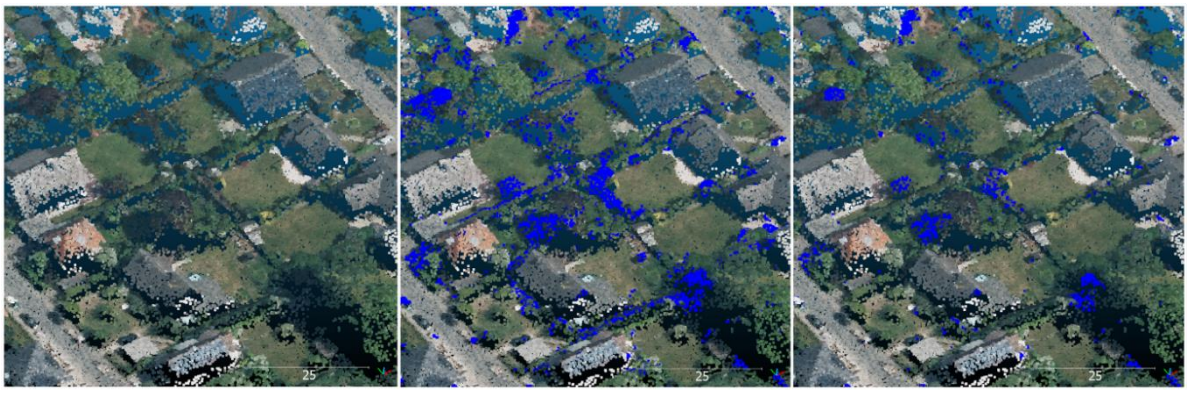


Abbildung 31: Ergebnisse nach K-Means. Unklassifizierte Punkte sind in blau dargestellt. Links: RGB; Mitte: nach K-Means 1; Rechts: nach K-Means 2

4.1.2.7. K-Means, DBSCAN & Random Forest

Nach dem zweiten K-Means-Clustering verbleiben häufig ca. 100.000 unklassifizierte Punkte pro Patch (ca. 2% der Punktwolke). Dabei handelt es sich hauptsächlich um menschgemachte Objekte, Gebäude und Vegetation in der Nähe von Gebäuden. Um diese verbleibenden Punkte endgültig zu klassifizieren, wird eine Kombination aus unüberwachten und überwachten Algorithmen gewählt. Die Grundidee besteht darin, die nicht klassifizieren Punkte nach ihren Farbeigenschaften zu clustern und anschließend jede radiometrische Gruppe räumlich zu clustern. Diese Cluster werden dann mit einen überwachten Algorithmus klassifiziert. Dieser Arbeitsablauf ist in Abbildung 32 visualisiert.

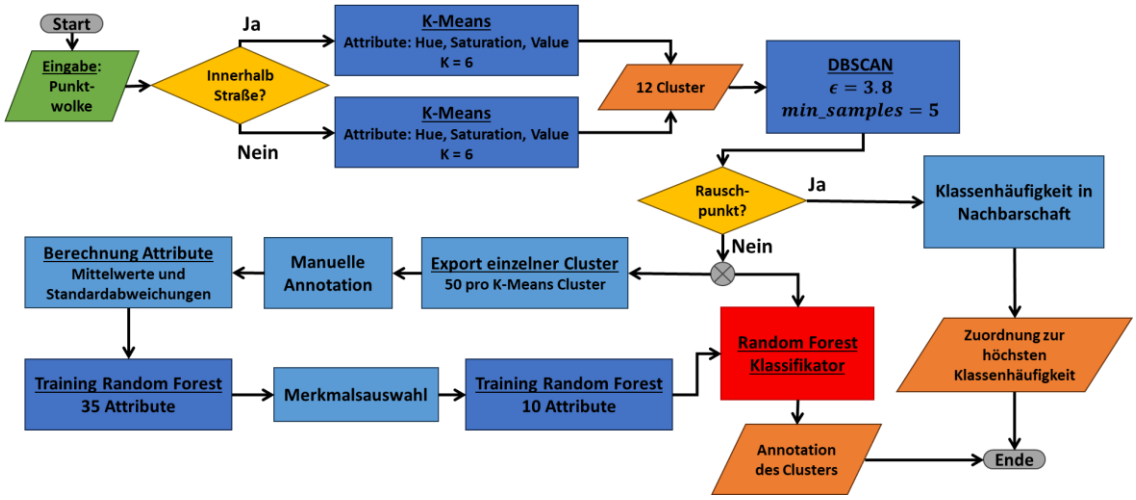


Abbildung 32: Ablaufschema zum radiometrischen und räumlichen Clustering und anschließende RF-Klassifikation

Zunächst wird ein K-Means-Clustering nur mit den Farbinformationen (Hue, Saturation und Value) separat für Punkte innerhalb und außerhalb der Straßenflächen durchgeführt. Dabei werden sechs Klassen verwendet, sodass insgesamt zwölf Cluster entstehen. Diese zwölf Cluster werden dann einzeln einem DBSCAN-Algorithmus übergeben, wobei nur die räumlichen Informationen (XYZ-Koordinaten) für das Clustering verwendet werden. Hierbei wird ein Radius von $\epsilon = 3{,}8\,m$ gewählt und die Mindestanzahl der Punkte auf $min\_samples = 5$ gesetzt. Um Trainingsdaten für einen Random Forest Klassifikator zu erzeugen, werden für jedes radiometrische Cluster einige der räumlichen Cluster exportiert und manuell annotiert. Dieser Prozess wird in einem Jupyter Notebook implementiert und ausgeführt, welches in Anlage A: *Jupyter-Notebooks* zu finden ist.

Einige der exportierten Cluster sind in Abbildung 33 dargestellt, wobei zu erkennen ist, dass in a) und b) die Zuordnung zu einer Klasse sehr eindeutig ist: Vegetation in a) und menschgemacht in b). Allerdings ist die Zuordnung in Abbildung 33 c) und d) selbst für das menschliche Auge schwer zu erkennen: in c) ist nicht zu erkennbar, ob sich ein Auto auf der Straße befindet – in d) liegt das Cluster augenscheinlich auf einem Weg, wobei die Geometrie eher auf Vegetation hindeutet. Insgesamt wurden 838 Cluster manuell annotiert und die zugehörigen Klassen in einer CSV-Datei gespeichert. Im Detail wurden 192 Cluster der Klasse Menschgemacht, 344 der Klasse Vegetation, 110 dem der Klasse Boden, 192 der Klasse Gebäude zugeordnet.

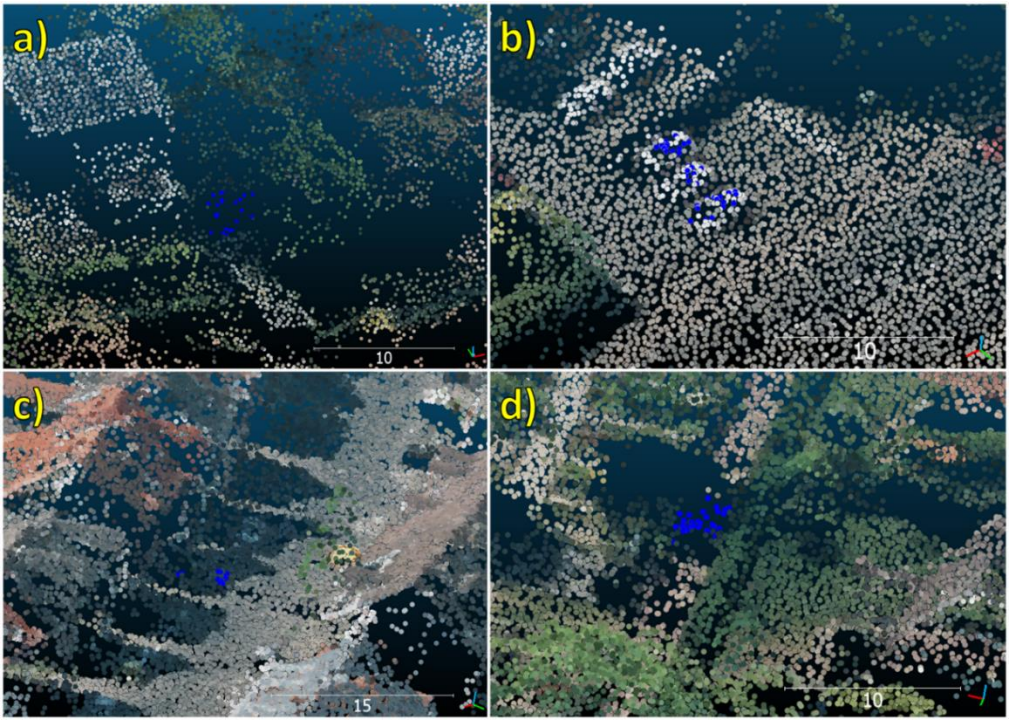


Abbildung 33: Beispiele einiger räumlicher Cluster aus DBSCAN

Nach dem Import der annotierten Cluster in ein weiteres Jupyter-Notebook (*Anlage A: Jupyter-Notebooks*) können diese analysiert und anschließend ein RF-Klassifikator trainiert werden. Für jedes Cluster wird die Anzahl der Punkte ermittelt und sowohl der Mittelwert als auch die Standardabweichung aller vorhandenen Attribute berechnet. Auf diese Weise entstehen für jedes Cluster 35 Attribute, die als Grundlage für das Training des RF-Klassifikators dienen. Zu Beginn wird ein RF-Klassifikator mit allen verfügbaren Attributen trainiert, was zu einer Genauigkeit von 75,5% führt. Anschließend werden die Attribute nach ihrer Relevanz geordnet, um dann dem Random Forest mit einer reduzierten Anzahl der als am relevantesten erachteten Attribute zu trainieren. Die Abhängigkeit der Genauigkeit des Klassifikators von der Anzahl der verwendeten Attribute ist in Abbildung 34 dargestellt.

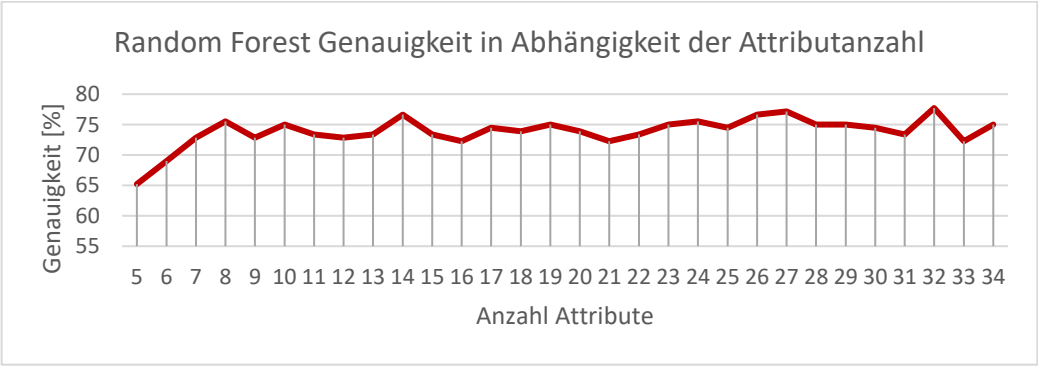


Abbildung 34: Genauigkeit des Random Forest Klassifikator in Abhängigkeit von der Attributanzahl

Die Analyse zeigt, dass bereits ab einer Anzahl von acht Attributen keine signifikante Steigerung der Genauigkeit mehr zu verzeichnen ist. Final wird der RF-Klassifikator mit den zehn relevantesten

Merkmalen trainiert und für die zukünftige Verwendung gespeichert. Mit diesem trainierten Klassifikator können nun alle Cluster klassifiziert werden. In Abbildung 35 ist ein Histogramm der erreichten Klassenwahrscheinlichkeiten dargestellt. Es zeigt, dass nur wenige Cluster eine Wahrscheinlichkeit $< 40\%$ aufweisen. Die verbleibenden Punkte, die DBSCAN als Rauschpunkte markiert, werden der Klasse zugeordnet, die in ihrer Nachbarschaft am häufigsten vorkommt.

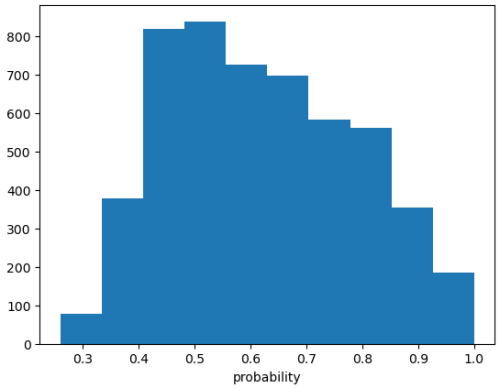


Abbildung 35: Wahrscheinlichkeiten des RF-Klassifikators

4.1.3. Evaluierung der erzeugten Trainingsdaten

Die beschriebenen Schritte zur Optimierung der annotierten Punktwolke zeigen, dass die unüberwachte semantische Segmentierung einer Punktwolke eine komplexe Aufgabe ist, die eine Kombination verschiedener Ansätze erfordert. Durch den Einsatz des CSF kann die Bodenklasse deutlich verbessert werden. Gleichzeitig ist die Wahl der Parameter nicht trivial und abhängig von der vorherrschenden Topographie. Insbesondere Fahrzeuge werden durch den CSF an einigen Stellen aufgrund ihrer geringen Höhe als Boden klassifiziert. Diese Fehlzuordnungen treten in geringem Umfang auf und müssen für die weitere Verarbeitung akzeptiert werden. Bei der Bereinigung der Gebäudeklasse wird mit der Definition von Schwellwerten ein simples Verfahren angewendet, das einen Großteil der Gebäudeklasse verbessert. Gleichzeitig wurden die Schwellwerte lediglich auf Basis eines einzelnen 1×1 m² Patches gewählt. Es gibt Sonderfälle, in denen diese Schwellwerte nicht geeignet sind und eine manuelle Annotation erforderlich ist. Zur Bereinigung der *Nicht-Gelände*-Klasse werden zunächst einige geometrische Merkmale berechnet, was eine geeignete Nachbarschaftsdefinition erfordert. Um einen optimalen Parameter für die Anzahl der Punktnachbarn zu finden, wurden verschiedene Nachbarschaftsgrößen gewählt und die Ergebnisse visuell überprüft.

Um die unklassifizierten Punkte in Bereiche mit ähnlichen Eigenschaften aufzuteilen, wurden verschiedene unüberwachte Verfahren getestet. Die direkte Anwendung von DBSCAN liefert auch bei unterschiedlichen Merkmalskombinationen und verschiedenen Parametern keine guten Ergebnisse. Der Einsatz von K-Means hingegen erweist sich als deutlich vielversprechender – auch wenn die Wahl der richtigen Merkmalskombination nicht trivial ist. Insbesondere die Farbinformation erweist sich als

geeignetes Merkmal, um die Punktwolke in geeignete Cluster einzuteilen. Viele Cluster enthalten jedoch nicht nur eine semantische Klasse, sodass häufig eine manuelle Nachbearbeitung erforderlich ist. Dabei werden einige Klassen durch die Verwendung weiterer Schwellwerte oder durch die manuelle Auswahl einzelner Punkte verbessert. Hier muss der Aufwand gegen den Nutzen abgewogen werden. Da auf diese Weise nicht alle Punkte der richtigen semantischen Klasse zugeordnet werden können, ist eine weitere Bearbeitung notwendig. Der Einsatz eines zweiten K-Means-Clustering ist notwendig, da immer noch zu viele Punkte, die räumlich nahe beieinander liegen, vorhanden sind, um diesen Schritt zu überspringen und DBSCAN mit RF anzuwenden. Wie bereits nach dem ersten K-Means-Clustering ist eine manuelle Nachbearbeitung bzw. das Setzen von Schwellwerten bei einigen Clustern erforderlich. Die nun verbleibenden unklassifizierten Punkte liegen häufig in Schattenbereichen, aber auch in Grenzbereichen zwischen verschiedenen semantischen Klassen – eine Zuordnung ist auch bei manueller Bearbeitung oft schwierig. Um dennoch die Informationen dieser Punkte zu erhalten, werden sie durch eine Kombination aus radiometrischem und räumlichem Clustering mit anschließender RF klassifiziert. Da es sich bei den verbleibenden Punkten um Grenzfälle handelt, die schwer zu klassifizieren sind, kann mit Random Forest keine höhere Genauigkeit als 75% erreicht werden. Für die geringe Anzahl unklassifizierter Punkte ist diese Genauigkeit jedoch ausreichend.

Insgesamt stellt der beschriebene Workflow eine systematische Methode zur Generierung einer gelabelten Punktwolke für das Training eines Deep Learning Modells dar. Obwohl einige manuelle Schritte und eine zeitaufwändige Nachbearbeitung erforderlich sind, konnte eine akzeptable Genauigkeit bei der Klassenzuordnung erreicht werden, um ein optimales Training zu gewährleisten. Tabelle 3 zeigt die absolute und relative Anzahl unklassifizierter Punkte nach jedem Schritt für ein einzelnes Patch.

Tabelle 3: Anzahl unklassifizierter Punkte nach dem jeweiligen Bearbeitungsschritt für ein einzelnes Patch

	Geom. Klassifikation	**CSF**	**K-Means 1**	**K-Means 2**	**K-Means, DBSCAN & RF**
Punktanzahl	2.196.631	1.964.533	193.111	38.188	0
Rel. Anteil	100%	89,40%	8,79%	1,74%	0%

Zudem sind die Zwischenergebnisse jedes Schrittes beispielhaft in Anlage B: *Zwischenergebnisse des Workflows zur Optimierung des Trainingsdatensatzes* visualisiert. Es ist zu erkennen, dass jeder Schritt entscheidend zur Optimierung der geometrischen Klassifikation beiträgt und letztlich alle Punkte erhalten bleiben. Der entstandene Trainingsdatensatz enthält insgesamt 66,8 Millionen Punkte und beinhaltet verschiedenste Topographien. Die absoluten und relativen Anteile an Punkten pro Klasse sind in Tabelle 4 zusammengefasst. Es ist zu erkennen, dass vor allem die Klassen *Menschgemacht* und *Brücken* stark unterrepräsentiert sind.

Tabelle 4: Absoluter und relativer Anteil an Punkten pro Klasse im gesamten Trainingsdatensatz

	Boden	**Gebäude**	**Vegetation**	**Menschgemacht**	**Brücken**
Punktanzahl	31.463.945	12.384.375	22.498.089	330.857	101.342
Rel. Anteil	47.12%	18.55%	33.69%	0.50%	0.15%

Wie Abbildung 36 zeigt, werden sowohl ländliche Bereiche mit Wäldern und Wiesen als auch urbane Bereiche mit den verschiedensten Baustilen abgebildet. Zudem sind auch Besonderheiten, wie Baustellen mit Kränen, ein Schienen- und Straßenbahnnetz und Autobahnen enthalten. Der generierte Datensatz enthält diverse heterogene Objekte und sollte eine geeignete Grundlage für das Training darstellen. Die nach diesem Workflow erstellten annotierten Punktwolken jedes einzelnen Patches können unter Anlage C: *Annotierter Trainingsdatensatz* heruntergeladen werden.

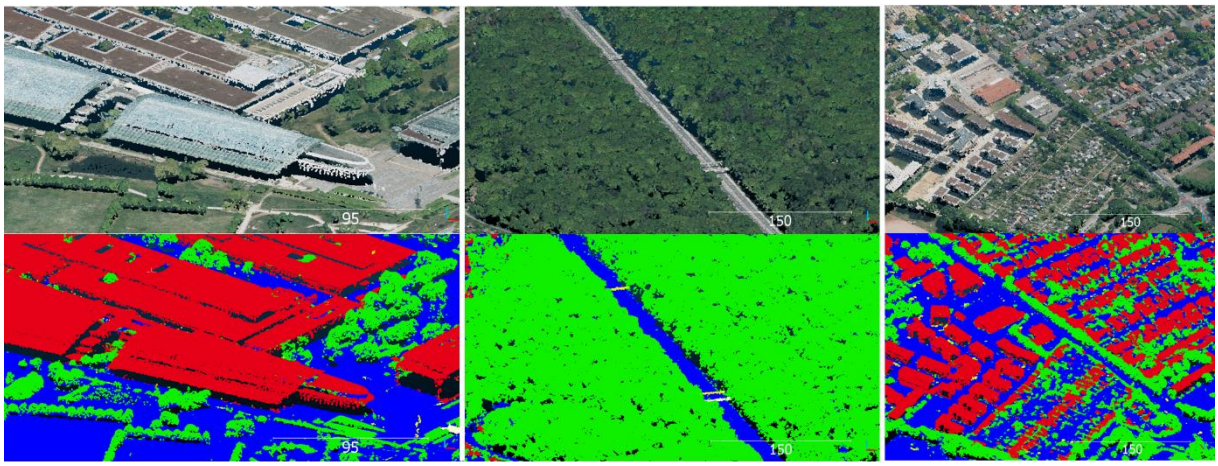


Abbildung 36: Beispielhafte Ausschnitte des Trainingsdatensatzes. Oben RGB, unten Klassifikation nach Bearbeitung.

4.2. Aufteilung der Trainingsdaten & Batching der Punktwolke

Für das Training des Deep-Learning-Modells werden die Trainingsdaten in *Training* und *Validierung* aufgeteilt. Für die weiteren Untersuchungen wird das Patch *555000_5800000* für die Validierung verwendet, sodass 11 km^2 der 12 km^2 für das Training zur Verfügung stehen. Bei der Auswahl der Validierungsdaten ist darauf zu achten, dass sie sich nicht mit den Trainingsdaten überlappen und dass alle Objektklassen enthalten sind. Außerdem sollten die Validierungsdaten eine ähnliche Topographie wie die Trainingsdaten aufweisen – Sonderfälle, wie z. B. besondere Gebäudeformen, sollten nicht in den Validierungsdaten enthalten sein. Abbildung 37 visualisiert den ausgewählten Validierungsdatensatz und zeigt, dass alle verfügbaren Klassen in der Punktwolke vorhanden sind und verschiedene Gebäudetypen enthalten sind. Zusätzliche enthält das Validierungsgebiet eine Autobahn inklusive zweier Brücken. Objekte der Klasse Menschgemacht sind ebenfalls enthalten, auch wenn sie aufgrund ihrer geringen räumlichen Ausdehnung in der Abbildung nicht ersichtlich sind.

Abbildung 37: Ausgewählter Validierungsdatensatz. Links: Klassifikation der Ground Truth; Rechts: Farbliche Darstellung

Die Verarbeitung eines gesamten Punktwolken-Patches in einer Trainingsepoche ist allein aufgrund des Speicherdarfs nicht möglich. Aus diesem Grund und zur Erhöhung der Redundanz ist ein Batching der Punktwolken notwendig. WINIWARTER (2018) schlägt vor, kreisförmig überlappende Batches zu wählen, indem für bestimmte Gitterpunkte eine feste Anzahl nächster Punkte gewählt wird. Durch die resultierende Kreisform kann die Isotropie gewährleistet werden. Der Gitterpunktabstand $\Delta_{x,y}$ wird nach Formel 24 berechnet, und basiert auf der Punktdichte ρ und der Anzahl der Punktnachbarn kNN. Mit dem Faktor $0{,}95$ wird ein Rand von 5% zur Kreisnachbarschaft addiert, um lokal variierende Punktdichte zu berücksichtigen.

Formel 24: Gitterabstand für das Batching der Punktwolke (WINIWARTER 2018:29)

$$\Delta_{x,y} = \sqrt{\frac{kNN}{\pi\rho}} \cdot \frac{\sqrt{2}}{2} \cdot 0.95$$

Für jeden Gitterpunkt wird die definierte Anzahl von Punktnachbarn durch einen Kd-Tree ausgewählt. Die Auswahl der Gitterpunkte stellt sicher, dass jeder Punkt in mindestens drei Batches repräsentiert ist. Zur weiteren Verwendung im Trainings- oder Klassifikationsprozess werden die einzelnen Batches z. B. als LAS-Datei gespeichert. Durch die Überlappung der Batches ist es möglich, dass derselbe Punkt in verschiedenen Batches unterschiedlichen Klassen zugeordnet wird. Um diese Herausforderung zu begegnen, muss vor dem Batching eine individuelle Punkt-ID vergeben werden. Nur so ist es möglich, nach der Klassifizierung der Batches identische Punkte zuzuordnen. Zusätzlich wird die Zuverlässigkeit der Ergebnisse durch eine Mittelbildung der Klassenwahrscheinlichkeiten erhöht.

Grundsätzlich wird in dieser Arbeit eine Punktanzahl von $K = 100.000$ Punkten pro Batch gewählt. Auf diese Weise werden nach dem beschriebenen Schema für das Training 4.287 Batches aus den 11 Patches und für die Validierung 400 Batches aus dem Patch 554000_5800000 erzeugt. Das Batching wird dabei für jedes Punktwolken-Patch einzeln durchgeführt, sodass an den Randbereichen keine Überlappung zum benachbarten Punktwolken-Patch entsteht. Die vollständigen Python-Skripte zur Generierung der Batches ist in Anlage D: *Python Quellcode* zu finden.

4.3. Anwendung des Deep Learning Modells PointNet++

Die Anwendung des PointNet++ Modells erfolgt nach den in Kapitel 2.3.8.2 erläuterten Grundlagen und wird mit dem Deep Learning Framework *PyTorch* umgesetzt. Basierend auf der Implementierung von YAN, X. (2019) werden zunächst die Set-Abstraction-Layer (SA-Layer) und die Feature-Propagation-Layer (FP-Layer) entwickelt. Im SA-Layer werden zunächst das Farthest-Point-Sampling (FPS) und die Ball-Abfrage durchgeführt. Das FPS benötigt zur Initialisierung die Anzahl der zu selektierenden Punkte, während die Ballabfrage den Radius und die maximale Anzahl der Nachbarn benötigt. Um die Geschwindigkeit dieser Operationen zu erhöhen, ist eine Implementierung in CUDA notwendig, damit die Rechenschritte auf der Grafikkarte durchgeführt werden können. In dieser Arbeit wird die Implementierung dieser Operationen von WIJMANS (2018) übernommen. Anschließend werden die Punkte an die MLPs mit geteilten Gewichten übergeben. Diese werden als 2D-Faltung mit einer Kernelgröße von eins implementiert. Es folgt eine *Batch-Normalisierung* und die ReLU-Aktivierungsfunktion. Im FP-Layer werden die Merkmale des letzten SA-Layers für die Punkte des aktuellen Layers auf Grundlage der drei nächstgelegenen Punkte interpoliert. Die Implementierung dieser Interpolation wurde ebenfalls von WIJMANS (2018) übernommen, sodass diese ebenfalls auf der GPU prozessiert wird. Anschließend wird wieder eine vorgegebene Anzahl von MLPs durchgeführt, die als 2D-Faltung mit einer vordefinierten Anzahl von Filtern implementiert werden. Für das PointNet++ Modell muss zunächst die Anzahl der SA- bzw. FP-Layer mit ihren jeweiligen Parametern festgelegt werden. Dabei werden die Parameter in Anlehnung an WINIWARTER (2018) gewählt, wo drei SA- und FP-Layer verwendet werden. Anschließend wird ein weiteres MLP mit 64 Filtern durchgeführt, gefolgt von einer Batch-Normalisierung und einem Dropout-Layer mit 50% Dropout. Im finalen Output MLP werden schließlich die Klassenwahrscheinlichkeiten zurückgegeben. Eine erste Implementierung zur weiteren Erläuterung der verwendeten Funktionen ist als Jupyter-Notebook in Anlage A: *Jupyter-Notebooks* hinterlegt. Die gesamte Implementierung inklusive der gewählten Parameter ist im Prozessablaufplan in Abbildung 38 visualisiert. Der vollständige Quellcode für die Anwendung und das Training des PointNet++ Modells ist in Anlage D: *Python Quellcode* zu finden.

In der Grundkonfiguration wird ein Batching mit $k = 100.000$ Punkten durchgeführt. Als Optimierer wird *Adam* mit einer Lernrate von $0{,}001$ verwendet, während als Verlustfunktionen der *Cross-Entropy-Loss* und der *Focal-Loss* mit verschiedenen Parametern getestet werden. Im Folgenden werden auch die Verwendung eines *Learning-Rate-Schedulers* und die Verwendung des *AdamW*-Optimierers untersucht. Im Lernprozess wird das Netz innerhalb einer Epoche auf alle 4.287 Trainingsbatches trainiert, um anschließend das trainierte Modell anhand der 400 Validierungsdaten zu validieren. Die vollständige Implementierung des Netzes einschließlich aller Funktionen ist abrufbar in Anlage D: *Python Quellcode*.

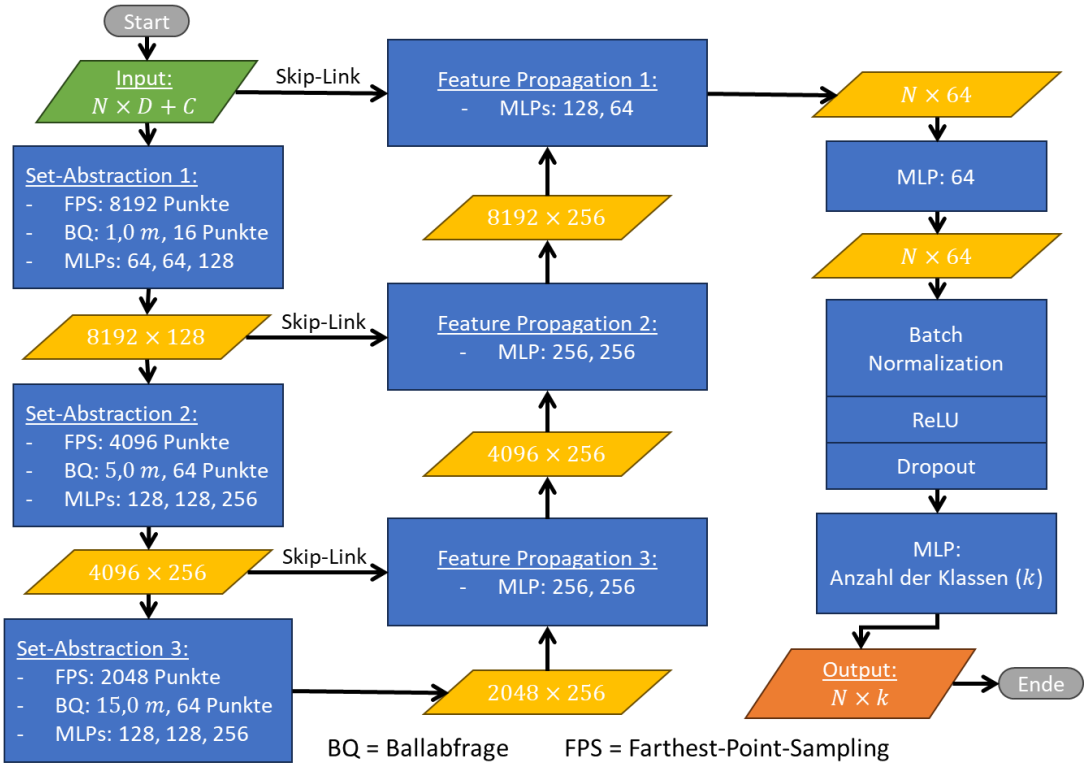


Abbildung 38: Prozessablaufplan des implementierten PointNet++ Modells inkl. der gewählten Parameter

4.4. Dataset Augmentation Methoden

Um die Generalisierbarkeit des Modells zu verbessern, werden verschiedene Methoden der Dataset-Augmentation implementiert, die bereits in Kapitel 2.2.5 erläutert werden und beispielsweise von HAHNER et al. (2020) beschrieben sind. Die mit diesen Methoden möglicherweise zu erzielenden Leistungsänderungen sollen nachfolgend in 5.3 untersucht werden. In Anlage E: *Beispielhafte Anwendung der Dataset Augmentation* sind die Auswirkungen der einzelnen Methoden auf ein einzelnes Batch beispielhaft dargestellt. Der Quellcode der einzelnen Funktionen ist in Anlage D: *Python Quellcode* zu finden. Im Folgenden werden die implementierten Methoden kurz erläutert:

Zufälliges Rauschen für die Z-Koordinate: Photogrammetrische Punktwolken aus Luftbildern haben die Eigenschaft, dass insbesondere die Z-Koordinate eine größere Ungenauigkeit aufweist als die X- und Y-Koordinate. Ziel des *Z-Jitterings* ist es, diesen Effekt zu verstärken, um das Modell unempfindlicher gegen Rauschen und generalisierbarer zu machen. Dabei wird jeder Z-Koordinate ein zufälliges, normalverteiltes Rauschen hinzugefügt. Der dazu notwendige Parameter ist die Standardabweichung der Normalverteilung.

Zufälliger Verlust der Farbinformation: Der zufällige Verlust der Farbinformation soll die Farbabhängigkeit des Modells reduzieren. Dazu wird die Farbinformation eines bestimmten Prozentsatzes der Punkte entfernt ($H, S, V = 0, 0, 0$). Die Auswahl der Punkte, die ihre Farbinformation verlieren, ist rein zufällig und ändert sich in jeder Epoche.

Zufälliges Rauschen der Farbinformation: Analog zum Rauschen auf der Z-Koordinate wird hier das Rauschen auf die Farbinformationen angewendet. Dabei ist die Wahl der Standardabweichungen für die einzelnen Farbkomponenten von besonderer Bedeutung, da diese in einem plausiblen Bereich liegen müssen. Auch mit dieser Methode soll die Abhängigkeit von der Farbinformation reduziert werden.

Verschiebung der Farbinformation des Batches: Hierbei werden die Farbattribute des gesamten Punktwolken-Batches um gleiche Beträge verschoben. Die Werte für die einzelnen Farbkomponenten basieren auf der Normalverteilung. Diese Methode führt dazu, dass das Punktwolken-Batch in jeder Epoche unterschiedliche Farben aufweist, die in einem plausiblen Bereich liegen. Die Generalisierbarkeit des Modells kann dadurch erhöht werden.

Zufällige Drehung um die Z-Achse: Bei dieser Methode wird das gesamte Punktwolken-Batch im Mittelpunkt um die Z-Achse gedreht. Die Bestimmung des Drehwinkels erfolgt rein zufällig und ist für jede Epochen verschieden. Basierend auf dieser Methode wird dem Modell in jeder Epoche ein veränderter Datensatz präsentiert. Vor allem für unterrepräsentierte Daten kann so die Generalisierbarkeit erhöht werden.

4.5. Software, Hardware & Programmierumgebungen

Für die Generierung der Trainingsdaten, sowie für die Implementierung und das Training des PointNet++ Modells wird verschiedene Hard- und Software eingesetzt. Alle Prozesse basieren auf der Programmiersprache *Python*, wobei Python in der Version 3.11 eingesetzt wird. Für die Entwicklung erster Prototypen werden hauptsächlich Jupyter-Notebooks verwendet, welche mit Hilfe der Python-Distribution *Anaconda Navigator* in der Version 2.4.2 (© *Anaconda Inc.*) programmiert werden. Die Entwicklung vollständiger Skripte erfolgt in der Entwicklungsumgebung *PyCharm* (© *Jet Brains*) in der quelloffenen Community Version 2023.1.1. Für die Entwicklung des PointNet++ Modells wird insbesondere das von © *Meta AI* entwickelte und zur *Linux Foundation* gehörende Machine Learning Framework *PyTorch* verwendet. PyTorch wird in der Version 2.01 in Verbindung mit *CUDA-Toolkit 11.8* (© NVIDIA) eingesetzt, um Berechnungen auf der GPU zu ermöglichen. Punktwolken werden mit der Open Source Software *CloudCompare* in der Version 2.12.4 (*Kyiv*) visualisiert und verarbeitet.

Hardwaretechnisch wird für die Entwicklung auf einem *Lenovo® ThinkPad P1 Gen 2* mit einem *Intel® i7* Prozessor mit 12 Kernen und 32 GB Arbeitsspeicher gearbeitet. Dieser Rechner verfügt über eine *NVIDIA Corporation TU117GLM* Grafikkarte mit 4 GB Speicher und verwendet die Linux-Distribution *Ubuntu 22.04.2 LTS* als Betriebssystem. Für das Modelltraining steht ein Rechner mit dem Betriebssystem *Windows 10 Enterprise* zur Verfügung, der mit einer *AMD Ryzen Threadripper 3960X* CPU mit 256GB Arbeitsspeicher und 24 Kernen sowie einer *NVIDIA GeForce RTX 3090* mit 24 GB Arbeitsspeicher ausgestattet ist.

5. Experimente

Nach der Implementierung des PointNet++ Modells kann das Netz mit dem annotierten Datensatz trainiert werden. Die Wahl der Hyperparameter ist jedoch nicht trivial und sollte in verschiedenen Konfigurationen getestet werden. Dazu gehört die Wahl der Verlustfunktion und des Optimierers, aber auch die Entscheidung welche Attribute für das Training sinnvoll sind. Weiterhin ist zu untersuchen, ob die implementierten Methoden der Dataset-Augmentation und die Größe der Batches einen Einfluss auf die Performance des Modells haben. Sofern nicht anders beschrieben, wird in den folgenden Tests der Adam-Optimierer mit einer Lernrate von 0,001 verwendet und der Focal Loss mit $\gamma = 4$ als Verlustfunktion gesetzt. Darüber hinaus besteht der Trainingsdatensatz aus allen verfügbaren Daten, mit Ausnahme des Patches *555000_5800000*, der für die Validierung der Modelle verwendet wird.

5.1. Variation der Verlustfunktion

Typischerweise wird für die semantische Klassifikation der *Cross-Entropy-Loss* (CEL) als Verlustfunktion verwendet. Wie bereits in Kapitel 2.2.3 beschrieben, wurde für stark unbalancierte Trainingsdaten von LIN, T.-Y. et al. (2017) der *Focal Loss* (FL) vorgestellt, der über einen Parameter γ den relativen Verlust korrekt klassifizierter Ergebnisse reduziert. Um die Auswirkungen der Verlustfunktion zu untersuchen, werden verschiedene Tests durchgeführt. Da der Focal-Loss nicht standardmäßig in PyTorch implementiert ist, wird die Entwicklung von CARWIN (2017) verwendet.

Um zunächst Rechenkapazität zu sparen, wird das Training auf ein einzelnes Patch (554000_5798000) beschränkt. Die Validierung erfolgt bereits mithilfe des gesamten Validierungsdatensatz. Bereits hier zeigt sich, dass mit dem Focal Loss (mit $\gamma = 2$) für stark unterrepräsentierte Klassen bessere Genauigkeiten erreicht werden als mit dem CEL, wie die Konfusionsmatrizen in Abbildung 39 zeigen.

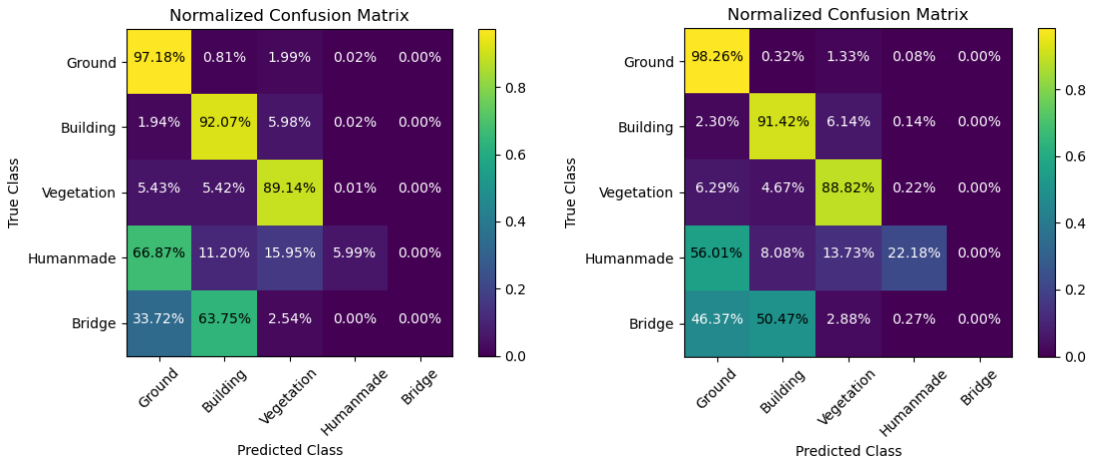


Abbildung 39: Konfusionsmatrix für Validierungsdaten. Links: Cross-Entropy-Loss; Rechts: Focal Loss

Hier ist zu erkennen, dass die Genauigkeit für die Klasse *Menschgemacht* deutlich höher ist als bei der Verwendung des CEL. Die Gesamtgenauigkeit unterscheidet sich kaum zwischen dem Cross-Entropy-

Loss (92,8%) und Focal-Loss (93,0%), da die Klasse Menschgemacht nur einen geringen Teil des gesamten Validierungsdatensatzes ausmacht. Dies wird auch durch die ROC-Kurven in Abbildung 40 bestätigt. Es ist zu erkennen, dass die ROC-Kurve für *Menschgemacht* unter Verwendung des CEL näher an der Diagonalen verläuft als bei der Verwendung des FL. Dies deutet darauf hin, dass die Verwendung des FL für diese Klasse leistungsfähiger ist.

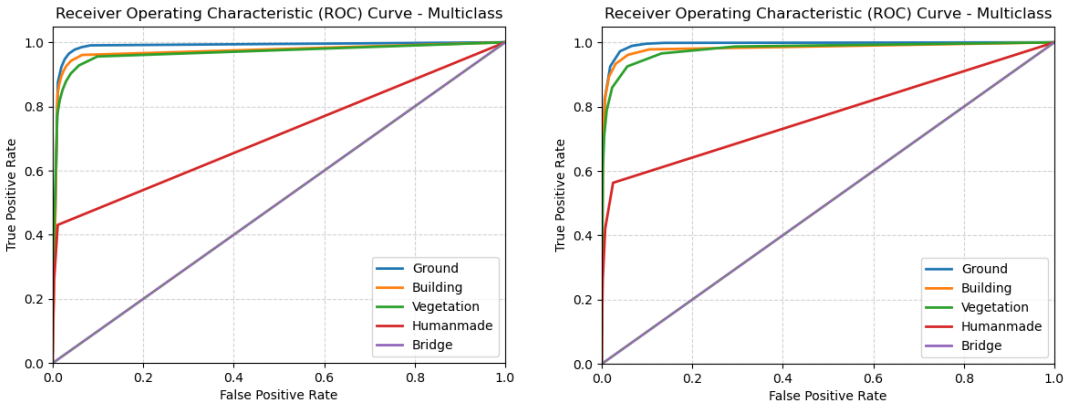


Abbildung 40: ROC-Kurven für Validierungsdaten mit Cross-Entropy-Loss (links) und Focal Loss mit $\gamma = 2$ (rechts)

Gleichzeitig verläuft die ROC-Kurve für die Brückenklasse entlang der Diagonalen, was auf einen Zufallsprozess hindeutet. Zur Optimierung des Modells wird zunächst die Trainingsmenge auf den gesamten Trainingsdatensatz erhöht. Die berechneten Modellparameter des vorherigen Trainings, mit nur einem Patch, werden als Ausgangspunkt für das weitere Training verwendet. Für $\gamma = 2$ zeigt sich, dass bereits nach wenigen Epochen eine Validierungsgenauigkeit von 93,5% erreicht wird. Wie die Konfusionsmatrix in Abbildung 41 zeigt, können alle Klassengenauigkeiten verbessert und erstmals auch Brücken korrekt klassifiziert werden.

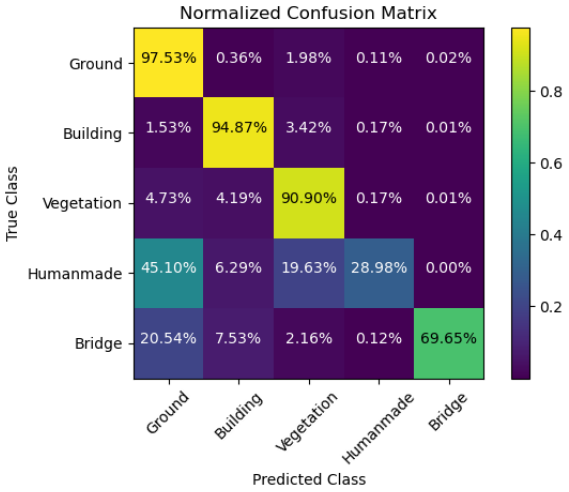


Abbildung 41: Konfusionsmatrix für FL $\gamma = 2$ mit gesamten Trainingsdaten

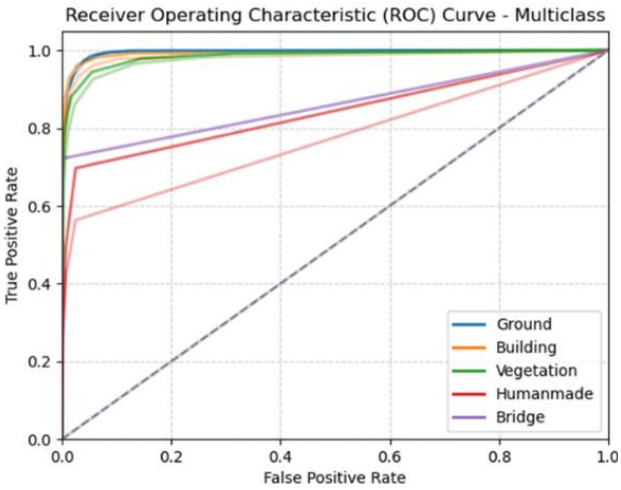


Abbildung 42: ROC-Kurven für FL ($\gamma = 2$) mit gesamten Trainingsdaten

Die Genauigkeiten der einzelnen Klassen sind in Tabelle 5 zusammengefasst. Die ROC-Kurven in Abbildung 42 zeigen, dass durch die größere Menge an Trainingsdaten nun eine deutlich bessere

Performance erzielt werden kann; die ROC-Kurven aus dem Modell mit geringerer Trainingsdatenmenge (Abbildung 40 rechts) sind hier mit geringerer Sättigung dargestellt.

Bei der Wahl von $\gamma = 3$ wird eine Validierungsgenauigkeit von 95,1% erreicht. Die dabei erzielten Klassengenauigkeiten sind in Tabelle 5 zusammengefasst. Sowohl die Konfusionsmatrix als auch die ROC-Kurven (siehe Anlage F-1) zeigen, dass die Klasse *Menschgemacht* mit dem Parameter $\gamma = 3$ deutlich besser klassifiziert werden kann als mit dem Parameter $\gamma = 2$. Gleichzeitig nimmt die Leistung für die Brückenklasse sowohl in der ROC-Kurve als auch in der Genauigkeit ab. Bei der Verwendung von $\gamma = 4$ und von $\gamma = 5$ kann die Gesamtgenauigkeit auf 95,5% erhöht werden. Es gibt jedoch Unterschiede in der Klassengenauigkeit, wie Tabelle 5 zeigt. Vor allem für die unterrepräsentierten Klassen ergeben sich durch die Verwendung von $\gamma = 4$ bessere Genauigkeiten. Die ROC-Kurven in Anlage F-3 und F-4 zeigen nur minimale Unterschiede.

In Tabelle 5 sind die Gesamtgenauigkeiten sowie die Klassengenauigkeiten bei Verwendung des jeweiligen Parameters angegeben. Es wird deutlich, dass insgesamt die Verwendung von $\gamma = 4$ die besten Ergebnisse liefert. Insbesondere die unterrepräsentierten Klassen Menschgemacht und Brücken weisen hier eine hohe Genauigkeit auf. Die Konfusionsmatrix und die ROC-Kurven, die mit dem Focal Loss bei Verwendung von $\gamma = 4$ erzielt werden, sind in Abbildung 43 und Abbildung 44 dargestellt.

Tabelle 5: Genauigkeiten bei Variation des γ-Parameter in der Verlustfunktion. Maximale Werte sind Rot hinterlegt.

	Gesamt	**Boden**	**Gebäude**	**Vegetation**	**Menschgemacht**	**Brücken**
$\boldsymbol{\gamma = 2}$	93,50%	97,53%	94,87%	90,90%	28,98%	**69,65%**
$\boldsymbol{\gamma = 3}$	95,10%	97,27%	94,51%	91,82%	32,16%	47,93%
$\boldsymbol{\gamma = 4}$	**95,50%**	97,53%	**95,35%**	**92,23%**	**36,34%**	69,45%
$\boldsymbol{\gamma = 5}$	**95,50%**	**97,71%**	95,24%	91,81%	25,60%	67,22%

Aufgrund der hier dargestellten Beobachtungen wird der Focal Loss mit $\gamma = 4$ als optimal für das Training angesehen und in den folgenden Untersuchungen als Standardwert verwendet. Die Performance des Modells mit $\gamma = 4$ mit einer Gesamtgenauigkeit von 95,5% dient nun als Referenz für die nachfolgenden Untersuchungen.

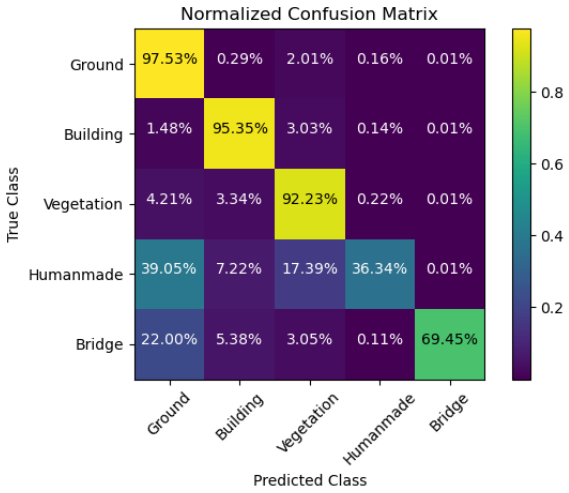


Abbildung 43 Konfusionsmatrix FL mit $\gamma = 4$

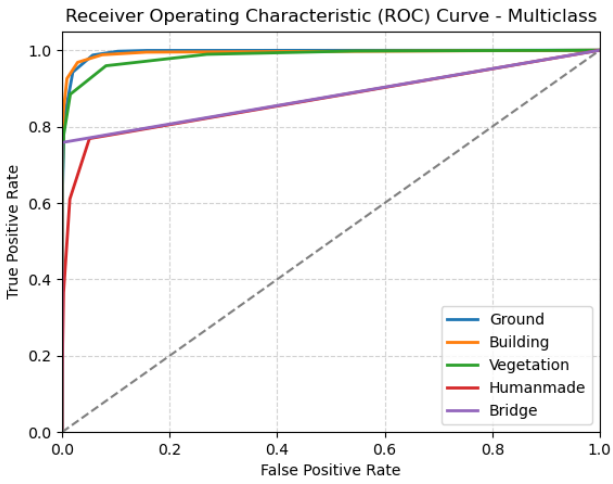


Abbildung 44: ROC-Kurve mit $\gamma = 5$

5.2. Einfluss der verwendeten Attribute

Für die verwendeten photogrammetrischen Punktwolken stehen neben den XYZ-Koordinaten auch Farbinformationen und Zusatzinformationen wie Punktpräzision und Anzahl der Stereomodelle zur Verfügung. Das hier implementierte Modell ist in der Lage mit verschiedenen Merkmalsteilmengen zu lernen, sodass der Einfluss der einzelnen Attribute bestimmt werden kann. Im Folgenden wird das Modell zunächst nur mit den geometrischen Informationen und anschließend mit den geometrischen und radiometrischen Informationen trainiert. Die Ergebnisse der einzelnen Epochen im Trainingsprozess sind in Anlage G: *Training mit verschiedenen Attributteilmengen* wiedergegeben.

5.2.1. Modell-Training mit geometrischen Informationen

Um den Einfluss der Farbinformation und der Zusatzinformation zu untersuchen, wird das Modell zunächst nur auf Grundlage der 3D-Koordinaten trainiert. Trotz des Informationsverlustes wird eine Validierungsgenauigkeit von $94{,}2\%$ erreicht, was einem Genauigkeitsverlust von $-1{,}3\%$ gegenüber der Verwendung aller Attribute entspricht. Dies deutet darauf hin, dass auch bei Verwendung aller verfügbaren Attribute eine starke Abhängigkeit von der Geometrie besteht.

Abbildung 45 zeigt die aus der Validierung resultierende Konfusionsmatrix: Im Vergleich zum Training mit allen verfügbaren Attributen (Abbildung 43) verringern sich vor allem die Genauigkeiten der Klassen Gebäude, Menschgemacht und Brücke um -10%, -20% und -20%. Auch die ROC-Kurven in Abbildung 46 zeigen einen Leistungsabfall insbesondere für Vegetation und Gebäude. Die ROC-Kurven bei der Verwendung aller verfügbaren Attribute sind hier mit einer geringeren Sättigung dargestellt.

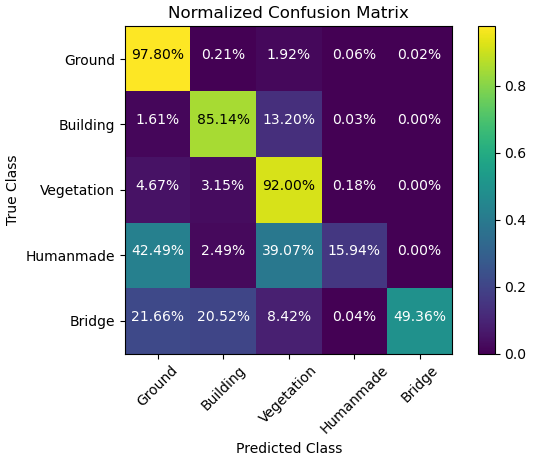


Abbildung 45: Konfusionsmatrix – Training mit Geometrie

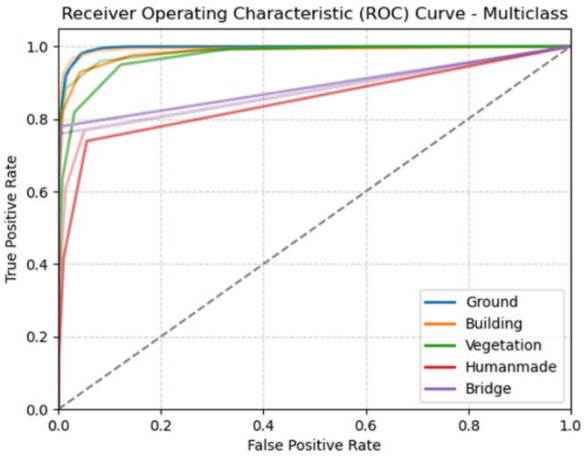


Abbildung 46: ROC-Kurven – Training mit der Geometrie

5.2.2. *Training des Modells mit geometrischen und radiometrischen Informationen*

Werden sowohl geometrische als auch radiometrische Informationen im Trainingsprozess verwendet, so erhält das Modell sechs verschiedene Attribute zur Vorhersage der Klassenzugehörigkeit. Die Informationen über die Punktgenauigkeit und die Anzahl der Stereomodelle entfallen in diesem Prozess. Trotz des Verzichts auf die zusätzlichen Attribute wird eine Validierungsgenauigkeit von 95,6% erreicht. Sie ist damit um +0,1% höher als bei der Verwendung aller Attribute. Wie aus der Konfusionsmatrix (Abbildung 47) hervorgeht, erhöht die Verwendung der sechs Attribute die Genauigkeit für die Klassen *Boden* (+0,5%) und *Vegetation* (+0,6%). Gleichzeitig verringern sich die Genauigkeiten für die unterrepräsentierten Klassen *Gebäude* (−0,2%), *Menschgemacht* (−5,3%) und *Brücke* (−4,7%).

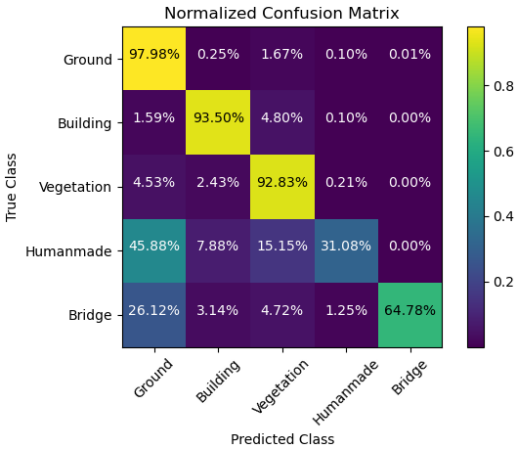


Abbildung 47: Konfusionsmatrix - Training mit Geometrie und Radiometrie

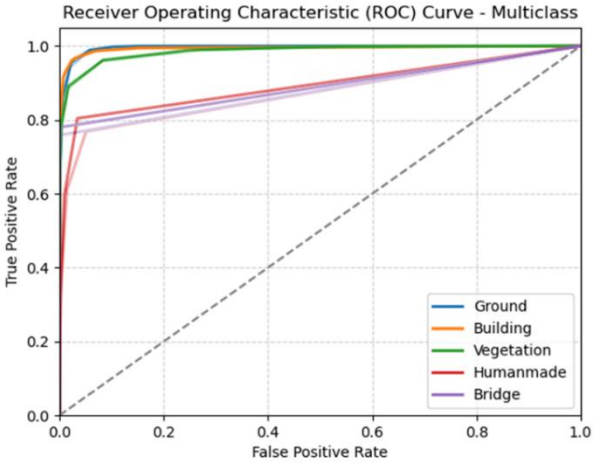


Abbildung 48: ROC-Kurven - Training mit der Geometrie und Radiometrie

Im Gegensatz dazu zeigen die ROC-Kurven (Abbildung 48) eine Verbesserung der Trennschärfe dieser beiden Klassen. Dies könnte auf ein Ungleichgewicht zwischen der True-Positiv-Rate und True-

Negativ-Rate hinweisen. Eine große Fläche unter der Kurve bedeutet oft, dass der Klassifikator eine geringere Falsch-Positiv-Rate aufweist, was zunächst positiv zu bewerten ist. Dies kann jedoch auf Kosten einer höheren Falsch-Negativ-Rate gehen, was zu einer geringeren Genauigkeit führen kann. Die zusätzlichen Informationen der Punktpräzision und der Anzahl der Stereomodelle scheinen keinen positiven Einfluss auf die Modellleistung zu haben. Lediglich die unterrepräsentierten Klassen *Menschgemacht* und *Brücken* weisen eine höhere Validierungsgenauigkeit auf. Offensichtlich korrelieren die Attribute nicht ausreichend mit den Eigenschaften der Klassen. Eine mögliche Erklärung kann aus den Überlappungsbereichen der Befliegung abgeleitet werden. Wie in Abbildung 49 zu erkennen ist, weisen die Überlappungsbereiche eine deutlich höhere Präzision und eine höhere Anzahl an Bildmessungen auf. Die Attribute der Objekte im Überlappungsbereich können daher nicht mit den Attributen der Objekte außerhalb des Überlappungsbereiches verglichen werden.

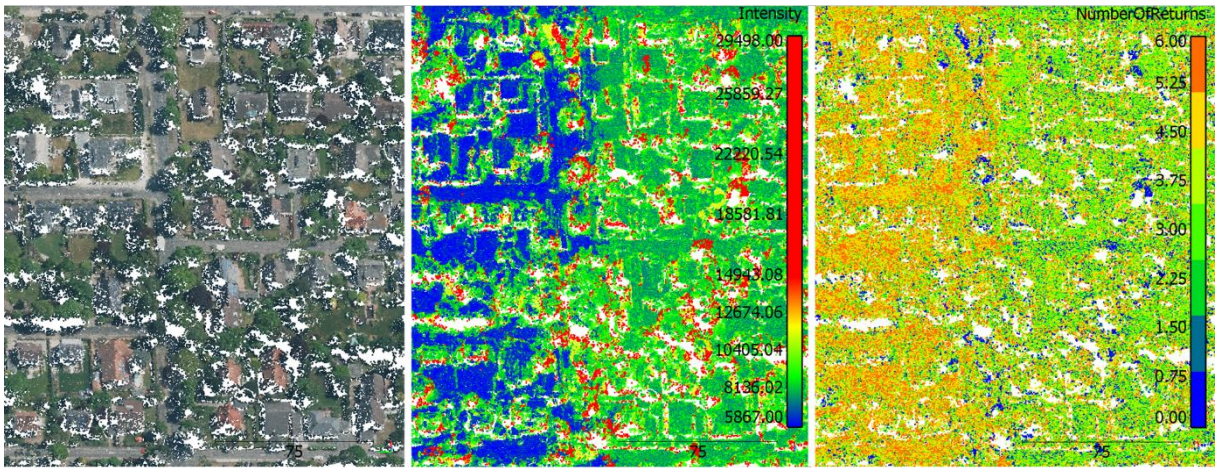


Abbildung 49: Auswirkungen der Überlappungsbereiche. links: RGB; Mitte: Punktpräzision; rechts: Anzahl der Stereomodelle

5.3. Untersuchung der Dataset Augmentation

Um den Einfluss der implementierten Dataset Augmentation (DA)-Methoden auf die Validierungsgenauigkeit zu untersuchen, werden diese einzeln im Training eingesetzt. Die Verwendung des Focal Loss mit $\gamma = 4$, sowie des Adam Optimierers mit einer Lernrate von 0.001 aus Kapitel 5.1 erzielte bisher die beste Validierungsgenauigkeit mit $95{,}5\%$. Dieses Modell wird als Ausgangspunkt für das nachfolgende Modelltraining verwendet, um zu überprüfen, ob eine Verbesserung zu beobachten ist. Für das Training mit DA werden die gleichen Hyperparameter verwendet. Die Auswirkungen der einzelnen Methoden sind dargestellt in Anlage E: *Beispielhafte Anwendung der Dataset Augmentation*.

5.3.1. Zufällige Rotation um die Z-Achse

Bei der zufälligen Rotation um die Z-Achse wird jede Punktwolke in jeder Epoche um einen zufälligen Winkel um die Z-Achse gedreht. Die Metriken zur Validierung der Modellzuverlässigkeit nach jeder

Epoche sind in Tabelle 6 dargestellt. In der ersten Zeile sind die Referenzdaten des Trainings ohne DA-Methoden dargestellt. Maximalwerte der Genauigkeit, bzw. Minimalwerte der Verluste, sind in Rot dargestellt.

Tabelle 6: Lernprozess mit zufälliger Drehung um Z-Achse. Maximale bzw. Minimale Werte in Rot. (Val. = Validierung)

Epoche	Loss	Genauigkeit	Val. Loss	Val. Genauigkeit	Val. IoU
Referenz	**0.0105**	**96.1%**	**0.0250**	**95.5%**	**0.646**
1	0.0143	96.8%	0.0226	95.3%	0.613
2	0.0124	97.0%	0.0216	95.4%	0.616
3	0.0117	97.1%	0.0215	95.5%	0.607
4	0.0130	96.8%	0.0259	94.6%	0.591
5	0.0125	96.9%	0.0221	95.4%	0.610
6	0.0115	97.1%	0.0217	95.4%	0.619
7	0.011	97.1%	0.0223	95.4%	0.614
8	0.0107	97.2%	0.0223	95.4%	0.621
9	0.0105	97.2%	0.0227	95.3%	0.614
10	0.0105	97.2%	0.0224	95.4%	0.611
11	0.0102	97.3%	0.0237	95.3%	0.607
12	0.0150	96.4%	0.0514	89.6%	0.488
13	0.0200	95.5%	0.0314	93.7%	0.562
14	0.0159	96.1%	0.0257	94.6%	0.601
15	0.0134	96.7%	0.0254	94.8%	0.596

Es ist festzustellen, dass die Rotation nicht zu einer Verbesserung der Validierungsgenauigkeit oder der IoU geführt hat. Wie die Konfusionsmatrix in Abbildung 50 zeigt, können alle Klassengenauigkeiten mit Ausnahme der Klassen Vegetation ($-0{,}4\%$) und der Menschgemacht ($-14{,}6\%$) verbessert werden.

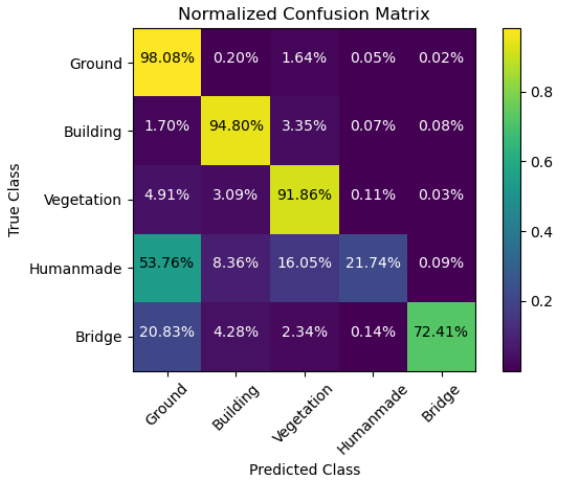


Abbildung 50: Konfusionsmatrix - Rotation um Z-Achse

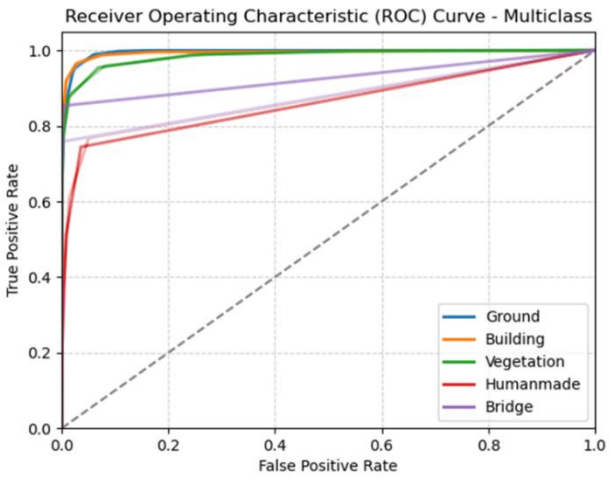


Abbildung 51: ROC-Kurven - Rotation um Z-Achse

Offensichtlich verschlechtert diese DA-Methode die Genauigkeit der Klasse Menschgemacht aufgrund ihrer Heterogenität. Die Klasse enthält verschiedene Objekte, von denen die meisten eine kleine Ausdehnung besitzen. Die zusätzliche geometrische Veränderung durch die Rotation scheint das

Modellrauschen zu erhöhen. Die ROC-Kurven in Abbildung 51 zeigen eine Verbesserung der Trennschärfe für Brücken hin. Die Resultate des Modelltrainings ohne Dataset Augmentation sind dieser Abbildung mit geringerer Sättigung dargestellt. Auch wenn die Gesamtgenauigkeit durch eine zufällige Drehung um die Z-Achse nicht gesteigert wird, kann die Genauigkeit der Brückenklasse optimiert werden. Offenbar ist das Modell durch die Rotation in der Lage, Brücken besser zu generalisieren. Da im Trainingsdatensatz nur wenige Brücken enthalten sind, stellt diese Methode eine sinnvolle Ergänzung für den Trainingsprozess dar.

5.3.2. Zufälliges Rauschen der Farbinformationen

Das Hinzufügen eines zufälligen Rauschens zu den Farbinformationen addiert den HSV-Attributen zufällige Werte hinzu, die einer Normalverteilung folgen. Dabei wird der Mittelwert der Normalverteilungen gleich Null gesetzt und ein plausibler Wert für die Standardabweichungen des jeweiligen Attributs verwendet:

$$s_{Hue} = 0{,}02 \quad s_{Saturation} = 0{,}1 \quad s_{Value} = 0{,}05$$

Die Plausibilität dieser Werte wurde anhand einiger exemplarischer Tests überprüft. Für das anschließende Training wird eine Wahrscheinlichkeit von 50% gewählt, sodass nur die Hälfte der Batches verrauscht wird. Die Auswahl der Batches erfolgt rein zufällig. Der Trainingsablauf ist in Tabelle 7 dargestellt.

Tabelle 7: Lernprozess mit Rauschen in Farbinformationen. Maximale bzw. minimale Werte in Rot. (Val. = Validierung)

Epoche	Loss	Genauigkeit	Val. Loss	Val. Genauigkeit	Val. IoU
Referenz	**0.0105**	**96.1%**	**0.0250**	**95.5%**	**0.646**
1	0.0105	96.1%	0.0247	95.4%	0.638
2	0.0095	97.3%	0.0258	95.5%	0.649
3	0.0087	97.5%	0.0254	95.5%	0.638
4	0.0086	97.5%	0.026	95.3%	0.638
5	0.0086	97.5%	0.0267	95.4%	0.621
6	0.0084	97.5%	0.0261	95.5%	0.627
7	0.0083	97.5%	0.0278	95.4%	0.633
8	0.0083	97.5%	0.0284	95.4%	0.635
9	0.0082	97.5%	0.0293	94.2%	0.581
10	0.0182	95.6%	0.0278	94.9%	0.615
11	0.0110	96.9%	0.0276	95.1%	0.610
12	0.0095	97.3%	0.0276	95.0%	0.609
13	0.0091	97.4%	0.0275	95.2%	0.622
14	0.0093	97.3%	0.0284	95.3%	0.636
15	0.0084	97.5%	0.0289	95.3%	0.622
16	0.0081	97.5%	0.0285	95.3%	0.628

Es zeigt sich, dass mit dieser Methode keine signifikante Verbesserung der Validierungsgenauigkeit und -IoU erreicht wird. Lediglich in der zweiten Epoche ergeben sich Verbesserungen der Validierungsgenauigkeit um 0,03% und der Validierungs-IoU um 0,3%, welche allerdings nicht signifikant sind. Auch die Genauigkeiten der einzelnen Klassen zeigen keine signifikante Verbesserung.

5.3.3. Rauschen in Z-Koordinate

Bei dieser Methode der Data Augmentation wird den Z-Koordinaten der Punktwolke ein zufälliges Rauschen hinzugefügt. Die Standardabweichung hierfür wird auf $5\,cm$ festgelegt. Die Wahrscheinlichkeit, mit der das Rauschen hinzugefügt wird, beträgt 50 %, sodass der Hälfte der Batches kein Rauschen hinzugefügt wird. Tabelle 8 zeigt die Ergebnisse für die einzelnen Epochen.

Tabelle 8: Lernprozess mit Anbringung eines zufälligen Rausches in der Z-Koordinate

Epoche	Loss	Genauigkeit	Val. Loss	Val. Genauigkeit	Val. IoU
Referenz	**0.0105**	**96.1%**	**0.0250**	**95.5%**	**0.646**
1	0.0092	97.4%	0.0249	95.4%	0.633
2	0.0090	97.4%	0.0267	94.9%	0.610
3	0.0135	96.5%	0.0281	94.7%	0.597
4	0.0102	97.2%	0.0253	95.3%	0.625
5	0.0090	97.4%	0.0256	95.4%	0.619
6	0.0087	97.5%	0.0263	95.4%	0.632
7	0.0085	97.5%	0.0262	95.4%	0.640
8	0.0106	97.0%	0.0251	95.3%	0.628
9	0.0087	97.5%	0.0261	95.3%	0.620
10	0.0085	97.5%	0.0268	95.4%	0.627
11	0.0083	97.5%	0.0267	95.4%	0.645
12	0.0084	97.5%	0.0267	95.4%	0.629
13	0.0082	97.5%	0.0271	95.4%	0.626
14	0.0081	97.6%	0.0263	95.5%	0.641
15	0.0081	97.6%	0.0271	95.3%	0.632

Es zeigt sich, dass mit dieser Methode keine Verbesserung der Genauigkeit erzielt werden kann. Auch bei den Genauigkeiten der einzelnen Klassen sind keine positive Verbesserungen zu erkennen. Die Genauigkeit der Klasse *Menschgemacht* sinkt zudem stark ab. In dieser Klasse befinden sich vorwiegend Fahrzeuge, die nur eine niedrige relative Höhe zur Fahrbahn aufweisen. Durch das zusätzliche Rauschen der Z-Koordinate können die Fahrzeuge kaum noch von der Straße unterschieden werden.

5.3.4. Farbänderung des gesamten Punktwolken-Batches

Für die Farbänderung des gesamten Batches werden die gleichen Standardabweichungen verwendet wie beim zufälligen Rauschen der Farbinformationen:

$$s_{Hue} = 0{,}02 \quad s_{Saturation} = 0{,}1 \quad s_{Value} = 0{,}05$$

Die Ergebnisse der einzelnen Epochen sind in Tabelle 9 dargestellt.

Tabelle 9: Lernprozess mit zufälliger Farbänderung. Maximale bzw. Minimale Werte in Rot. (Val. = Validierung)

Epoche	Loss	Genauigkeit	Val. Loss	Val. Genauigkeit	Val. IoU
Referenz	**0.0105**	**96.1%**	**0.0250**	**95.5%**	**0.646**
1	0.0093	97.3%	0.0256	95.4%	0.630
2	0.0179	96.2%	0.0309	93.6%	0.571
3	0.0144	96.3%	0.0267	94.6%	0.594
4	0.0111	97.0%	0.0257	95.1%	0.606
5	0.0100	97.2%	0.0247	95.2%	0.613
6	0.0093	97.3%	0.0255	95.3%	0.616
7	0.0090	97.4%	0.0255	95.3%	0.622
8	0.0088	97.4%	0.0265	95.3%	0.630
9	0.0087	97.4%	0.0263	95.4%	0.620
10	0.0086	97.5%	0.0262	95.4%	0.630
11	0.0085	97.5%	0.0261	95.3%	0.614
12	0.0084	97.5%	0.0262	95.4%	0.618
13	0.0126	96.6%	0.0262	94.9%	0.609
14	0.0099	97.1%	0.0267	95.1%	0.614
15	0.0092	97.3%	0.0269	95.3%	0.613
16	0.0087	97.4%	0.0265	95.3%	0.623

Die Ergebnisse der einzelnen Epochen zeigen, dass keine Genauigkeitssteigerung durch die Farbänderung des Batches erzielt werden kann. Da aus den Erkenntnissen aus dem Kap. 5.2.1 anzunehmen ist, dass das Modell primär von den geometrischen Attributen abhängt, wird durch diese Methode kein Genauigkeitsgewinn erzielt.

5.3.5. Zufälliger Verlust der Farbinformationen

Bei der Verwendung dieser Methode werden bei 30% der Punkte jedes Batches die Farbinformationen entfernt. Wie in Tabelle 10 ersichtlich ist, wird auch mit dieser Methode kein Genauigkeitsgewinn erzielt. Dies gilt sowohl für die Gesamtgenauigkeit als auch für die Genauigkeiten der einzelnen Klassen.

Tabelle 10: Lernprozess mit zufälligem Farbverlust. Maximale bzw. Minimale Werte in Rot. (Val. = Validierung)

Epoche	Loss	Genauigkeit	Val. Loss	Val. Genauigkeit	Val. IoU
Referenz	**0.0105**	**96.1%**	**0.0250**	**95.5%**	**0.646**
1	0.0099	97.3%	0.0261	95.2%	0.612
2	0.0103	97.1%	0.0280	94.5%	0.594
3	0.0132	96.7%	0.0260	95.0%	0.600
4	0.0098	97.3%	0.0264	95.1%	0.619
5	0.0092	97.4%	0.0261	95.3%	0.619
6	0.0089	97.4%	0.0275	95.2%	0.611
7	0.0087	97.5%	0.0284	95.2%	0.622
8	0.0086	97.5%	0.0291	95.0%	0.616
9	0.0085	97.5%	0.0309	95.1%	0.632
10	0.0085	97.5%	0.0306	95.0%	0.618
11	0.0134	96.6%	0.0434	91.6%	0.528
12	0.0157	96.1%	0.0287	94.4%	0.577
13	0.0110	97.0%	0.0278	94.9%	0.614
14	0.0094	97.3%	0.0270	95.1%	0.622
15	0.0089	97.4%	0.0280	95.1%	0.616
16	0.0086	97.5%	0.0293	95.1%	0.621

Wie bereits bei den anderen DA-Methoden zur Änderung der radiometrischen Informationen wird hier kein Genauigkeitsgewinn erzielt.

5.4. Verwendung verschiedener Punktanzahlen im Batching

Um die Punktwolke an das neuronale Netz zu übergeben ist ein Batching erforderlich. Das Vorgehen hierbei wurde in Kap. 4.2 erläutert und basiert auf den Ansatz von WINIWARTER & MANDLBURGER, G. (2019). Die Autoren bestimmten die Größe ihrer Batches anhand des limitierten GPU-Speichers. Allerdings ist anzunehmen, dass die Punktanzahl eines Batches einen Einfluss auf die Performance des PointNet++ Modells hat. In den Set-Abstraction-Layern wird durch das Farthest-Point-Sampling eine bestimmte Anzahl an Punkten jedes Batches selektiert, um Merkmale abzuleiten. Bei kleiner Batchgröße steigt die Gesamtanzahl der Batches, sodass in der Summe mehr Punkte in den SA-Layern ausgewählt und insgesamt mehr Merkmale berechnet werden. Gleichzeitig verringern sich bei kleiner Batchgröße die Kontextinformationen, da nur noch ein kleinerer Bereich in einer Punktwolke abgebildet wird.

In den bisher vorgestellten Untersuchungen wird eine Punktanzahl von $K = 100.000$ Punkt für das Batching gewählt, wobei eine maximale Gesamtgenauigkeit von 95,5% erzielt wurde. Nachfolgend soll die Performanceänderung durch Batchgrößen von $K = 50.000$ und $K = 200.000$ untersucht werden. Die Ergebnisse in jeder Epoche des Trainingsprozesses sind in Anlage H: *Trainingsprotokolle bei verschiedenen Batchgrößen* dargestellt. Die räumlichen Ausdehnungen der so generierten Batches sind in Abbildung 52 visualisiert.

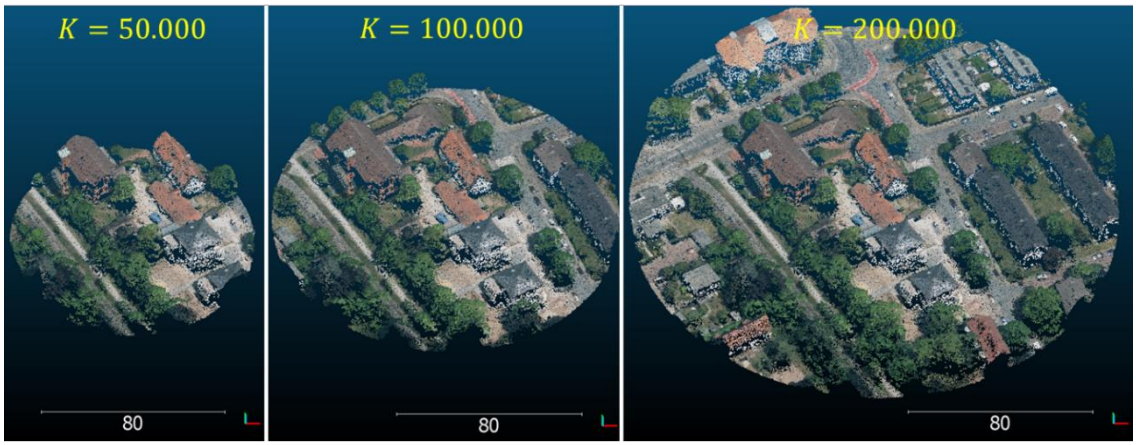


Abbildung 52: Batches mit verschiedener Punktanzahl. Links: 50.000; Mitte: 100.000; Rechts: 200.000

Bei der Verwendung von 200.000 Punkten pro Batch wurde eine maximale Validierungsgenauigkeit von 94,8% erreicht, was einem Genauigkeitsverlust von −0,7% gegenüber einer Batchgröße von 100.000 Punkten entspricht. Wie die Konfusionsmatrix in Abbildung 53 zeigt, wirkt sich die größere Punktanzahl vor allem negativ für die Klasse *Menschgemacht* aus, da hier ein Genauigkeitsverlust von 8,9% zu verzeichnen ist.

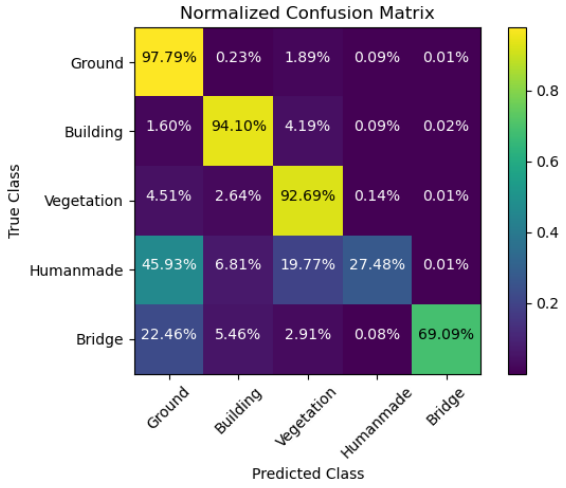


Abbildung 53: Konfusionsmatrix - Training mit Batchsize $K = 200.000$

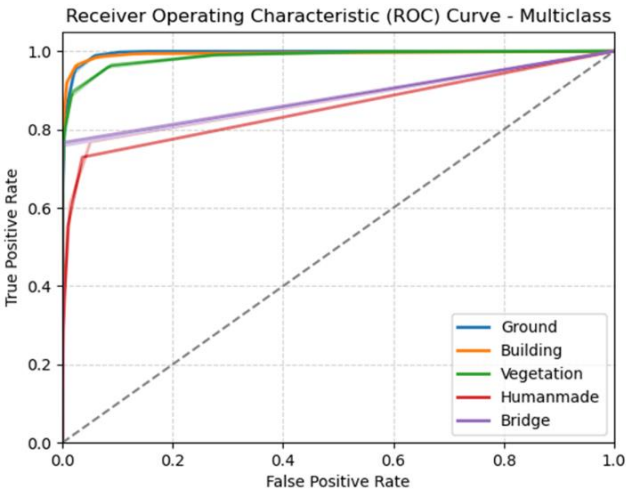


Abbildung 54: ROC-Kurven – Training mit Batchsize $K = 200.000$

Auch die ROC-Kurven in Abbildung 54 zeigen eine geringere Performance für diese Klasse, wobei die Kurven der anderen Klassen keine Änderung erfahren haben. Die ROC-Kurven bei einer Batchgröße von 100.000 Punkten sind hier durch eine geringere Sättigung dargestellt. Da Objekte in der Klasse *Menschgemacht* im Gegensatz zu den anderen Objekten klein sind, werden sie durch die größere Punktanzahl mutmaßlich nicht richtig erfasst.

Die Verwendung von 50.000 Punkten pro Batch erzielt einen positiven Einfluss auf die Gesamtperformance des Modells, sodass eine Validierungsgenauigkeit von 96,0% (+0,5%) erreicht werden. Gleichzeitig zeigt die Konfusionsmatrix in Abbildung 55 nur eine minimale Verbesserung der

Genauigkeit der Vegetationsklasse um 0,4%, während die Klassen *Menschgemacht* und *Brücke* eine Verringerung von −2,5% und −3,1% hinnehmen müssen. Auch die ROC-Kurven zeigen nur minimale Änderungen gegenüber einer Batchgröße von 100.000 Punkten, wie in Abbildung 56 dargestellt.

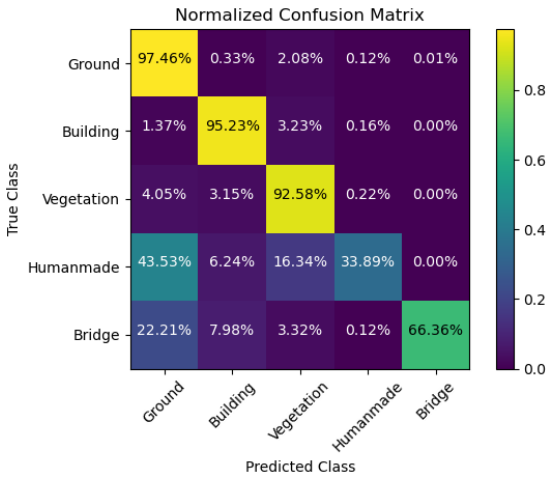


Abbildung 55: Konfusionsmatrix - Training mit Batchsize $K = 50.000$

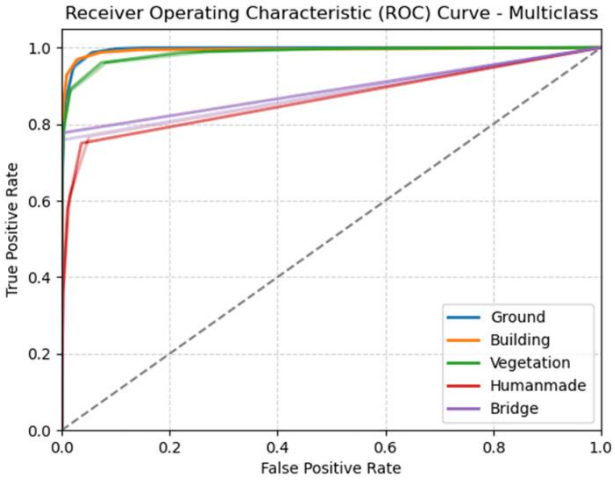


Abbildung 56: ROC-Kurven – Training mit Batchsize $K = 50.000$

Auch wenn die Gesamtgenauigkeit durch die Verwendung von 50.000 Punkten pro Batch gesteigert werden kann, verringern sich die Klassengenauigkeiten, mit Ausnahme der Vegetation. Offensichtlich sind die Kontextinformationen bei dieser Batchgröße für die Klassen *Menschgemacht* und *Brücke* nicht mehr ausreichend, um ein optimales Ergebnis zu erzielen. Zudem bewirkt die gesteigerte Anzahl an Trainingsbatches auch eine höhere zeitliche Dauer des Trainingsprozesses, sodass mit der eingesetzten Hardware eine Trainingsepoche ca. 4 Minuten länger benötigt.

5.5. Veränderung der Netzarchitektur

Bisher wurde das PointNet++ nach der Architektur von WINIWARTER (2018) mit drei Set-Abstraction-Layers verwendet. Die Anzahl dieser kann einen erheblichen Einfluss auf die Performance des Modells nehmen. KADA & KURAMIN (2021) erzielten beispielsweise eine deutliche Genauigkeitssteigerung durch die Verwendung eines vierten SA-Layers. Nachfolgend soll auch hier die Verwendung eines zusätzlichen SA-Layers untersucht werden. Dazu wird die Netzarchitektur aus Abbildung 38 auf die in Tabelle 11 ersichtlichen Parameter erweitert. Die Ergebnisse jeder Epoche des Trainings sind wiedergegeben in Anlage H: *Trainingsprotokolle bei Variation der Netzarchitektur*.

Tabelle 11: Veränderte Netzarchitektur von des PointNet++ mit zusätzlichem Set-Abstraction-Layer

Set-Abstraction-Layer 1 • FPS: 16.384 Punkte • Ball-Abfrage: $\mathbf{1,0}\ \boldsymbol{m}$ Radius; 16 Punkte • MLPs: 64, 64, 128	Feature-Propagation-Layer 1 • MLPs: 128, 64

Set-Abstraction-Layer 2 • FPS: 8.192 Punkte • Ball-Abfrage: $\mathbf{2,0}\ \boldsymbol{m}$ Radius; 16 Punkte • MLPs: 64, 64, 128	Feature-Propagation-Layer 2 • MLPs: 128, 128
Set-Abstraction-Layer 3 • FPS: 4.096 Punkte • Ball-Abfrage: $\mathbf{5,0}\ \boldsymbol{m}$ Radius; 64 Punkte • MLPs: 128, 128, 256	Feature-Propagation-Layer 3 • MLPs: 256, 256
Set-Abstraction-Layer 4 • FPS: 2.048 Punkte • Ball-Abfrage: $\mathbf{15,0}\ \boldsymbol{m}$ Radius; 64 Punkte • MLPs: 128, 128, 256	Feature-Propagation-Layer 4 • MLPs: 256, 256

Durch die Veränderung der Netzarchitektur kann eine Gesamtgenauigkeit von 94,9% erreicht werden, was einer Änderung von −0,6% gegenüber der Implementierung mit drei SA-Layern entspricht. Mit Ausnahme der Bodenklasse haben alle Klassen einen Genauigkeitsverlust erfahren, wie die Konfusionsmatrix in Abbildung 57 zeigt. Am deutlichsten verringern sich die Genauigkeiten der stark unterrepräsentierten Klassen *Menschgemacht* und *Boden* (−16,8% und −13,4%). Auch die ROC-Kurven zeigt eine minimal verringerte Performance, wie in Abbildung 58 angedeutet.

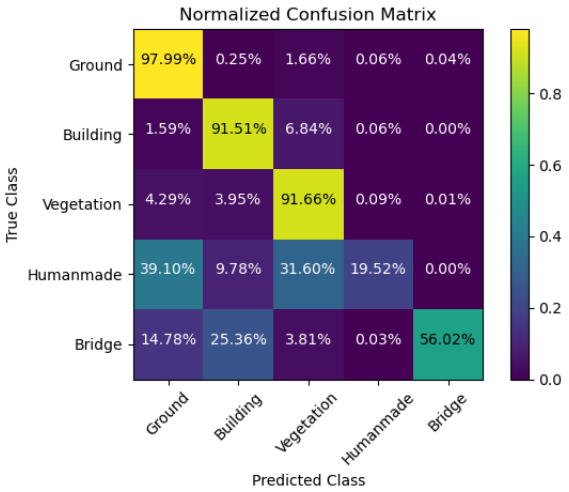


Abbildung 57: Konfusionsmatrix – Netzarchitektur mit vier SA-Layern

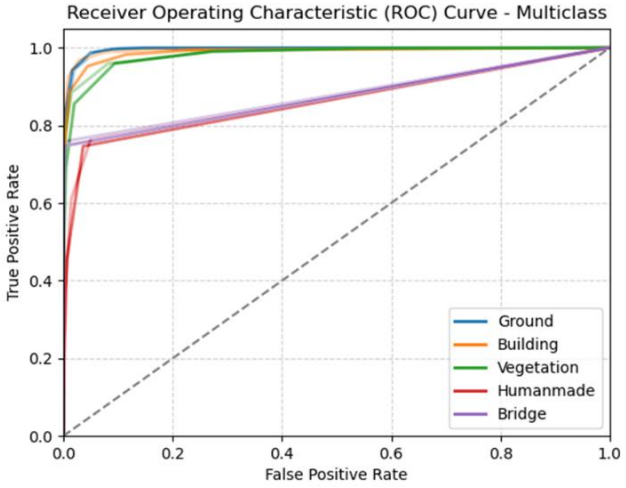


Abbildung 58: ROC-Kurven – Netzarchitektur mit vier SA-Layern

KADA & KURAMIN (2021) begründeten die gesteigerten Performance bei der Verwendung eines vierten SA-Layer in ihrer Untersuchung durch eine hohe Punktdichte. Diese Aussage stimmt mit den hier resultierenden Erkenntnisse überein, da der verwendete Trainingsdatensatz keine hohe Punktdichte aufweist.

5.6. Fine-Tuning der Hyperparameter

Eine Stellschraube zur Optimierung eines Deep-Learning-Modells ist das Fine-Tuning einiger Parameter. So kann die Performance durch die Verwendung eines anderen Optimierers gesteigert werden. Beispielsweise erzielten QIAN et al. (2022) eine Genauigkeitssteigerung von 0,5% durch die Verwendung von *AdamW* als Optimierer. Hierbei handelt es sich um eine Weiterentwicklung von LOSHCHILOV & HUTTER (2017) die eine Gewichtsverlust-Komponente (*Weight Decay*) hinzufügt.

Einer der wichtigsten Hyperparameter im Trainingsprozess ist die Lernrate, welche die Anpassung der Gewichte des Netzwerks in Bezug auf den Verlustgradienten definiert. In vielen Deep Learning Anwendung wird die Lernrate innerhalb des Trainingsprozess variiert, um ein optimales Minimum zu erzielen. So können Lernraten beispielsweise linear oder exponentiell in jeder Epoche reduziert werden. Diese Methoden werden auch für das hier implementierte Modell untersucht und nachfolgend dargestellt. Die Ergebnisse jeder Epoche des Lernprozesses sind in Anlage I: *Fine-Tuning des PointNet++ Modells* zu finden.

5.6.1. Exponentielle Abnahme der Lernrate

In den bisherigen Untersuchungen wird eine Lernrate von 0,001 verwendet. Um ein bestmögliches lokales Minimum der approximierten Funktion zu erhalten ist es sinnvoll die Lernrate (lr) iterativ zu verringern. Verschiedene sogenannte *Learning-Rate-Scheduler* bieten die Möglichkeit, die Lernrate linear, exponentiell oder durch individuelle Funktionen iterativ zu verändern. In dieser Untersuchung wird die initiale Lernrate (lr_0) von 0,001 exponentiell mit dem Faktor $\lambda = 0{,}85$ mit jeder Epoche (epo) verringert. Dabei folgt die Lernrate dem in Formel 25 dargestellten Schema:

Formel 25: Exponentielle Verringerung der Lernrate

$$lr = lr_0 \cdot \lambda^{epo}$$

Durch die Verwendung der abnehmenden Lernrate kann eine Genauigkeitssteigerung von insgesamt +0,1% erzielt werden. In Abbildung 59 ist die Validierungsgenauigkeit jeder Epoche und die Referenzgenauigkeit dargestellt.

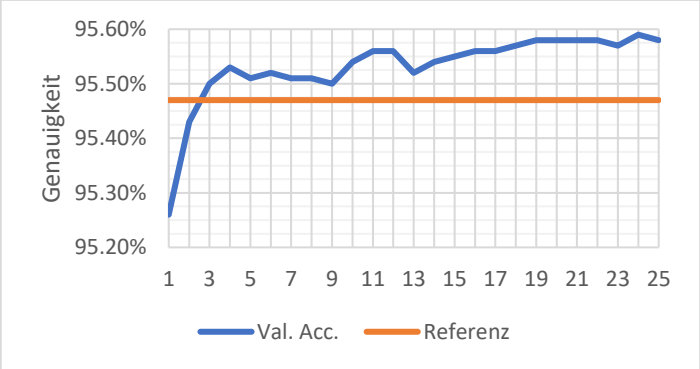


Abbildung 59: Validierungsgenauigkeit pro Epoche bei Verwendung einer sinkenden Lernrate

Wie zu erkennen ist, steigt durch die sinkende Lernrate die Validierungsgenauigkeit konstant über den Referenzwert von 97,5%. Daraus ist zu schließen, dass die Verwendung der exponentiell abnehmenden Lernrate die Gesamtgenauigkeit verbessern kann – wenn auch nur im geringen Maße.

5.6.2. *Verwendung des Optimierers AdamW*

Um zu untersuchen, ob die Verwendung von AdamW eine Auswirkung auf die Performance des Modells hat, wird das Referenzmodell mit dem Adam-Optimierer als Ausgangspunkt für den Lernprozess verwendet und anschließend in weiteren Epochen der AdamW-Optimierer verwendet. Hierbei kann allerdings keine signifikante Verbesserung der Gesamtgenauigkeit erzielt werden. Mit Ausnahme der Klasse *Menschgemacht* ($-2{,}0\%$) sind ebenfalls nur geringfüge Veränderungen ($\leq 0{,}6\%$) der Klassengenauigkeiten zu beobachten, wie die Konfusionsmatrix in Abbildung 60 darstellt. Die ROC-Kurven (Abbildung 61) verlaufen absolut identisch.

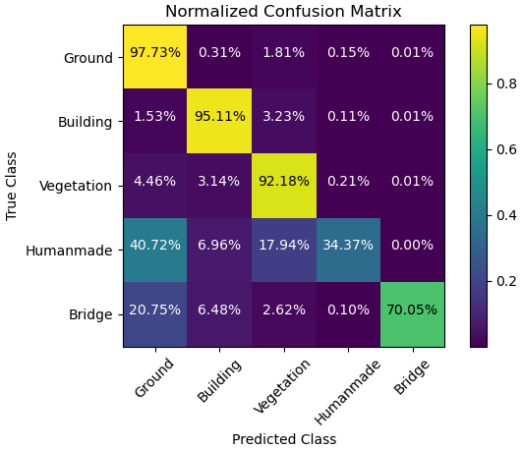


Abbildung 60: Konfusionsmatrix – Verwendung des AdamW-Optimierers

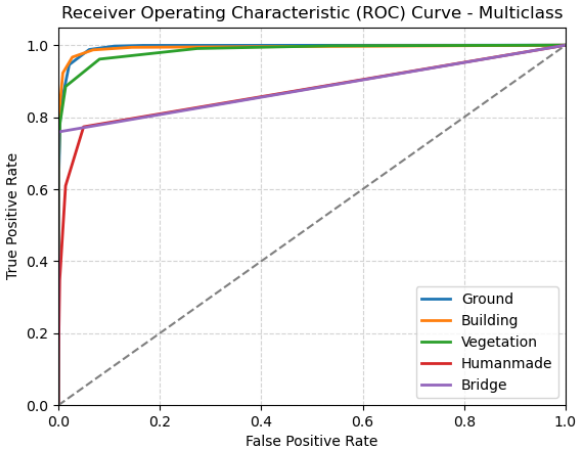


Abbildung 61: ROC-Kurven – Verwendung des AdamW-Optimierers

6. Finales Training & Evaluierung der Ergebnisse

Aus den Untersuchungen in Kapitel 5 ergeben sich Tendenzen für ein optimales Training. In einem abschließenden Trainingsprozess sollen die Einstellungen und Parameter verwendet werden, die eine Verbesserung des Modells erwarten lassen. Da die Verwendung der optimalen Attributteilmenge nicht eindeutig belegt werden konnte, werden zwei Trainingsprozesse gestartet: Ein Trainingsprozess mit allen acht Attributen und ein Trainingsprozess bei dem nur die Geometrie und die Radiometrie (XYZ + HSV) verwendet werden. Durch das gewählte Batching werden dieselben Punkte in mehreren Batches abgebildet, sodass durch die Rückführung eine mittlere Genauigkeit berechnet werden kann. Nachfolgend sollen die Ergebnisse der Trainingsprozesse und die Auswirkungen der Rückführung der Batches dargestellt werden. Anschließend werden die Ergebnisse der berechneten Modelle betrachtet und evaluiert.

6.1. Finaler Trainingsprozess

Für das abschließende Training des Modells wird der Focal Loss mit $\gamma = 4$ als Verlustfunktion verwendet und zur Dataset-Augmentation wird lediglich die Drehung um die Z-Achse verwendet. Als Optimierer wird AdamW mit einer Lernrate von 0,001 verwendet. Gleichzeitig wird ein Learning-Rate-Scheduler mit exponentieller Reduzierung der Lernrate mit einem Faktor von $\lambda = 0.9$ verwendet. Dabei wird die Lernrate erst ab der zehnten Epoche reduziert. Die Ergebnisse jeder Epoche der beiden Trainingsprozesse sind in Anlage J: *Finale Trainingsprozesse* abgelegt.

Bei der Verwendung von allen verfügbaren Attributen wird eine Validierungsgenauigkeit von 95,8% erreicht, was eine Steigerung um +0,3% gegenüber dem vergleichbaren Modelltraining aus Kapitel 5.1 bedeutet. Alle in Abbildung 62 dargestellten Klassengenauigkeiten zeigen mit Ausnahme der Menschgemacht Klasse (−0,2%) eine Steigerung: Boden +0,6%, Gebäude +0,2%, Vegetation +0,4% und Brücke +16,8%. Vor allem die Brückenklasse zeigt eine starke Verbesserung, was auch die ROC-Kurven in Abbildung 63 zeigt. Hier sind die ROC-Kurven des Modells mit der bisher besten Performance in geringerer Sättigung dargestellt. Die ROC-Kurven zeigen zudem eine Verbesserung der Performance für die Menschgemacht Klasse.

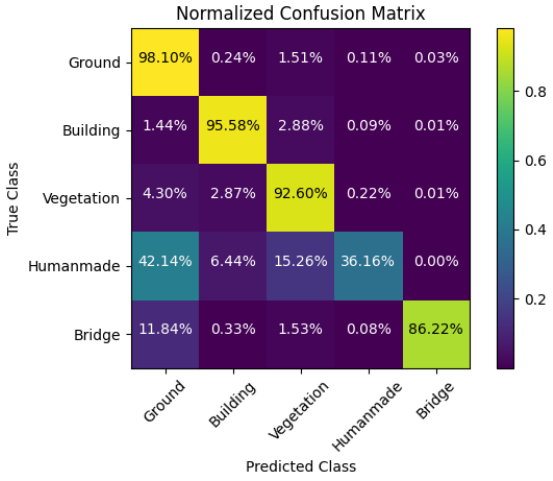


Abbildung 62: Konfusionsmatrix des finalen Trainings mit allen Attributen

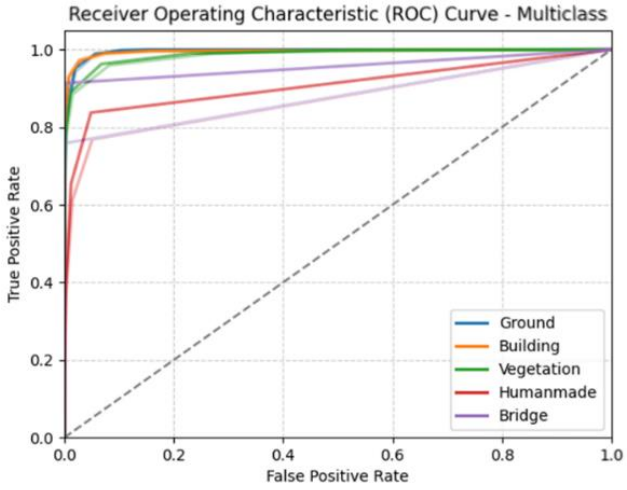


Abbildung 63:ROC-Kurven des finalen Trainings mit allen Attributen

Die Verwendung der sechs Attribute (XYZ + HSV) für das Training erzielt eine Gesamtgenauigkeit von 96,0%, was eine Steigerung von +0,4% gegenüber dem Training mit sechs Attributen aus Kap. 5.2.2 bedeutet. Wie die Konfusionsmatrix in Abbildung 64 zeigt, können die Klassengenauigkeiten ausnahmslos gesteigert werden: Boden +0,2%, Gebäude +2,2%, Vegetation +0,4%, Menschgemacht +5,3% und Brücken +22,65%. Erneut wird beobachtet, dass sich vor allem die Brückenklasse stark verbessert, wie auch die ROC-Kurve in Abbildung 65 darstellt.

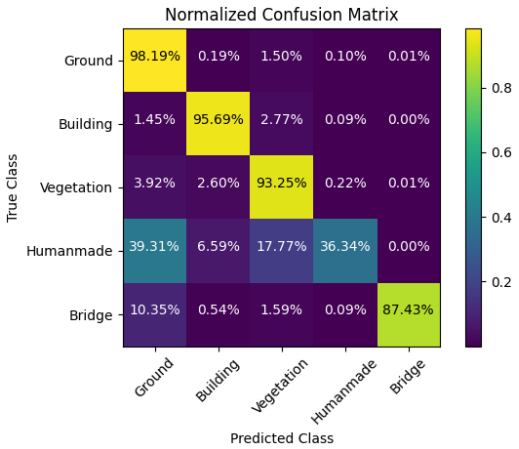


Abbildung 64: Konfusionsmatrix des finalen Trainings mit Geometrie & Radiometrie

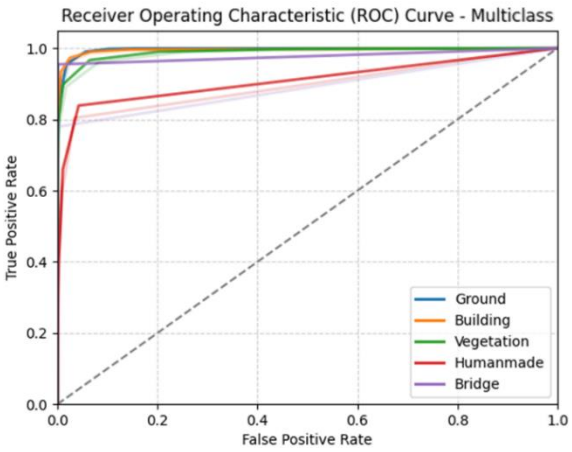


Abbildung 65:ROC-Kurven des finalen Trainings mit Geometrie & Radiometrie

Untereinander verglichen zeigt die Verwendung der sechs Attribute eine insgesamt bessere Performance. Sowohl die Validierungsgenauigkeit als auch die Klassengenauigkeit liegen ausnahmslos höher. Die ROC-Kurven zeigen einen ähnlichen Verlauf, wobei die ROC-Kurve der Brückenklasse bei Verwendung von sechs Attributen eine bessere Trennschärfe aufweist.

6.2. Rückführung der Batches zu einer Punktwolke

Die bisher betrachteten Genauigkeitsmetriken beziehen sich auf einzelne Batches. Allerdings werden durch das gewählte Batching dieselben Punkte in mehreren Batches abgebildet, wodurch derselbe Punkt mehrfach einer Klasse zugeordnet wird. Diese Redundanz kann bei der Rückführung der Batches in eine Punktewolke genutzt werden, um eine mittlere Wahrscheinlichkeit für jede Klasse zu berechnen. Die Anzahl an Batches pro Punkt sind in einem Histogramm in Abbildung 66 dargestellt.

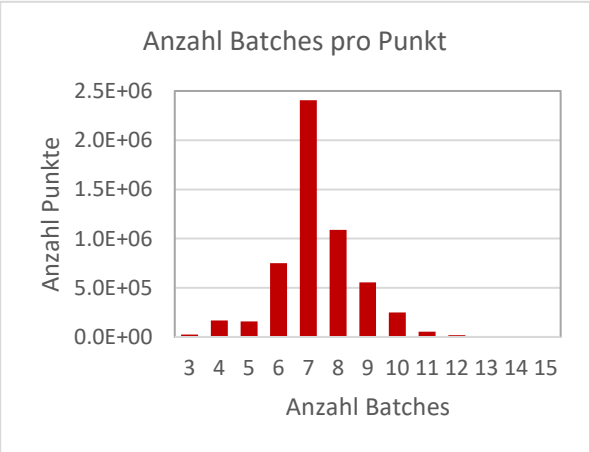


Abbildung 66: Histogramm Anzahl der Batches pro Punkt

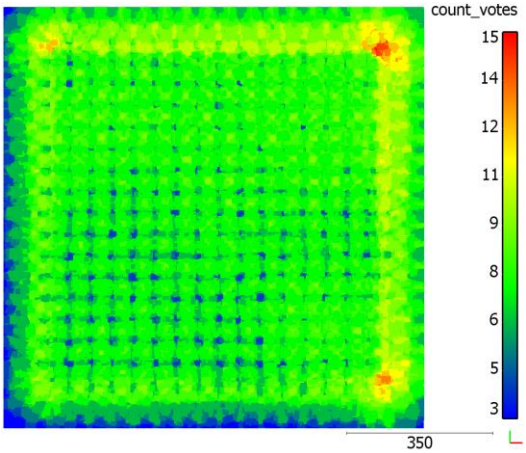


Abbildung 67: Anzahl von Batches pro Punkt

Abbildung 67 zeigt welcher Punkt in wie vielen Batches auftritt. Hier sind die überlappenden kreisförmigen Batches erkennbar. Durch das Batching wird jeder Punkt in mindestens drei Batches

abgebildet, während ca. 44% der Punkte in sieben Batches auftreten. Durch die Rückführung des Validierungsbatches zu einer gesamten Punktwolke kann eine Genauigkeit von 96,4% erzielt werden, sodass die Gesamtgenauigkeit um +0,6% gegenüber der Genauigkeit für einzelne Batches (Kap. 6.1) gesteigert werden kann. Wie die Konfusionsmatrix in Abbildung 68 zeigt, können die Genauigkeiten aller Klassen gesteigert werden. Insbesondere die ROC-Kurve für die Brückenklasse wird auf einen nahezu optimalen Wert gesteigert, wie Abbildung 69 zeigt. 10,6% aller Brückenpunkte werden fälschlicherweise als Boden klassifiziert. Für die Klasse Menschgemacht kann insgesamt lediglich eine Genauigkeit von 38,5% erreicht werden, wobei 41,4% der Menschgemacht-Punkte als Boden klassifiziert werden.

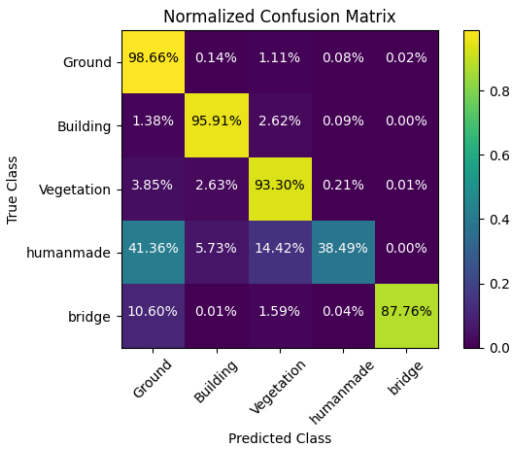


Abbildung 68: Konfusionsmatrix nach Rückführung in eine gesamte Punktwolke des Modells mit allen Attributen

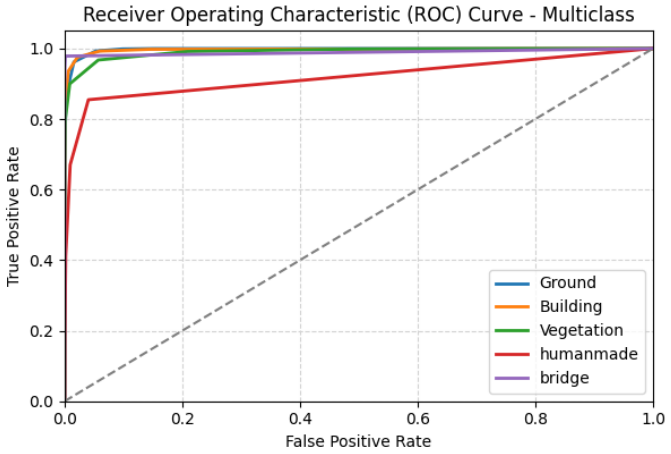


Abbildung 69:ROC-Kurven nach Rückführung in eine gesamte Punktwolke des Modells mit allen Attributen

Bei der Verwendung von sechs Attributen kann durch die Rückführung der Batches in eine gesamte Punktewolke eine Genauigkeit von 96,5% erreicht werden, was einer Steigerung von +0,5% gegenüber der Validierung pro Batch entspricht. Auch hier wird bei allen Klassen eine Steigerung der Genauigkeit erzielt, wie in der Konfusionsmatrix in Abbildung 70 zu erkennen ist. Die ROC-Kurve für die Brückenklasse (Abbildung 71) erzielt nun eine optimale Kurve, was auf eine sehr niedrige Falsch-Positiv-Rate schließen lässt. Gleichzeitig werden noch 8,2% aller Brückenpunkte als Boden klassifiziert. Auch hier kann für die Klasse Menschgemacht nur eine Genauigkeit von 38,1% erzielt werden, wobei 38,1% der Menschgemacht-Punkte als Boden und 17,4% der Punkt als Vegetation klassifiziert werden.

Untereinander verglichen weist die Verwendung von sechs Attributen eine gering verbesserte Performance gegenüber der Verwendung aller Attribute auf. Die Gesamtgenauigkeit ist hier um +0,1% besser, während die Genauigkeiten der Klassen Boden, Gebäude, Vegetation und Menschgemacht in einem ähnlichen Bereich liegen. Einzig die Brückenklasse wird bei der Verwendung von sechs Attributen um +2,7% verbessert. In Anlage K: *Klassifizierte Punktwolke aus dem Validierungsdatensatz* sind die klassifizierten Punktwolken herunterladbar.

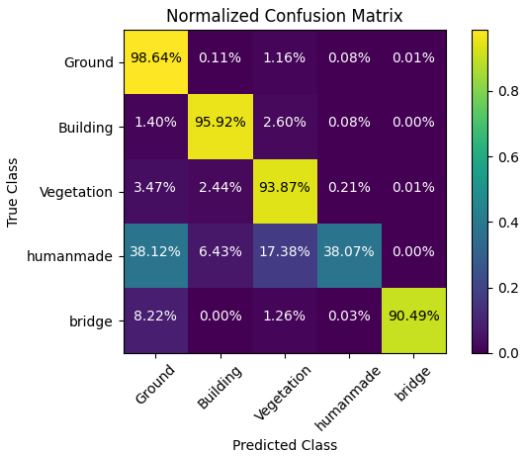


Abbildung 70: Konfusionsmatrix nach Rückführung in eine gesamte Punktwolke des Modells mit Geometrie und Radiometrie

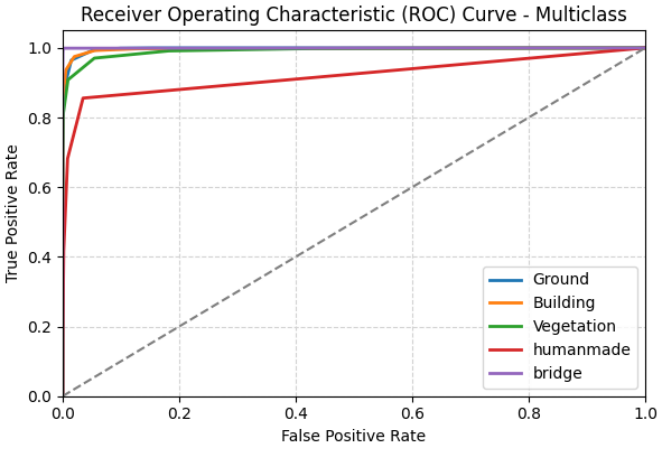


Abbildung 71:ROC-Kurven nach Rückführung in eine gesamte Punktwolke des Modells mit Geometrie und Radiometrie

6.3. Evaluierung der Modelle

Die trainierten Modelle erreichen, mit Ausnahme für die Menschgemacht-Klasse, für alle Klassen eine Genauigkeit von $> 87{,}7\%$. Die Klasse Menschgemacht erzielt unabhängig von der Attributanzahl lediglich eine Genauigkeit von $\sim 38\%$, wobei die meisten Punkte Menschgemacht-Punkte als Boden klassifiziert werden. Diese visuelle Betrachtung der Klassifikation in Abbildung 72 zeigt beispielhaft, dass vor allem Fahrzeuge als Bodenpunkte klassifiziert werden.

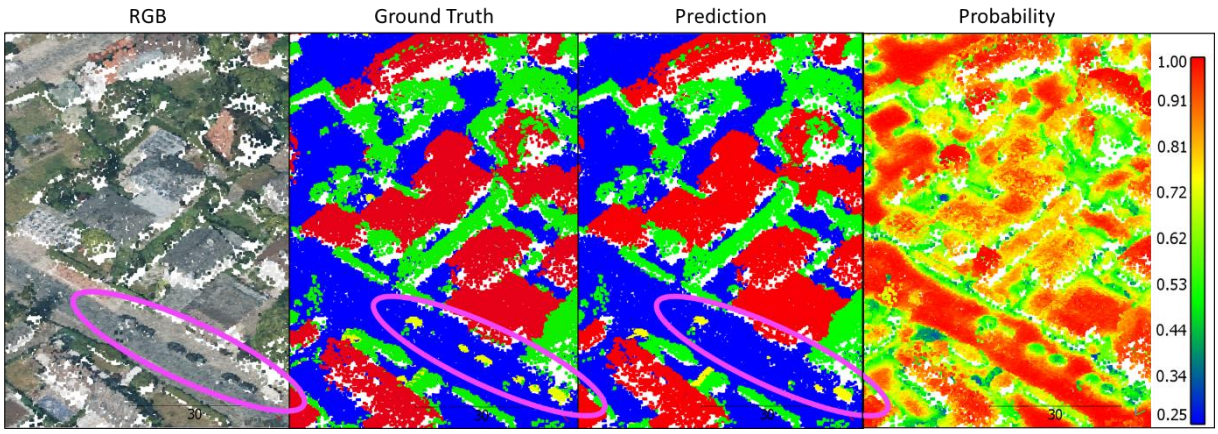


Abbildung 72: Fehlerhafte Klassifikation von Fahrzeugen als Boden. Hohe Wahrscheinlichkeiten in Rot, geringe in Blau.

Es ist zu erkennen, dass die Wahrscheinlichkeiten der tatsächlichen Fahrzeuge geringer ausfallen als beispielweise für den Boden. Die Modelle sind offensichtlich nicht in der Lage anhand der Attribute ausreichend zwischen Boden und Fahrzeugen zu unterscheiden. Dies kann auf verschiedene Ursachen hindeuten. Zum einen weisen die Fahrzeuge aufgrund des Messrauschens in der Höhe nur geringe Höhenveränderung im Gegensatz zum Boden auf, sodass Fahrzeuge ausschließlich anhand der Geometrie nur schwer zu identifizieren sind. Zum anderen ist diese Klasse in den Trainingsdaten

deutlich unterrepräsentiert und zudem teilweise ungenau klassifiziert, da der CSF viele Fahrzeuge als Boden klassifiziert hat. Darüber hinaus weist diese Klasse eine höheres Rauschen in der Klassenzuordnung auf und sie ist sehr heterogen, da viele verschiedene Objekte enthalten sind.

Für die Brückenklasse wird durch die Verwendung aller Attribute eine geringe Genauigkeit erzielt ($-2{,}7\%$). Abbildung 73 zeigt beispielhaft die visuellen Ergebnisse der Klassifikation mit den beiden Modellen: Oben mittig bei der Verwendung aller Attribute; unten mittig bei der Verwendung der sechs Attribute. Auch hier wird auf eine bessere Performance bei der Verwendung der sechs Attribute hingedeutet. Es wird ersichtlich, dass vor allem mittig der Straße die Kanten abgerundet sind. Dieser Effekt tritt bei der Verwendung aller Attribute stärker auf als bei der Verwendung von sechs Attributen. Allerdings ist es auch für den Menschen schwierig den Beginn und das Ende der Brücke auf der Straße zu beurteilen, da sich sowohl die Geometrie als auch die Radiometrie beim Übergang zwischen Boden und Brücke nicht ändert.

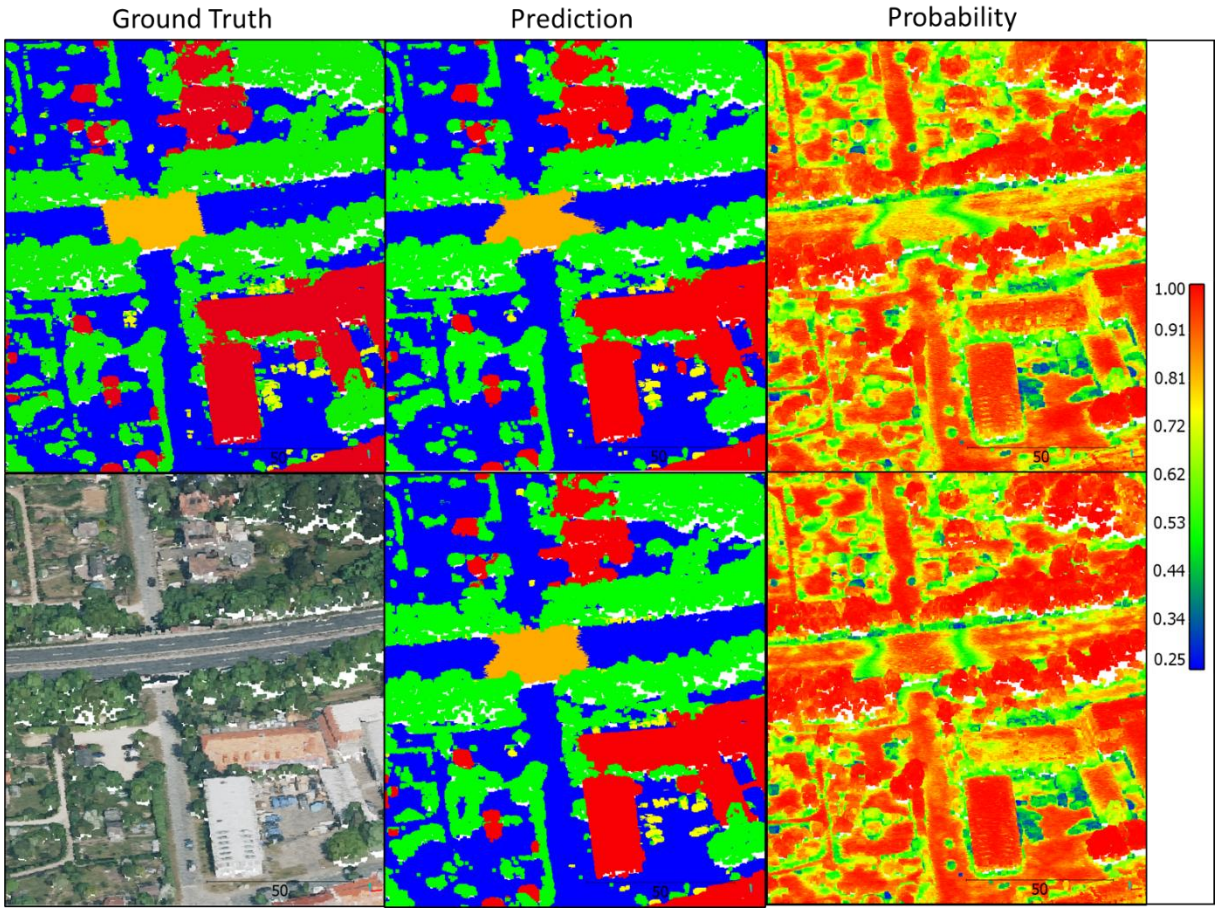


Abbildung 73: Klassifikation einer Brücke. Oben: bei Verwendung aller Attribute. Unten: Bei Verwendung von sechs Attributen

Beide Modelle erzielen hohe Wahrscheinlichkeiten für die Klassenzuordnung, wobei die Hälfte aller Punkte eine Wahrscheinlichkeit $> 80\%$ erreicht. Gleichzeitig erzielt das Modell bei der Verwendung von sechs Attributen deutlich häufiger Wahrscheinlichkeiten über 90%, wie das Histogramm in Abbildung 74 zeigt. Dieses Modell ist sich in der Klassenzuordnung also sicherer. Beide Modelle

erzielen vor allem für Fahrzeuge eine geringe Wahrscheinlichkeit, aber auch bei kleinen Hecken und Zäunen ist sich das Modell unsicher. Zudem ist zu beobachten, dass Grenzbereiche zwischen zwei Klassen eine geringe Wahrscheinlichkeit erreichen.

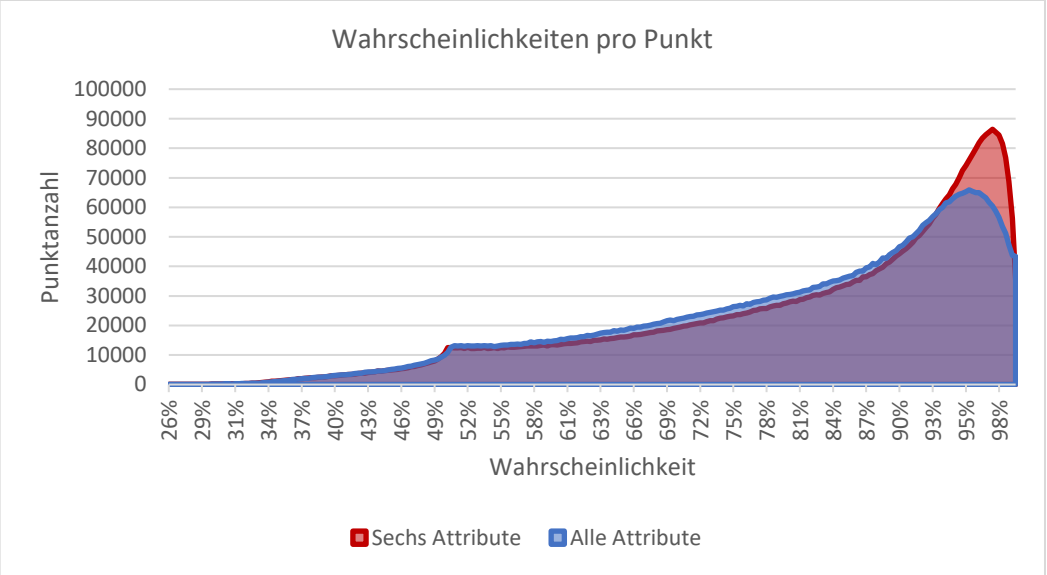


Abbildung 74: Histogramm der erreichten Wahrscheinlichkeiten der Klassenzuordnungen

Darüber hinaus wird beobachtet, dass die Klassifikation mithilfe der Modelle teilweise eine höhere Qualität liefert als die Ground Truth bietet. In Abbildung 75 ist ersichtlich, dass ein Gebäude richtigerweise als solches klassifiziert wird, während es im Ground-Truth-Datensatz als Vegetation gekennzeichnet ist. Dies lässt darauf schließen, dass die tatsächliche Validierungsgenauigkeit eher höher ist als es die Berechnungen widerspiegeln.

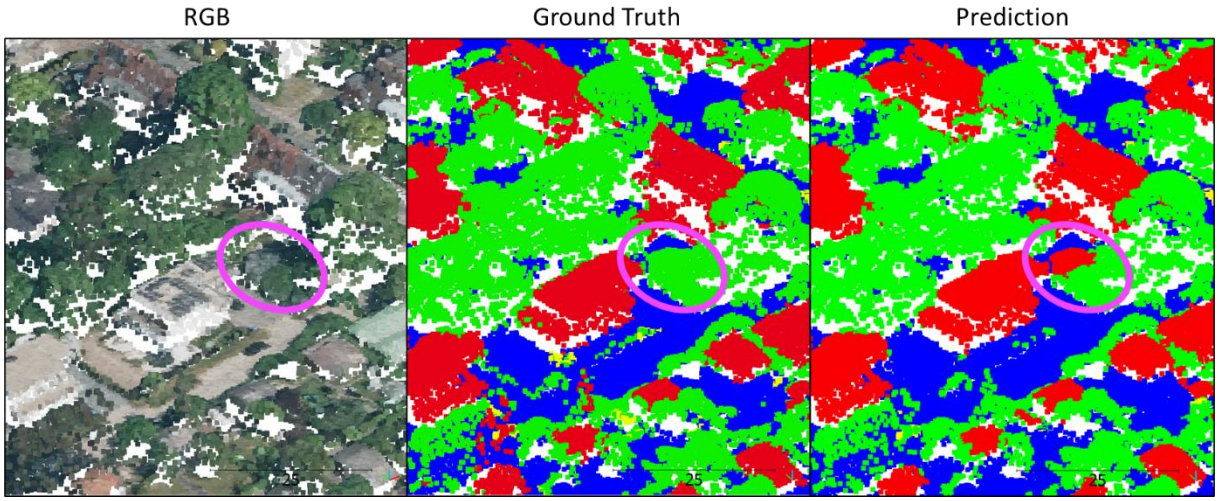


Abbildung 75: Fehlerhafte Zuordnung in der Ground Truth

Die beiden trainierten Modelle erzielen Ergebnisse in einem ähnlichen Genauigkeitsbereich und die Performance ist vergleichbar. Daher kann angenommen werden, dass durch die Verwendung der zusätzlichen photogrammetrischen Attribute keine Genauigkeitssteigerung erreicht werden kann. Die

Gesamtgenauigkeit des Modells bei der Verwendung der Radiometrie und der Geometrie liegt um $+0{,}1\%$ oberhalb der Genauigkeit mit allen verfügbaren Attributen. Auch die Brückenklasse zeigt hier eine bessere Genauigkeit ($+2{,}7\%$). Gleichzeitig sinkt die ohnehin geringe Genauigkeit der Menschgemacht-Klasse ($-0{,}4\%$). Weitere beispielhafte Darstellungen der klassifizierten Punktwolke sind in Anlage L: *Beispielhafte Darstellungen der klassifizierten Punktwolke* dargestellt.

7. Fazit

Im Rahmen dieser Arbeit wird der gesamte Arbeitsablauf zur Erstellung eines Deep-Learning-Modells zur semantischen Segmentierung von photogrammetrischen Punktwolken dargestellt. Zunächst wird mit Hilfe verschiedener unüberwachter Verfahren ein umfangreicher Ground-Truth-Datensatz generiert, der eine elementare Grundlage für das Training des Modells bietet. Die Generierung des Datensatzes stellt eine umfangreiche und zeitintensive Aufgabe dar. Gleichzeitig ist der Umfang und die Qualität der Annotation von entscheidender Bedeutung für die Leistungsfähigkeit des Klassifikators. Durch die Kombination verschiedener unüberwachter Lernverfahren wird ein effizienter Weg aufgezeigt, einen qualitativ hochwertigen Datensatz zu generieren. Auch wenn die Ground Truth ein gewisses Rauschen in der Klassenzuordnung aufweist, wird ein angemessener Kompromiss zwischen Aufwand und Nutzen erzielt. Der auf diese Weise erzeugte Ground-Truth-Datensatz umfasst 12 km² urbanes Gelände, in dem viele heterogene Objekte repräsentiert sind. Der Datensatz wird verwendet, um eine PointNet++ Architektur zu trainieren, wobei ein Großteil der gewählten Parameter auf der Arbeit von WINIWARTER & MANDLBURGER, G. (2019) basiert.

Um den größtmöglichen Nutzen aus den Trainingsdaten zu ziehen, werden verschiedene Experimente mit dem Modell durchgeführt und unterschiedliche Parameter und Dataset-Augmentation-Methoden getestet. Insbesondere der Focal Loss mit der Verwendung von $\gamma = 4$ hat sich als geeignete Verlustfunktion für das Training herausgestellt. Weiterhin kann gezeigt werden, dass die zusätzlichen photogrammetrischen Informationen der Punktpräzision und der Anzahl der Stereomodelle keinen Mehrwert für das DL-Modell bieten. Darüber hinaus scheint vor allem die Geometrie eine entscheidende Rolle für die Leistungsfähigkeit des Modells zu spielen, da ein Training allein auf Basis der Geometrie bereits eine Gesamtgenauigkeit von 94% erreicht. Die radiometrische Information ist jedoch insbesondere für die Klasse Menschgemacht von entscheidender Bedeutung. Im Weiteren wird der Einfluss verschiedener Dataset-Augmentation-Methoden untersucht. Dabei kann nur mit der Rotation um die Z-Achse eine Verbesserung des Modells erzielt werden. Radiometrische Veränderungen der Punktwolken bringen keine Verbesserung der Performance, was die These stützt, dass das Modell hauptsächlich auf Basis der Geometrie klassifiziert. Des Weiteren kann festgestellt werden, dass die Größe des Batchings einen Einfluss auf die Performance des Modells hat. So musste bei der Verwendung von 200.000 Punkten pro Batch ein Genauigkeitsverlust im Vergleich zu 100.000 Punkten in Kauf genommen werden. Die Verwendung von 50.000 Punkten pro Batch führt eher zu einer

Verbesserung der Genauigkeit, wobei sich die Verbesserung hauptsächlich auf die Vegetationsklasse auswirkt. Gleichzeitig erhöht eine geringere Punktzahl pro Batch auch die Gesamtanzahl der Batches, was sich negativ auf die Berechnungszeit einer Trainingsepoche auswirkt. Für diese Arbeit wird eine Batchgröße von 100.000 Punkten als Kompromiss verwendet. Darüber hinaus wird die Relevanz der gewählten Hyperparameter des Trainings hervorgehoben. Während die Verwendung des AdamW-Optimierers keinen großen Einfluss auf die Gesamtgenauigkeit hat, kann durch die exponentielle Abnahme der Lernrate eine deutliche Genauigkeitssteigerung erzielt werden.

In einem abschließenden Trainingsprozess werden die optimalen Parameter angewendet, die sich in den Experimenten als vielversprechend erwiesen haben. Zwei verschiedene Modelle werden trainiert: Das eine nutzt alle verfügbaren Attribute für das Training. Das andere verwendet nur die radiometrischen und geometrischen Informationen der Punktwolke. Durch die Überlappung der Batches wird jeder Punkt in mehreren Batches abgebildet, sodass derselbe Punkt mehrfach in unterschiedlichen Kontexten klassifiziert wird. Durch diese Redundanz kann das Modell noch zuverlässiger und korrekter klassifizieren. Nach dem abschließenden Training kann eine Validierungsgenauigkeit von 96,4% bei der Verwendung aller Attribute und 96,5% bei der Verwendung der radiometrischen und geometrischen Informationen erreicht werden. Die erzielten Genauigkeiten liegen im Bereich des aktuellen Forschungsstandes und über den Ergebnissen von WINIWARTER & MANDLBURGER, G. (2019) mit 95,8% und KADA & KURAMIN (2021) mit 95,5%. Allerdings wurden in diesen Arbeiten eine größere Anzahl von Klassen und andere Trainingsdaten verwendet, sodass die Genauigkeiten nicht direkt vergleichbar sind. Mit Ausnahme der Menschgemacht-Klasse werden für alle Klassen Genauigkeiten von $> 87\%$ bei der Verwendung aller Attribute und $> 90\%$ bei der Verwendung von Radiometrie und Geometrie erzielt. Für die Menschgemacht-Klasse werden mit den generierten Trainingsdaten und den gewählten Trainingsparametern nur Genauigkeiten von 38,5% bzw. 38,1% erreicht. Dies ist zurückzuführen auf die Heterogenität der Klasse, da sowohl Fahrzeuge als auch Kräne und Zäune in der Klasse enthalten sind. Zudem weist diese Klasse ein größeres Rauschen in der Klassenzuordnung auf. Ein großer Teil der Punkte der Klasse Menschgemacht wird fälschlicherweise als Boden klassifiziert. Dies liegt zum einen an der hohen geometrischen Ähnlichkeit zwischen Boden und Fahrzeugen und zum anderen an der Ungenauigkeit der Trainingsdaten, da einige Fahrzeuge als Boden klassifiziert wurden. Die visuelle Betrachtung der Klassifikation des Validierungsdatensatzes zeigt eine hohe Zuverlässigkeit der Klassifikation. Punkte, die vom Modell falsch klassifiziert wurden, sind auch mit menschlichem Auge nur schwer einer Klasse zuzuordnen.

8. Diskussion & Ausblick

Die Ergebnisse dieser Arbeit zeigen, dass die semantische Segmentierung von photogrammetrischen Punktwolken möglich ist und dass mit Hilfe der PointNet++ Architektur Ergebnisse erzielt werden können, die dem aktuellen Stand der Forschung entsprechen. Gleichzeitig kann im Rahmen dieser Arbeit

nicht untersucht werden, ob PointNet++ die optimale Architektur für diese Aufgabe ist, oder ob eine andere Deep Learning Architektur, wie z. B. KPConv, bessere Ergebnisse liefern kann. Gleichzeitig wäre der Einsatz einer komplexeren Architektur wie KPConv deutlich rechenintensiver, sodass es bei großen Datenmengen ineffizienter ist. Die PointNet++ Architektur ist im Training und in der Klassifikation deutlich effizienter und bietet den Vorteil, dass verschiedene Parameter an den spezifischen Anwendungsfall angepasst werden können. Darüber hinaus ist die Klassenzuordnung der Ground-Truth mit einem gewissen Rauschen belegt und keineswegs perfekt. Erst durch eine Verbesserung des Ground-Truth-Datensatzes wird der Einsatz einer komplexeren Architektur sinnvoll – allerdings ist auch dann nur ein Genauigkeitsgewinn im Bereich von einem Prozentpunkt zu erwarten.

Außerdem wird das trainierte Modell nur auf einem Validierungsdatensatz getestet, der in seiner Topographie dem Trainingsdatensatz sehr ähnlich ist. Es ist zu erwarten, dass das Modell schlechtere Ergebnisse liefert, wenn die Topographie von den Trainingsdaten abweicht, z. B. in Gebieten mit Bergen oder landwirtschaftlich genutzten Flächen. Außerdem ist davon auszugehen, dass das Modell stark von den Eigenschaften der Punktwolke abhängt. Punktwolken einer anderen Erhebung mit einer verschiedenen Aufnahmekonfiguration weisen eine andere Punktdichte oder unterschiedliches Messrauschen auf. Es bleibt zu untersuchen, wie das Modell auf diese unterschiedlichen Eigenschaften der Punktwolke reagiert.

Gleichzeitig bietet dieses Modell eine geeignete Grundlage für die Generierung weiterer Trainingsdaten. Durch semantische Segmentierung weiterer Punktwolken mit leicht veränderten Eigenschaften können weitere klassifizierte Punktwolken erzeugt werden. Zur Erzeugung von Ground-Truth-Daten ist bestenfalls nur eine geringfügige manuelle Optimierung erforderlich. Auf diese Weise kann die Trainingsdatenmenge erweitert werden und die Generalisierbarkeit des Modells weiter erhöht werden, sodass es auch auf andere Topographien angewendet werden kann.

Eine weitere Optimierung der Ergebnisse bieten die erhobenen Luftbilder selbst. Auch wenn durch die Ableitung der Punktwolken aus den Luftbildern geometrische Informationen gewonnen werden, gehen Kontextinformationen verloren – insbesondere, weil nicht für jedes Pixel eine dreidimensionale Koordinate berechnet werden kann. Viele Objekte sind in den Bildern wesentlich einfacher zu segmentieren, sodass zu erwarten ist, dass ein CNN in den 2D-Bildern gute Ergebnisse liefert. Da jeder Punkt der DIM-Punktwolke auf einen Bildpunkt zurückgeführt werden kann, ist es möglich, jedem Punkt eine semantische Klasse zuzuordnen. Durch eine Fusion des Modells zur semantischen Segmentierung von Punktwolken mit PointNet++ und dem CNN könnte eine weitere Optimierung der Modellperformance erreicht werden. Durch die Kombination der dreidimensionalen Informationen der Punktwolke mit den zweidimensionalen Kontextinformationen der Luftbilder kann das volle Potential der semantischen Segmentierung aus Luftbilddaten ausgeschöpft werden. Insbesondere für die Klasse der menschgemachten Objekte könnte auf diese Weise eine deutliche Performancesteigerung erreicht werden, da diese Objekte in Bildern wesentlich leichter zu erkennen sind als in einer Punktwolke.

Literaturverzeichnis

AGGARWAL, C. C. (2015): Data Mining: The Textbook. New York: Springer International Publishing.

AKAGIC, A., S. KRIVIC, H. DIZDAR & J. VELAGIC (2022): Computer Vision with 3D Point Cloud Data: Methods, Datasets and Challenges. In: XXVIII International Conference on Information, Communication and Automation Technologies (ICAT). IEEE, 1–8.

ARBEITSGEMEINSCHAFT DER VERMESSUNGSVERWALTUNGEN DER LÄNDER DER BUNDEREPUBLIK DEUTSCHLAND (2018): Dokumentation zur Modellierung der Geoinformationen des amtlichen Vrmessungswesens: GeoInfoDok.

ARTHUR, D; S. VASSILVITSKII (2007): K-Means++: The Advantages of Careful Seeding. – SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms 8, 1027–1035.

ATTA, S. (2023): Implementing K-Means++ Algorithm in Python: Step-by-Step Guide, <https://levelup.gitconnected.com/implementing-k-means-algorithm-in-python-step-by-step-guide-5b5cf77ff533> (Stand: 2023-06-11) (Zugriff: 2023-06-14).

BALLARD, D. H. (1981): Generalizing the Hough transform to detect arbitrary shapes. – Pattern Recognition 13, 2, 111–122.

BELLO, S. A., S. YU, C. WANG, J. M. ADAM & J. LI (2020): Review: Deep Learning on 3D Point Clouds. – Remote Sensing 12, 11, 1729.

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

BERG, M. de, O. CHEONG, M. VAN KREVELD, M. OVERMARS & M. T. de BERG (2008): Computational geometry: Algorithms and applications. Berlin, Heidelberg: Springer.

BLOMLEY, R., M. WEINMANN, J. LEITLOFF & B. JUTZI (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-3, 9–16.

BOULCH, A. (2019): ConvPoint: Continuous Convolutions for Point Cloud Processing, <http://arxiv.org/pdf/1904.02375v5> (Stand: 2019-04-04).

BREIMAN, L. (2001): Random Forest. – Machine Learning 45, 1, 5–32.

BREIMAN, L., J. FRIEDMAN, C. J. STONE & R. A. OLSHEN (1984): Classification and regression trees. New York: Chapman & Hall.

BUDACH, L., M. FEUERPFEIL, N. IHDE, A. NATHANSEN, N. NOACK, H. PATZLAFF, F. NAUMANN & H. HARMOUCH (2022): The Effects of Data Quality on Machine Learning Performance, <http://arxiv.org/pdf/2207.14529v4> (Stand: 2022-07-29).

CARWIN (2017): Focal Loss for Dense Object Detection in PyTorch, <https://github.com/clcarwin/focal_loss_pytorch/> (Stand: 2017-08-23) (Zugriff: 2023-08-08).

CHEHATA, N., L. GUO & C. MALLET (2009): AIRBORNE LIDAR FEATURE SELECTION FOR URBAN CLASSIFICATION USING RANDOM FORESTS. In: Laserscanning.

CHEN, M., Q. HU, Z. YU, H. THOMAS, A. FENG, Y. HOU, K. MCCULLOUGH, F. REN & L. SOIBELMAN (2022): STPLS3D: A Large-Scale Synthetic and Real Aerial Photogrammetry 3D Point Cloud Dataset, <http://arxiv.org/pdf/2203.09065v3> (Stand: 2022-03-17).

CORTES, C; V. VAPNIK (1995): Support-vector networks. – Machine Learning 20, 3, 273–297.

DEMANTKÉ, J., C. MALLET, N. DAVID & B. VALLET (2011): 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.

DOBILAS, S. (2021): DBSCAN Clustering Algorithm - How to Build Powerful Density-Based Models: A detailed guide to using Density-Based Spatial Clustering of Applications with Noise, <https://towardsdatascience.com/dbscan-clustering-algorithm-how-to-build-powerful-density-based-models-21d9961c4cec> (Stand: 2021-06-13) (Zugriff: 2023-05-23).

DÖRRE, J., P. GERSTL & R. SEIFFERT (2001): Volltextsuche und Text Mining. In: CARSTENSEN, K.-U., C. EBERT, S. JEKAT, R. KLABUNDE & H. LANGER (Hrsg.). Computerlinguistik und Sprachtechnologie: Eine Einführung. Spektrum Lehrbuch. Heidelberg: Spektrum Akademischer Verlag, 425–441.

DU, S. (2021): Deep Learning on 3D Point Clouds, <https://medium.com/mlearning-ai/deep-learning-on-3d-point-clouds-1c79d2fc3fd0> (Stand: 2021-12-27) (Zugriff: 2023-07-04).

ELDAR, Y., M. LINDENBAUM, M. PORAT & Y. Y. ZEEVI (1997): The farthest point strategy for progressive image sampling. – IEEE transactions on image processing a publication of the IEEE Signal Processing Society 6, 9, 1305–1315.

ESTER, M., H.-P. KRIEGEL, J. SANDER & X. XU (1996): A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. KDD'96. AAAI Press, 226–231.

FEI, B., W. YANG, W.-M. CHEN, Z. LI, Y. LI, T. MA, X. HU & L. MA (2022): Comprehensive Review of Deep Learning-Based 3D Point Cloud Completion Processing and Analysis. – IEEE Transactions on Intelligent Transportation Systems 23, 12, 22862–22883.

FILIN, S; N. PFEIFER (2005): Neighborhood Systems for Airborne Laser Data. – Photogrammetric Engineering & Remote Sensing 71, 6, 743–755.

FISCHLER, M. A; R. C. BOLLES (1981): Random sample consensus. – Communications of the ACM 24, 6, 381–395.

GÉRON, A. (2019[2]): Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools and techniques to build intelligent systems. Beijing, Boston, Farnham [etc.]: O'Reilly.

GONZALEZ, R. C. & R. E. WOODS (2018): Digital Image Processing. New York: Pearson.

GOODFELLOW, I., Y. BENGIO & A. COURVIELLE (2016): Deep learning. Adaptive computation and machine learning series. Cambridge, Massachusetts, London: MIT Press.

GRILLI, E., F. MENNA & F. REMONDINO (2017): A REVIEW OF POINT CLOUDS SEGMENTATION AND CLASSIFICATION ALGORITHMS. – The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XLII-2/W3, 339–344.

GUO, Y., H. WANG, Q. HU, H. LIU, L. LIU & M. BENNAMOUN (2021): Deep Learning for 3D Point Clouds: A Survey. – IEEE transactions on pattern analysis and machine intelligence 43, 12, 4338–4364 (Stand: 2021-11-03).

HACKEL, T., J. D. WEGNER & K. SCHINDLER (2016): FAST SEMANTIC SEGMENTATION OF 3D POINT CLOUDS WITH STRONGLY VARYING DENSITY. – ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences III-3, 177–184, <https://ethz.ch/content/dam/ethz/special-interest/baug/igp/photogrammetry-remote-sensing-dam/documents/pdf/timo-jan-isprs2016.pdf> (Zugriff: 2023-03-06).

HAHNER, M., D. DAI, A. LINIGER & L. VAN GOOL (2020): Quantifying Data Augmentation for LiDAR based 3D Object Detection, <https://arxiv.org/pdf/2004.01643> (Stand: 2020-04-03).

HANEKE, U., S. TRAHASCH, M. ZIMMER & C. FELDEN (2021[2]): Data Science: Grundlagen, Architekturen und Anwendungen. Heidelberg: dpunkt.verlag.

HE, Y., H. YU, X. LIU, Z. YANG, W. SUN, Y. WANG, Q. FU, Y. ZOU & A. MIAN (2021): Deep Learning based 3D Segmentation: A Survey (Stand: 2021).

HILDEBRAND, J. (2023): Generating Training Data for Deep Learning on LiDAR Point Clouds (2023-01-26). 3rd International Workshop on Point Cloud Processing. Stuttgart.

HILDEBRAND, J., S. SCHULZ, R. RICHTER & J. DÖLLNER (2022): SIMULATING LIDAR TO CREATE TRAINING DATA FOR MACHINE LEARNING ON 3D POINT CLOUDS. – ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences X-4/W2-2022, 105–112.

HIRSCHMÜLLER, H. (2005): Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information. In: IEEE (Hrsg.). Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). IEEE, 807–814.

HIRSCHMÜLLER, H. (2017): Dichte Bildzuordnung. In: HEIPKE, C. (Hrsg.). Photogrammetrie und Fernerkundung. Berlin, Heidelberg: Springer Berlin Heidelberg.

HITACHI AUTOMOTIVE AND INDUSTRY LABORATORY (2021): Semantic Segmentation Editor, <https://github.com/Hitachi-Automotive-And-Industry-Lab/semantic-segmentation-editor> (Stand: 2021-10-05) (Zugriff: 2023-06-22).

IBRAHIM, M., N. AKHTAR, M. WISE & A. MIAN (2021): Annotation Tool and Urban Dataset for 3D Point Cloud Semantic Segmentation. – IEEE Access 9, 35984–35996.

JACCARD, P. (1901): Distribution de la flore alpine dans le Bassin des Dranses et dans quelques régions voisines.

JADON, S. (2020): A survey of loss functions for semantic segmentation. In: IEEE (Hrsg.). Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB). IEEE, 1–7.

JOHNSON, A. E; M. HEBERT (1999): Using spin images for efficient object recognition in cluttered 3D scenes. – IEEE Transactions on Pattern Analysis and Machine Intelligence 21, 5, 433–449.

JUTZI, B; H. GROSS (2009): NEAREST NEIGHBOUR CLASSIFICATION ON LASER POINT CLOUDS TO GAIN OBJECT STRUCTURES FROM BUILDINGS. – International Archives of Photogrammetry and Remote Sensing, 38, <https://www.isprs.org/proceedings/XXXVIII/1_4_7-W5/paper/Jutzi-199.pdf> (Zugriff: 2023-03-12).

KADA, M; D. KURAMIN (2021): ALS POINT CLOUD CLASSIFICATION USING POINTNET++ AND KPCONV WITH PRIOR KNOWLEDGE. – The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XLVI-4/W4-2021, 91–96.

LANDRIEU, L. & M. SIMONOVSKY (2017): Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs, <https://arxiv.org/pdf/1711.09869> (Stand: 2017-11-27).

LAWSON, C. L. (1972): Transforming triangulations. – Discrete Mathematics 3, 4, 365–372.

LECUN, Y., Y. BENGIO & G. HINTON (2015): Deep learning. – Nature 521, 7553, 436–444.

LEE, I; A. SCHENK (2002): Perceptual organization of 3D surface points. – International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 34.

LESKOVEC, J., A. RAJARAMAN & J. D. ULLMAN (2014[2]): Mining of massive datasets. Cambridge: Cambridge University Press.

LI, Y., R. BU, M. SUN, W. WU, DI XINHAN & B. CHEN (2018): PointCNN: Convolution On X-Transformed Points. – Advances in Neural Information Processing Systems 31, 820–830, <http://arxiv.org/pdf/1801.07791v5>.

LIN, T.-Y., P. GOYAL, R. GIRSHICK, K. HE & P. DOLLÁR (2017): Focal Loss for Dense Object Detection, <https://arxiv.org/pdf/1708.02002> (Stand: 2017-08-07).

LINSEN, L; H. PRAUTZSCH (2001): Local Versus Global Triangulations. – Eurographics Association, <https://diglib.eg.org/handle/10.2312/egs20011021> (Zugriff: 2023-03-08).

LOSHCHILOV, I. & F. HUTTER (2017): Decoupled Weight Decay Regularization, <https://arxiv.org/pdf/1711.05101> (Stand: 2017-11-14).

LUHMANN, T. (2018[4]): Nahbereichsphotogrammetrie: Grundlagen – Methoden – Beispiele. Berlin, Offenbach: Wichmann.

MACQUEEN, J. (1967): Some Methods for Classification and analysis of multivariate observations. In: LE CAM, L. M. & J. NEYMAN (Hrsg.). Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability. University of California Press, 281–297.

MALLET, C., F. BRETAR, M. ROUX, U. SOERGEL & C. HEIPKE (2011): Relevance assessment of full-waveform lidar data for urban area classification. – ISPRS Journal of Photogrammetry and Remote Sensing 66, 6, S71-S84, <https://www.sciencedirect.com/science/article/pii/S0924271611001055> (Zugriff: 2023-03-07).

MANDLBURGER, G., K. WENZEL, A. SPITZER, N. HAALA, P. GLIRA & N. PFEIFER (2017): IMPROVED TOPOGRAPHIC MODELS VIA CONCURRENT AIRBORNE LIDAR AND DENSE IMAGE MATCHING. – ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences IV-2/W4, 259–266.

MATURANA, D; S. SCHERER (2015): VoxNet: A 3D Convolutional Neural Network for real-time object recognition. In: IEEE/RSJ (Hrsg.). International Conference on Intelligent Robots and Systems (IROS). IEEE, 922–928.

MUMUNI, A; F. MUMUNI (2022): Data augmentation: A comprehensive survey of modern approaches. – Array 16, <https://www.sciencedirect.com/science/article/pii/S2590005622000911> (Stand: 2022-04-16).

NFRAMES (o.J.): Las format definition, <https://docs.nframes.com/input-%2526-output/output-formats/las-format-definition/> (Zugriff: 2023-08-21).

NIEMEYER, J., F. ROTTENSTEINER & U. SOERGEL (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, <https://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/I-3/263/2012/isprsannals-I-3-263-2012.pdf> (Zugriff: 2023-03-08).

OSADA, R., T. FUNKHOUSER, B. CHAZELLE & D. DOBKIN (2002): Shape distributions. – ACM Transactions on Graphics 21, 4, 807–832, <http://graphics.stanford.edu/courses/cs468-08-fall/pdf/osada.pdf> (Zugriff: 2023-03-07).

PAULY, M., R. KEISER & M. GROSS (2003): Multi-scale Feature Extraction on Point-Sampled Surfaces. – Computer Graphics Forum 22, 3, 281–289.

POUX, F. (2022): 3D Point Cloud Clustering Tutorial with K-means and Python, <https://towardsdatascience.com/3d-point-cloud-clustering-tutorial-with-k-means-and-python-c870089f3af8> (Stand: 2022-04-20) (Zugriff: 2023-05-04).

QI, C. R., O. LITANY, K. HE & L. GUIBAS (2019): Deep Hough Voting for 3D Object Detection in Point Clouds. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). IEEE, 9276–9285.

QI, C. R., H. SU, K. MO & L. J. GUIBAS (2016): PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, <http://arxiv.org/pdf/1612.00593v2> (Stand: 2016-12-02).

QI, C. R., L. YI, H. SU & L. J. GUIBAS (2017): PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space, <http://arxiv.org/pdf/1706.02413v1> (Stand: 2017-06-08).

QIAN, G., Y. LI, H. PENG, J. MAI, HAMMOUD, HASAN ABED AL KADER, M. ELHOSEINY & B. GHANEM (2022): PointNeXt: Revisiting PointNet++ with Improved Training and Scaling Strategies, <https://arxiv.org/pdf/2206.04670> (Stand: 2022-06-09).

QUINLAN, J. R. (1986): Induction of decision trees. – Machine Learning 1, 1, 81–106, <https://link.springer.com/article/10.1007/BF00116251>.

RUMELHART, D. E., G. E. HINTON & R. J. WILLIAMS (1986): Learning representations by back-propagating errors. – Nature 323, 6088, 533–536.

RUSU, R. B., N. BLODOW & M. BEETZ (2009): Fast Point Feature Histograms (FPFH) for 3D registration. In: IEEE (Hrsg.). International Conference on Robotics and Automation, 3212–3217.

SHAPOVALOV, R., A. VELIZHEV & O. BARINOVA (2010): Non-associative Markov networks for 3D point cloud classification. – Photogrammetric Computer Vision and Image Analysis 38, 103–108, <https://www.isprs.org/proceedings/xxxviii/part3/a/pdf/103_XXXVIII-part3A.pdf> (Zugriff: 2023-03-07).

SINGH, S. (2021): K Means Clustering on High Dimensional Data, <https://medium.com/swlh/k-means-clustering-on-high-dimensional-data-d2151e1a4240> (Stand: 2021-01-28) (Zugriff: 2023-06-29).

SRIVASTAVA, N., G. HINTON, A. KRIZHEVSKY, I. SUTSKEVER & R. SALAKHUTDINOV (2014): Dropout: A Simple Way to Prevent Neural Networks from Overfitting. – Journal of Machine Learning Research 15, 1929–1958.

SU, H., S. MAJI, E. KALOGERAKIS & E. LEARNED-MILLER (2015): Multi-view Convolutional Neural Networks for 3D Shape Recognition, <https://arxiv.org/pdf/1505.00880> (Stand: 2015-05-05).

SUPERVISE.LY (2019): Releasing first online 3D Point Cloud labeling tool in Supervisely, <https://medium.com/deep-systems/releasing-first-online-3d-point-cloud-labeling-tool-in-supervisely-4faca42b5d6e> (Stand: 2019-07-14) (Zugriff: 2023-06-22).

SZELISKI, R. (2022[2]): Computer Vision: Algorithms and Applications. Texts in computer science. Cham: Springer International Publishing; Springer.

THOMAS, H. (2019): Learning new representations for 3D point cloud semantic segmentation. Paris: Université Paris sciences et lettres.

THOMAS, H., C. R. QI, J.-E. DESCHAUD, B. MARCOTEGUI, F. GOULETTE & L. J. GUIBAS (2019): KPConv: Flexible and Deformable Convolution for Point Clouds, <https://arxiv.org/pdf/1904.08889> (Stand: 2019-04-18).

VANDERPLAS, J. (2018[1]): Data Science mit Python: Das Handbuch für den Einsatz von IPython, Jupyter, NumPy, Pandas, Matplotlib, Scikit-Learn. Frechen: MITP.

WANG, W., R. YU, Q. HUANG & U. NEUMANN (2018): SGPN: Similarity Group Proposal Network for 3D Point Cloud Instance Segmentation. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2018): Salt Lake City, Utah, USA, 18-22 June 2018. Piscataway, NJ: IEEE, 2569–2578.

WANG, Y., Y. SUN, Z. LIU, S. E. SARMA, M. M. BRONSTEIN & J. M. SOLOMON (2018): Dynamic Graph CNN for Learning on Point Clouds, <https://arxiv.org/pdf/1801.07829> (Stand: 2018-01-24).

WEI, S. (2023): K-means Clustering and Visualization with a Real-world Dataset, <https://12ft.io/proxy?&q=https%3A%2F%2Fmedium.com%2F%40shouke.wei%2Fk-means-clustering-and-visualize-the-results-with-a-real-world-data-3872487c4a4e> (Stand: 2023-03-25) (Zugriff: 2023-06-13).

WEINMANN, M., R. BLOMLEY, M. WEINMANN & B. JUTZI (2018): Investigations on the Potential of Binary and Multi-class Classification for Object Extraction from Airborne Laser Scanning Point Clouds. In: KERSTEN, T. P., E. GÜLCH, J. SCHWIEWE, T. H. KOLBE & U. STILLA (Hrsg.). Publikationen der Deutschen Gesellschaft für Photogrammetrie, Fernerkundung und Geofirmation e.V.: Photogrammetrie - Fernerkundung - Geoinformatik - Kartographie - 2018. Hamburg:

Deutsche Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation (DGPF) e.V., 408–421.

WEINMANN, M., B. JUTZI, S. HINZ & C. MALLET (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., B. JUTZI & C. MALLET (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., A. SCHMIDT, C. MALLET, S. HINZ, F. ROTTENSTEINER & B. JUTZI (2015b): CONTEXTUAL CLASSIFICATION OF POINT CLOUD DATA BY EXPLOITING INDIVIDUAL 3D NEIGBOURHOODS. – ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences II-3/W4, 271–278, <https://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/II-3-W4/271/2015/isprsannals-II-3-W4-271-2015.pdf> (Zugriff: 2023-03-07).

WENZEL, K. (2016): Dense image matching for close range photogrammetry. Stuttgart: Universität Stuttgart.

WEST, K. F., B. N. WEBB, J. R. LERSCH, S. POTHIER, J. M. TRISCARI & A. E. IVERSON (2004): Context-driven automated target detection in 3D data. In: SADJADI, F. A. (Hrsg.). Automatic Target Recognition XIV. SPIE Proceedings. SPIE, 133–143.

WIDYANINGRUM, E., Q. BAI, M. K. FAJARI & R. C. LINDENBERGH (2021): Airborne Laser Scanning Point Cloud Classification Using the DGCNN Deep Learning Method. – Remote Sensing 13, 5, 859.

WIJMANS, E. (2018): Pointnet++ Pytorch, <https://github.com/erikwijmans/Pointnet2_PyTorch> (Stand: 2021-06-30) (Zugriff: 2023-08-05).

WINIWARTER, L. (2018): Classification of 3D Point Clouds using Deep Neural Networks. Wien.

WINIWARTER, L; G. MANDLBURGER (2019): Classification of 3D Point Clouds using Deep Neural Networks. In: KERSTEN, T. P. (Hrsg.). Dreiländertragung der OVG, DGPF und SGPF: Photogrammetrie - Fernerkundung - Geoinformation - 2019 28 28, 663–674.

YAN, X. (2019): Pointnet_Pointnet2_pytorch, <https://github.com/yanx27/Pointnet_Pointnet2_pytorch> (Stand: 2021-03-27) (Zugriff: 2023-08-05).

YIU, T. (2019): Understanding Random ForestHow the Algorith Works and Why it Is So Effective, <https://towardsdatascience.com/understanding-random-forest-58381e0602d2> (Stand: 2019-06-12) (Zugriff: 2023-06-16).

ZHANG, J., X. LIN & X. NING (2013): SVM-Based Classification of Segmented Airborne LiDAR Point Clouds in Urban Areas. – Remote Sensing 5, 8, 3749–3775.

ZHANG, Q., L. T. YANG, Z. CHEN & P. LI (2018): A survey on deep learning for big data. – Information Fusion 42, 146–157.

ZHANG, W., J. QI, P. WAN, H. WANG, D. XIE, X. WANG & G. YAN (2016): An Easy-to-Use Airborne LiDAR Data Filtering Method Based on Cloth Simulation. – Remote Sensing 8, 6, <https://www.mdpi.com/2072-4292/8/6/501> (Stand: 2016-06-15) (Zugriff: 2023-06-17).

Eidesstattliche Erklärung

Erklärung gemäß § 18 (7) Allgemeiner Teil (A) der Prüfungsordnung für die Masterstudiengänge (MPO) an der Jade Hochschule Wilhelmshaven/Oldenburg/Elsfelth in der Fassung der Bekanntmachung vom 27. Juni 2017.

Hiermit erkläre ich, dass ich die vorliegende Arbeit eigenständig und ohne fremde Hilfe angefertigt habe. Textpassagen, die wörtlich oder dem Sinn nach auf Publikationen oder Vorträgen anderer Autoren beruhen, sind als solche kenntlich gemacht.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Otterndorf, den 25.08.2023

_____

Markus Hülsen

Anlagen

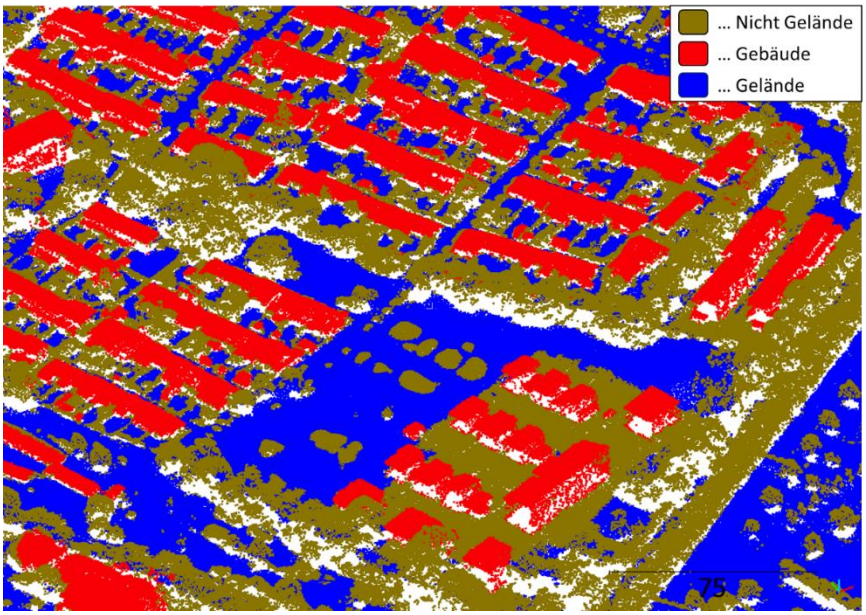
Anlage A: Jupyter-Notebooks

Alle beschriebenen Jupyter-Notebooks sind unter dem nachfolgenden Link abrufbar. Die Notebooks können als PDF, HTML oder als ausführbares Skript (**.ipynb*) heruntergeladen werden.

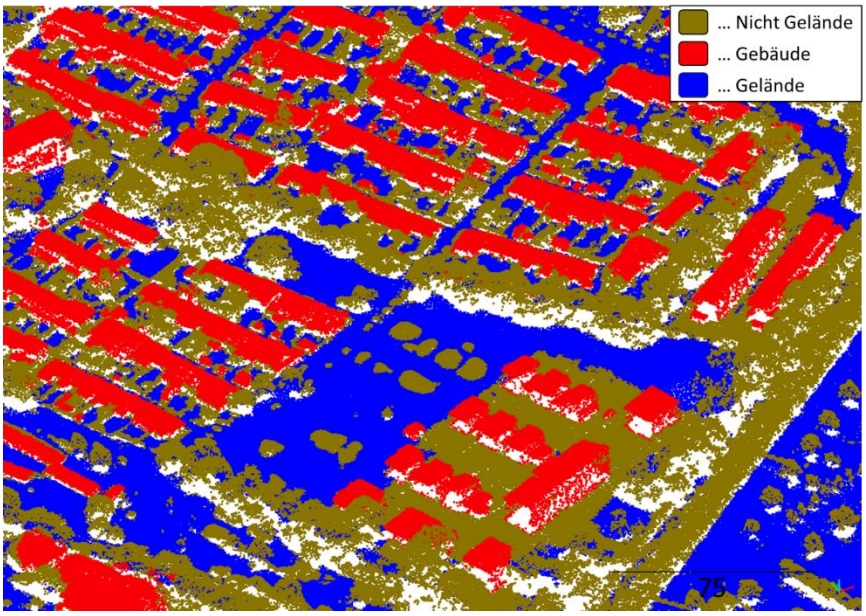
Link: https://s.gwdg.de/lwVGBk

Anlage B: Zwischenergebnisse des Workflows zur Optimierung des Trainingsdatensatzes

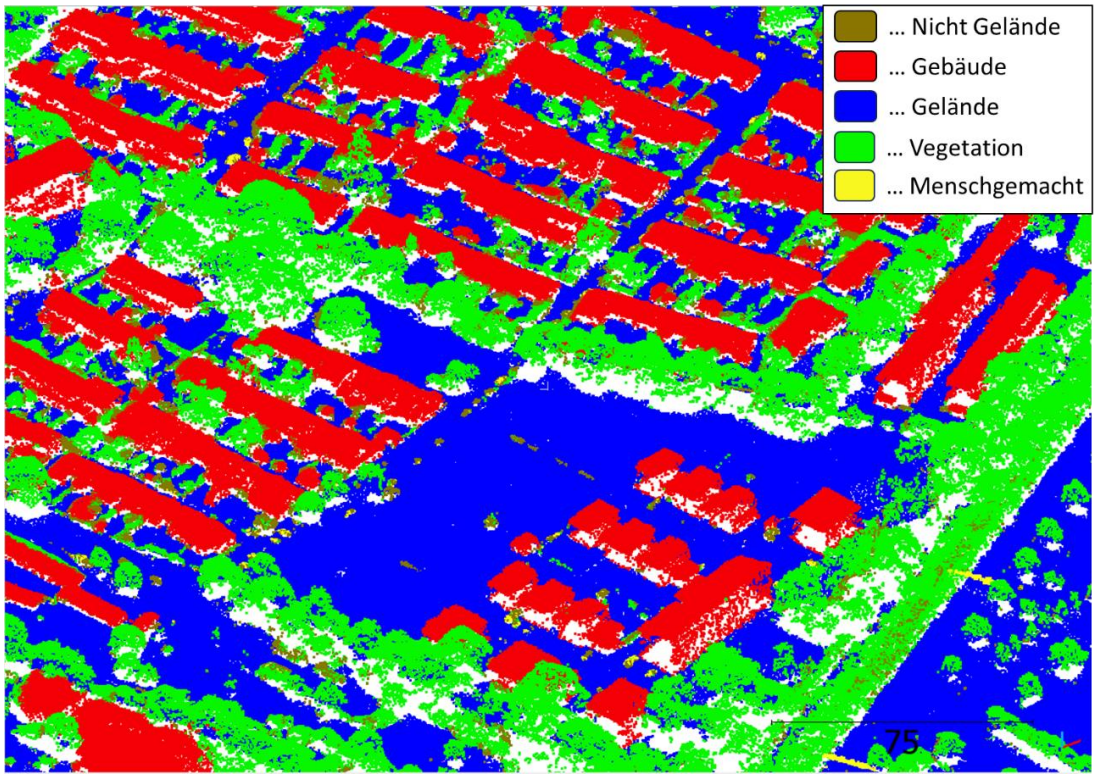
1. Ausgangslage und Ergebnis der geometrischen Klassifikation:



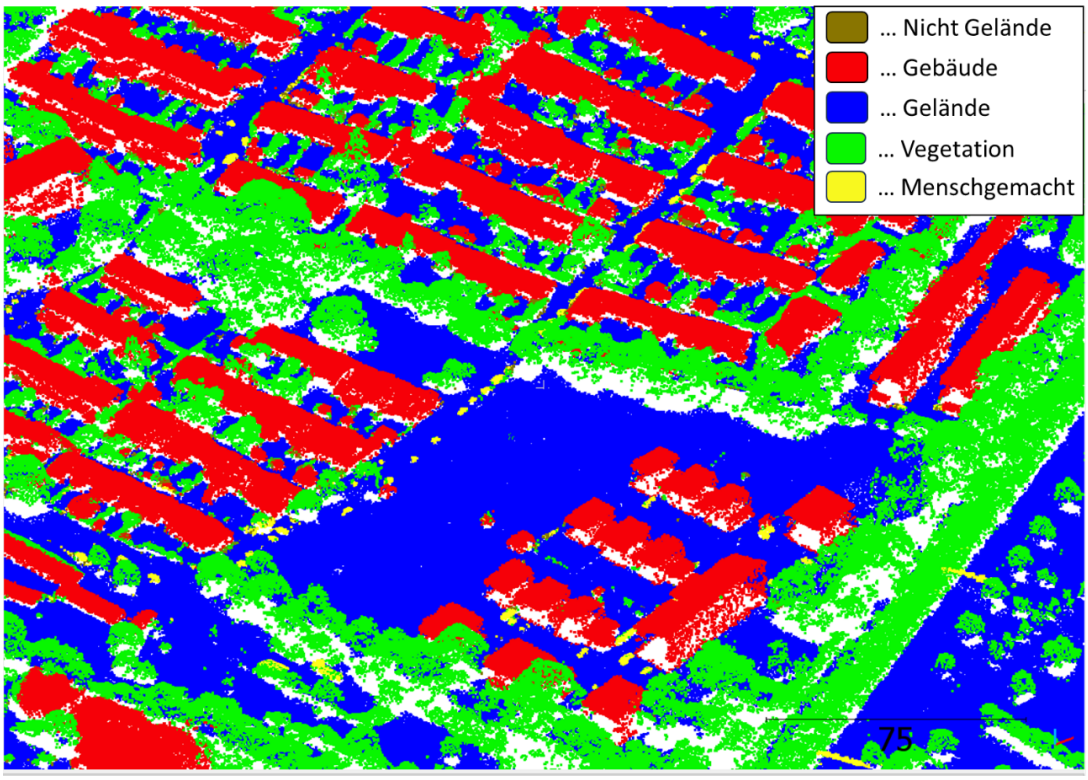
2. Ergebnisse der Optimierung der Bodenklasse nach dem CSF:



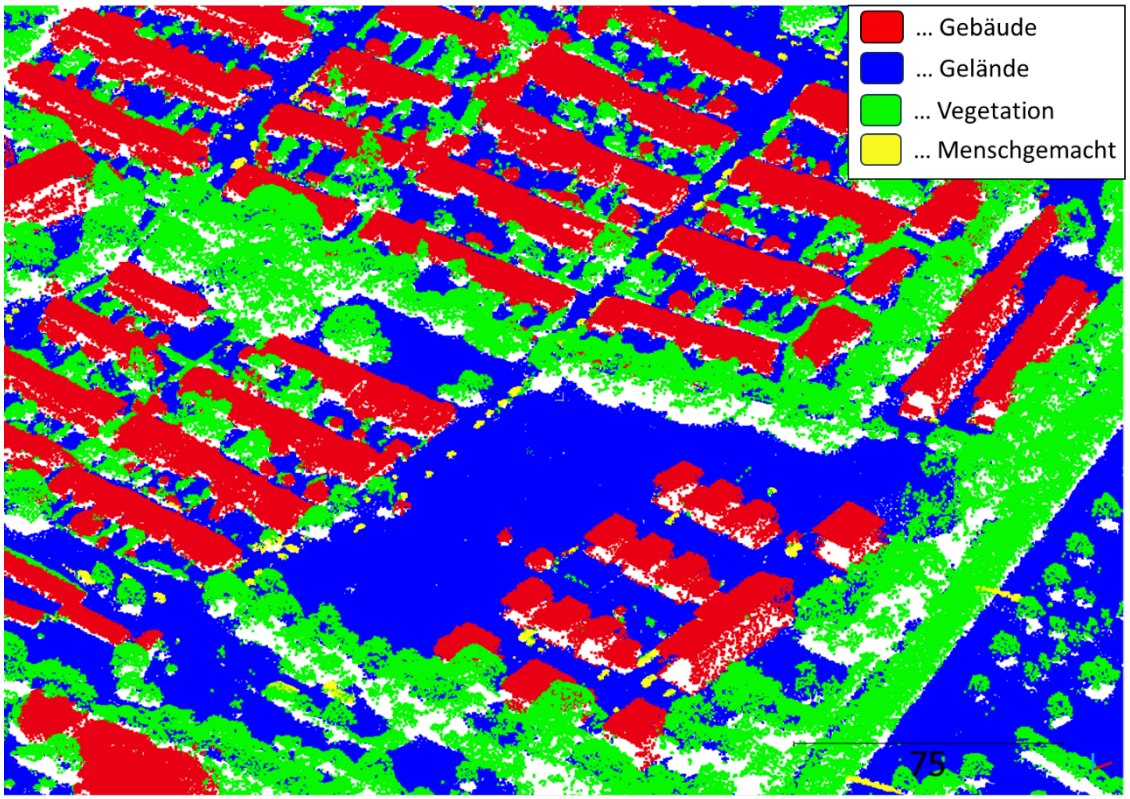
3. Ergebnisse nach dem ersten K-Means Clustering:



4. Ergebnisse nach dem zweiten K-Means-Clustering 2:



5. Ergebnisse nach der Kombination von K-Means, DBSCAN und Random Forest:



Anlage C: Annotierter Trainingsdatensatz

Der gesamte Datensatz, der wie unter 4.1.2 beschrieben annotiert wurde, kann unten den nachfolgendem Link heruntergeladen werden. Der Datensatz besteht dabei aus den zwölf $1 \times 1\ km^2$ großen Patchen und liegt im LAZ-Format vor.

Link: https://s.gwdg.de/Axda5q

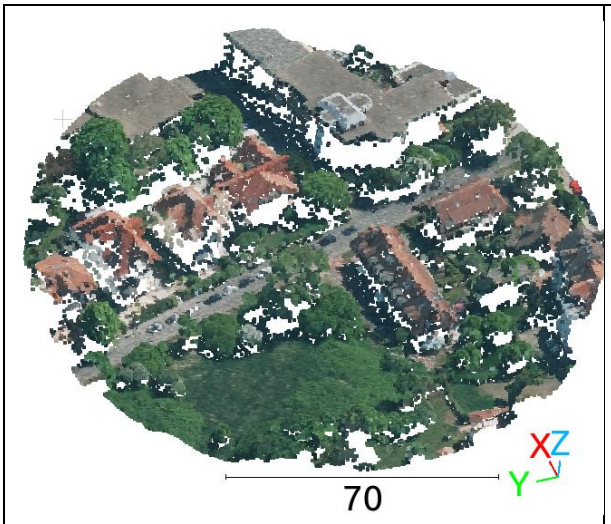
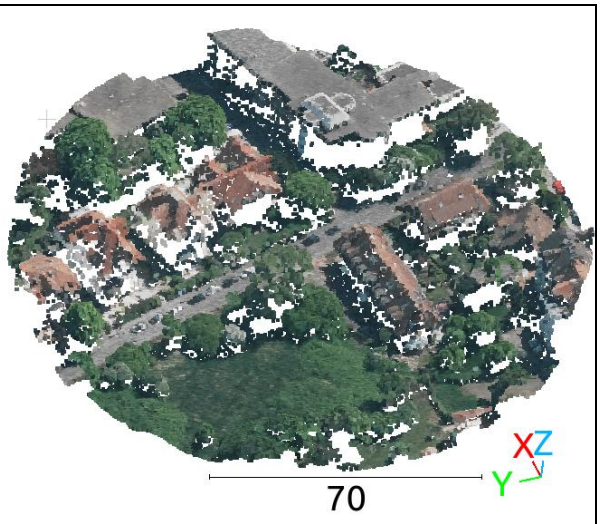
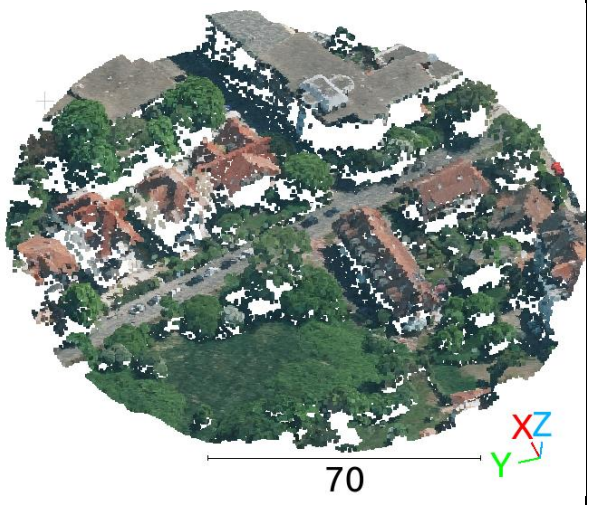
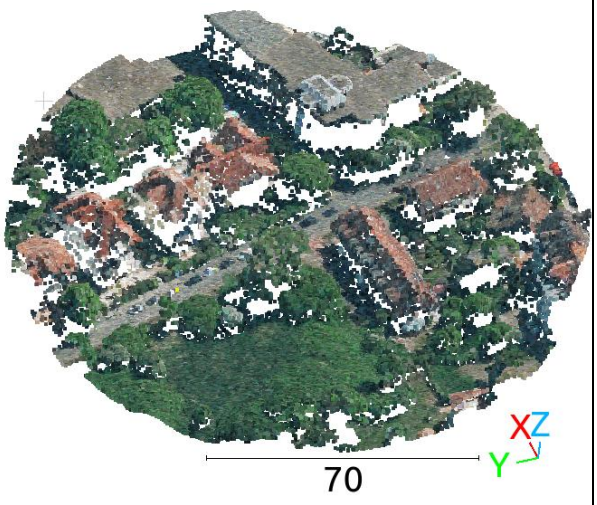
Anlage D: Python Quellcode

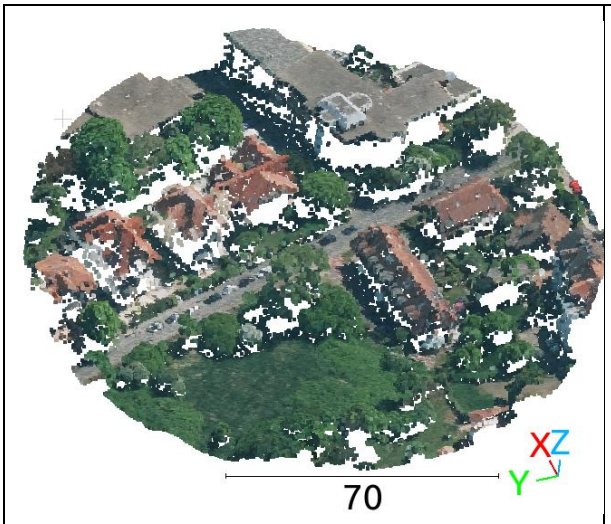
Der gesamte Python Quellcode ist als *PyCharm*-Projekt unter dem nachfolgenden Link abrufbar. Eine Verwendung des Codes ist auch ohne die *PyCharm* Umgebung möglich. Zur eigenen Verwendung ist ggf. die Installation der notwendigen Bibliotheken erforderlich.

Link: https://s.gwdg.de/twj6aD

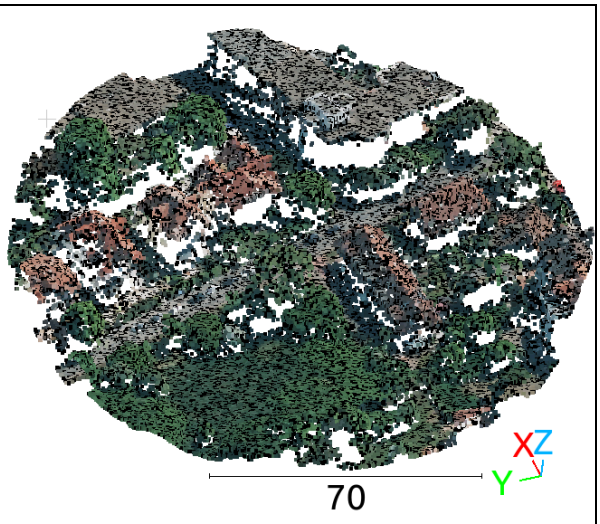
Anlage E: Beispielhafte Anwendung der Dataset Augmentation

In der nachfolgenden Tabelle werden die Dataset Augmentationen einzeln auf ein Batch ausgeführt und visuell dargestellt. Auf der linken Seite der Tabelle befindet sich immer die originale, unveränderte Punktwolke, während rechts die Auswirkungen der jeweiligen Dataset-Augmentation-Methode verdeutlicht wird. Für das HSV-Jitterung, sowie den HSV-Shift wurden die Standardabweichungen $s_{Hue} = 0{,}02$, $s_{Saturation} = 0{,}1$, $s_{Value} = 0{,}05$ verwendet.

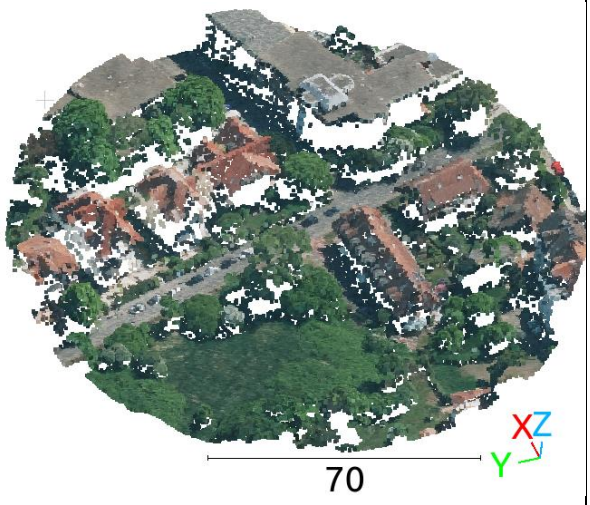
 Originale Punktwolke	 HSV-Shift
 Originale Punktwolke	 HSV-Jittering



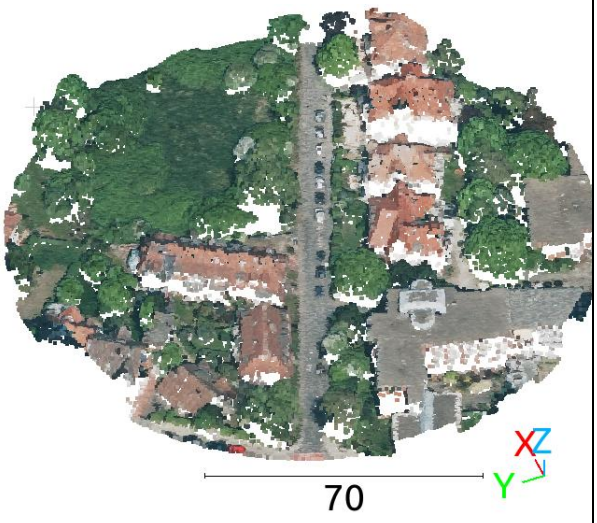
Originale Punktwolke



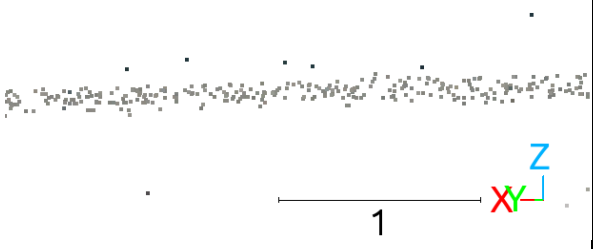
Random Colordrop (30%)



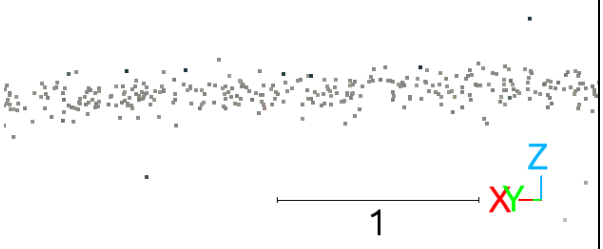
Originale Punktwolke



Random Rotation around Z



Originale Punktwolke



Z_Jittering ($s = 0.05$)

Anlage F: Trainingsprotokoll bei Variation der Verlustfunktion

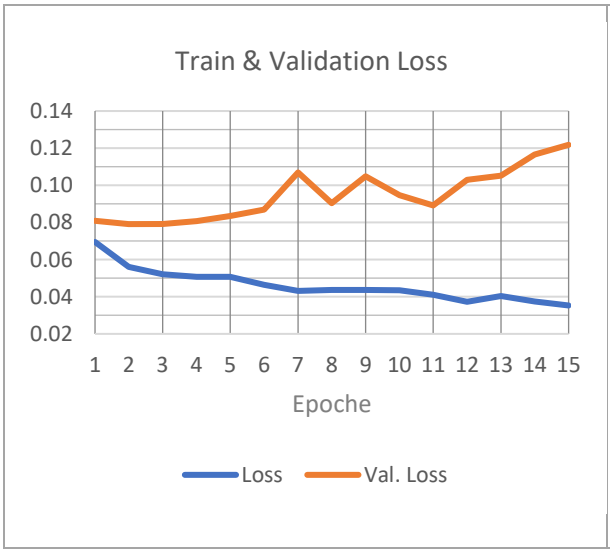
Nachfolgend sind die Resultate der Trainingsepochen tabellarisch dargestellt. Eine Epoche bezieht sich dabei auf die Verwendung aller Trainingsbatches. Maximale Werte in der Genauigkeit und der IoU

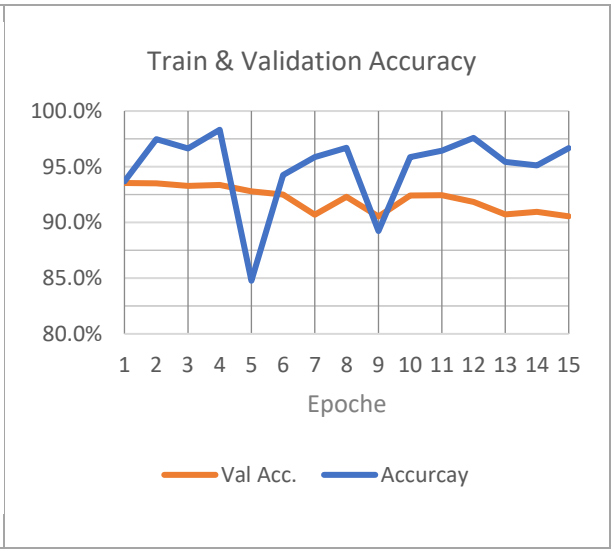
werden in Rot dargestellt, sowie minimale Werte in den Verlusten. Im Trainingsprozess wurde der Adam-Optimierer mit einer Lernrate von 0,01 verwendet. Des Weiteren werden die Trainingsprozesse graphisch verdeutlicht.

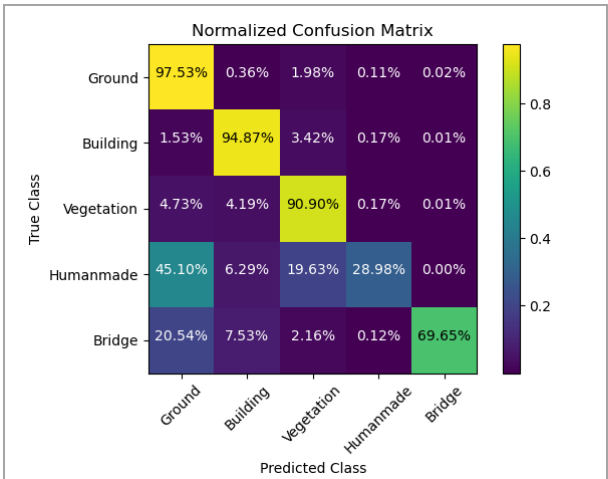
Anlage F-1: Focal Loss mit $\gamma = 2$

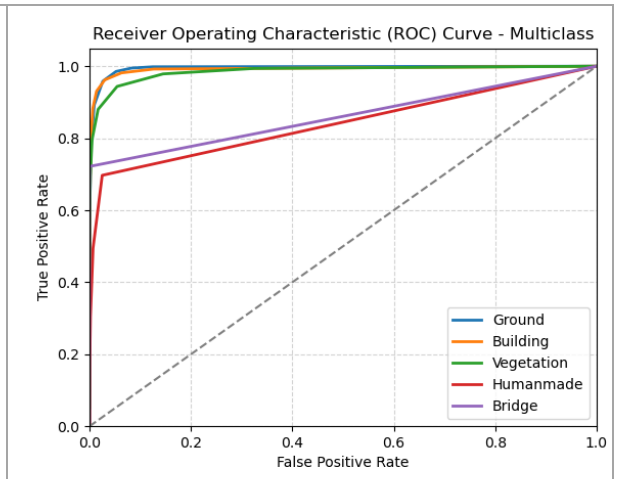
Epoche	Loss	Accurcay	Val. Loss	Val Acc.
1	0.0695	93.7%	0.0809	93.5%
2	0.0560	97.5%	0.0791	93.5%
3	0.0521	96.6%	0.0792	93.3%
4	0.0507	98.3%	0.0808	93.4%
5	0.0507	84.8%	0.0834	92.8%
6	0.0465	94.3%	0.0869	92.5%
7	0.0432	95.9%	0.1070	90.7%
8	0.0437	96.7%	0.0903	92.3%
9	0.0436	89.2%	0.1048	90.5%
10	0.0435	95.9%	0.0947	92.4%
11	0.0410	96.5%	0.0891	92.4%
12	0.0372	97.6%	0.1029	91.8%
13	0.0404	95.4%	0.1052	90.7%
14	0.0374	95.1%	0.1165	91.0%
15	0.0353	96.7%	0.1218	90.5%

Die Validierungsmetrik *Intersection-over-Union* (IoU) wurde für in diesem Prozess nicht berechnet.



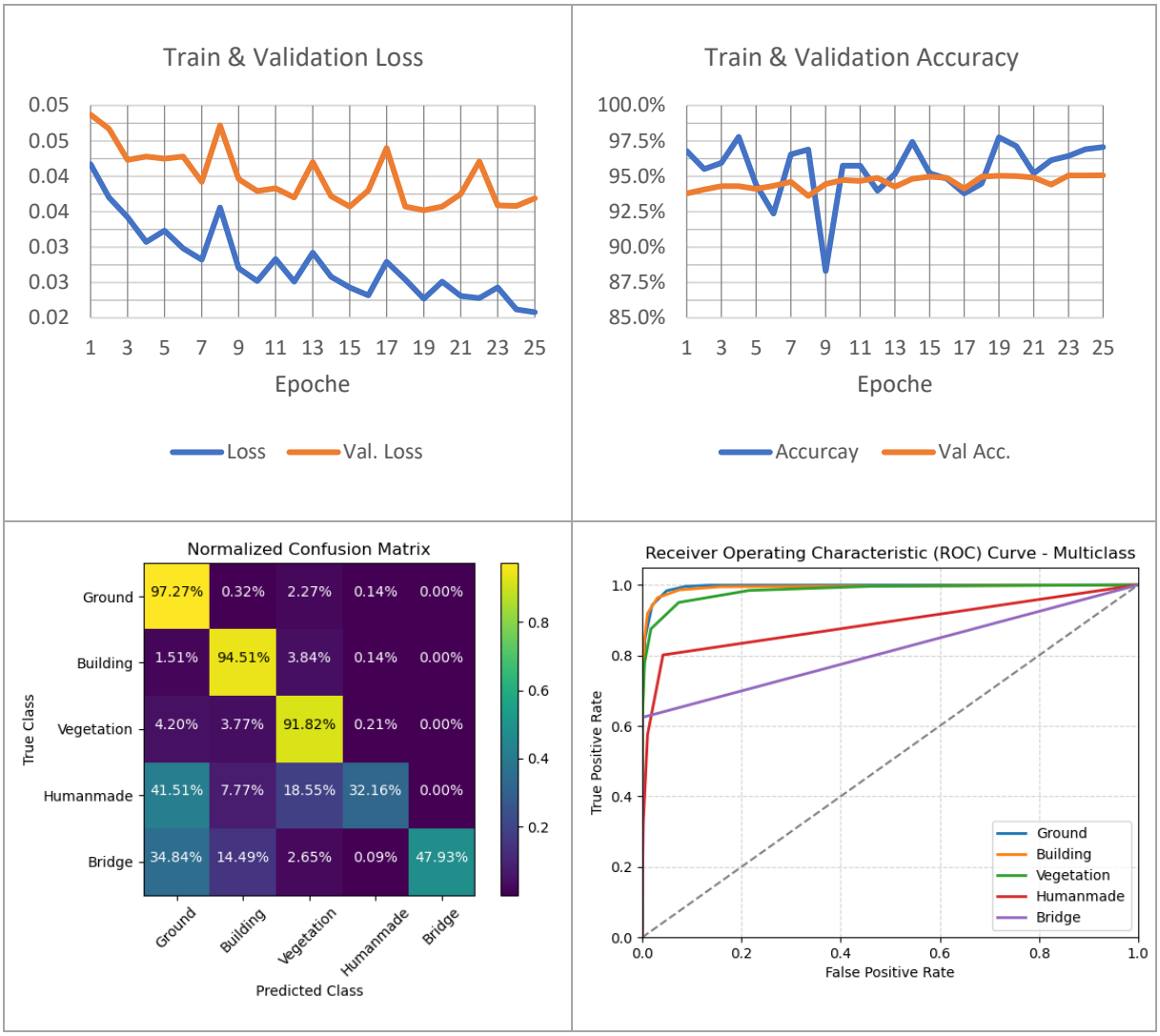






Anlage F-2: Focal Loss mit $\gamma = 3$

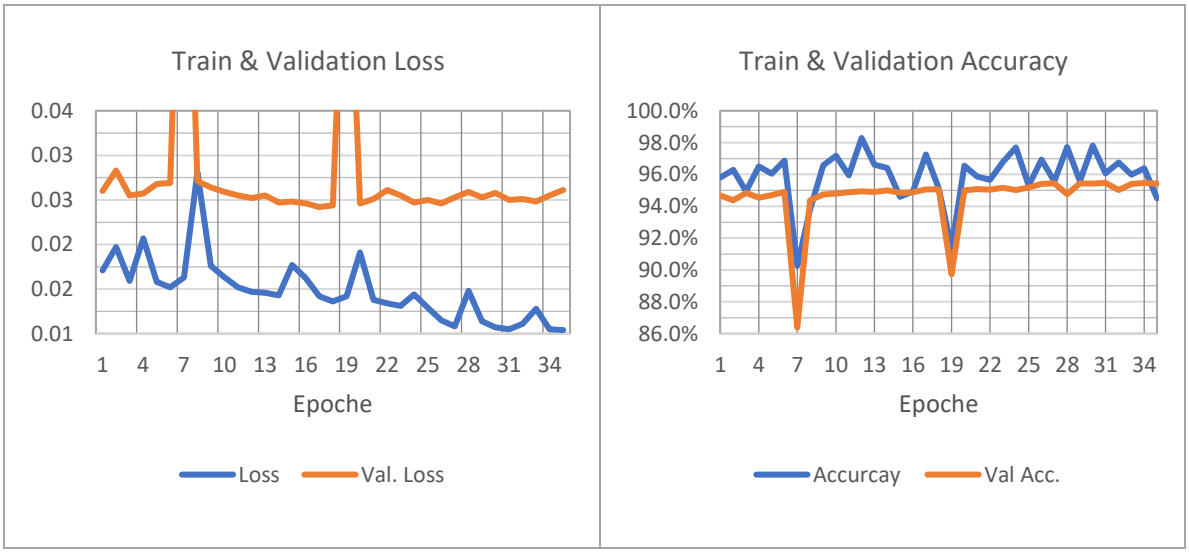
Epoche	Loss	Accurcay	Val. Loss	Val. Acc.	Val. IoU
1	0.0417	96.8%	0.0487	93.8%	0.666
2	0.037	95.5%	0.0467	94.1%	0.678
3	0.0342	96.0%	0.0423	94.3%	0.681
4	0.0307	97.8%	0.0428	94.3%	0.681
5	0.0323	94.4%	0.0425	94.1%	0.673
6	0.0298	92.4%	0.0428	94.3%	0.685
7	0.0282	96.6%	0.0392	94.6%	0.588
8	0.0356	96.9%	0.0472	93.6%	0.588
9	0.027	88.3%	0.0396	94.5%	0.649
10	0.0252	95.8%	0.0379	94.7%	0.632
11	0.0283	95.8%	0.0383	94.7%	0.634
12	0.0251	94.0%	0.0370	94.9%	0.636
13	0.0292	95.2%	0.0420	94.3%	0.614
14	0.0258	97.4%	0.0372	94.8%	0.627
15	0.0243	95.2%	0.0357	95.0%	0.621
16	0.0232	94.8%	0.038	94.9%	0.631
17	0.0279	93.8%	0.044	94.1%	0.598
18	0.0254	94.5%	0.0357	95.0%	0.635
19	0.0227	97.7%	0.0352	95.0%	0.609
20	0.0251	97.1%	0.0357	95.0%	0.638
21	0.0231	95.2%	0.0375	94.9%	0.613
22	0.0228	96.1%	0.0421	94.4%	0.588
23	0.0243	96.4%	0.0359	95.1%	0.629
24	0.0212	96.9%	0.0358	95.1%	0.628
25	0.0208	97.1%	0.0369	95.1%	0.623

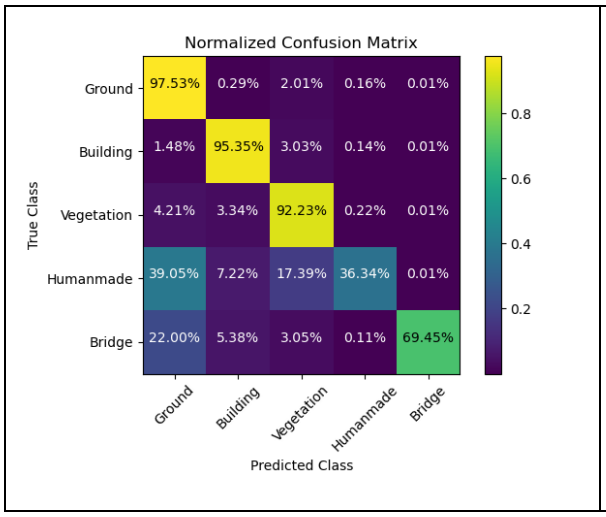


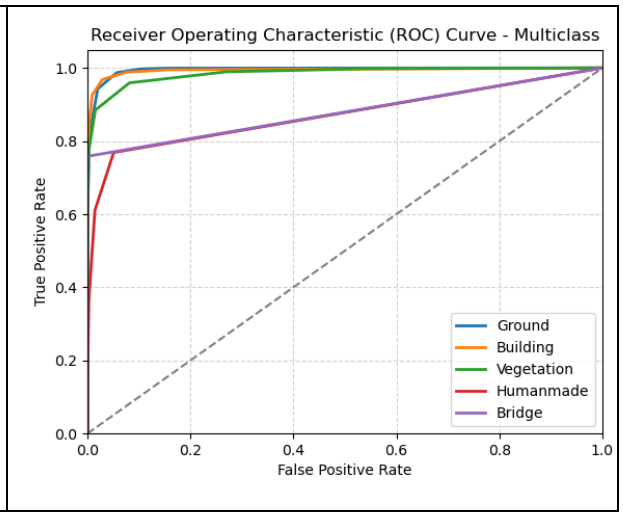
Anlage F-3: Focal Loss mit $\gamma = 4$

Epoche	Loss	Accurcay	Val. Loss	Val. Acc.	Val. IoU
1	0.0171	95.8%	0.0260	94.7%	0.622
2	0.0197	96.3%	0.0283	94.4%	0.625
3	0.0159	94.9%	0.0255	94.9%	0.634
4	0.0207	96.5%	0.0257	94.5%	0.613
5	0.0158	96.0%	0.0268	94.7%	0.631
6	0.0152	96.9%	0.0269	94.9%	0.627
7	0.0163	90.3%	0.0739	86.4%	0.448
8	0.0281	93.8%	0.0271	94.4%	0.594
9	0.0176	96.6%	0.0264	94.7%	0.623
10	0.0163	97.2%	0.0259	94.8%	0.639
11	0.0152	95.9%	0.0255	94.9%	0.611
12	0.0147	98.3%	0.0252	94.9%	0.618

13	0.0146	96.6%	0.0255	94.9%	0.617
14	0.0143	96.4%	0.0247	95.0%	0.623
15	0.0177	94.6%	0.0248	94.9%	0.601
16	0.0162	94.9%	0.0246	94.9%	0.624
17	0.0142	97.3%	0.0242	95.1%	0.622
18	0.0136	95.1%	0.0244	95.1%	0.632
19	0.0142	91.4%	0.0626	89.7%	0.485
20	0.0191	96.5%	0.0246	95.0%	0.611
21	0.0138	95.9%	0.0251	95.1%	0.623
22	0.0134	95.7%	0.0261	95.1%	0.609
23	0.0131	96.8%	0.0255	95.2%	0.636
24	0.0144	97.7%	0.0247	95.0%	0.600
25	0.0129	95.3%	0.0250	95.2%	0.626
26	0.0115	96.9%	0.0246	95.4%	0.622
27	0.0108	95.6%	0.0253	95.4%	0.645
28	0.0148	97.7%	0.0259	94.8%	0.608
29	0.0114	95.6%	0.0253	95.4%	0.647
30	0.0107	97.8%	0.0258	95.4%	0.640
31	0.0105	96.1%	0.0250	95.5%	0.646
32	0.0111	96.8%	0.0251	95.0%	0.601
33	0.0128	96.0%	0.0248	95.4%	0.638
34	0.0105	96.4%	0.0255	95.5%	0.655
35	0.0104	94.5%	0.0261	95.4%	0.632



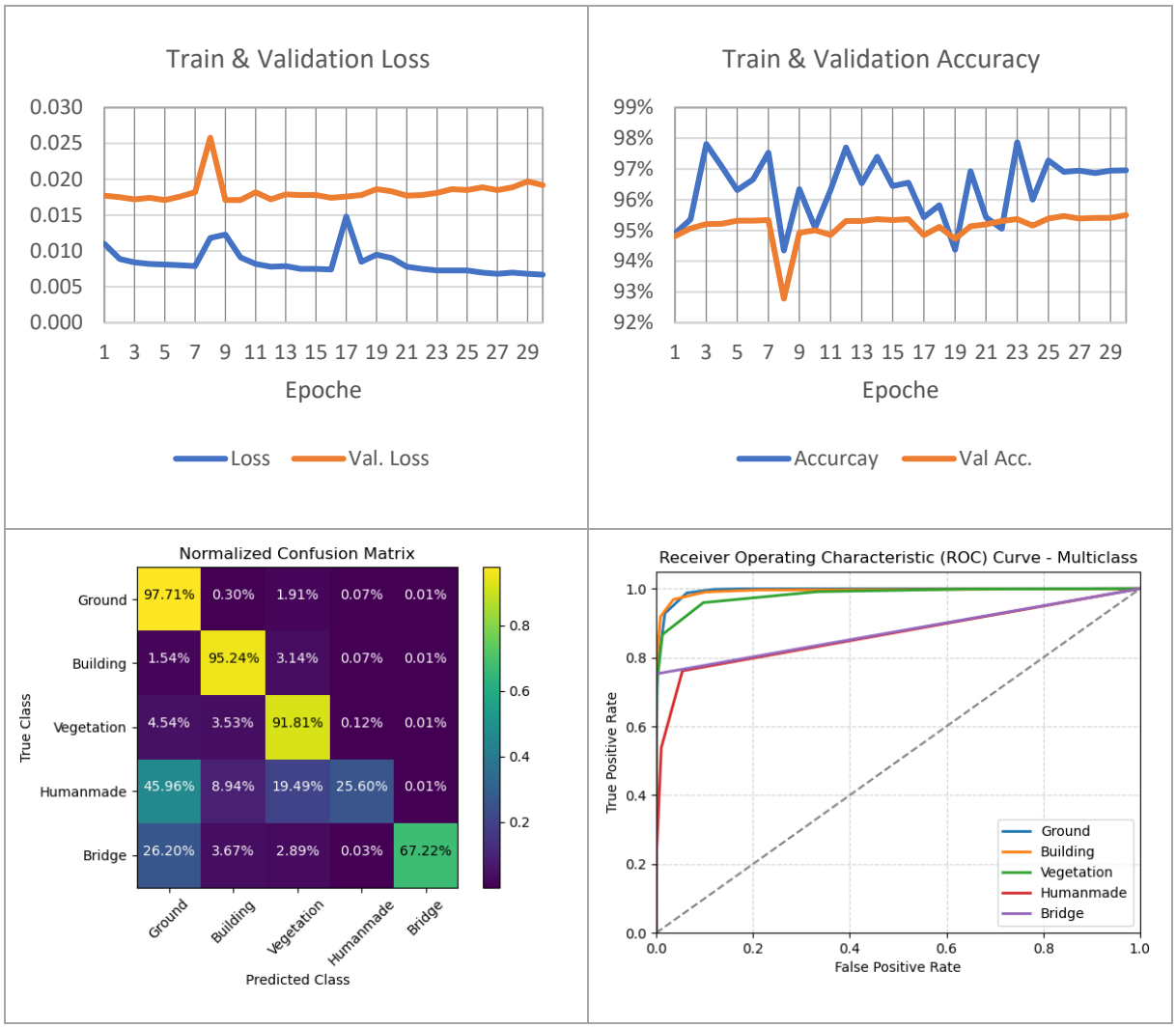




Anlage F-4: Focal Loss mit $\gamma = 5$

Epoche	Loss	Accurcay	Val. Loss	Val. Acc.	Val. IoU
1	0.0110	94.9%	0.0177	94.8%	0.630
2	0.0089	95.4%	0.0175	95.1%	0.626
3	0.0084	97.8%	0.0172	95.2%	0.627
4	0.0082	97.1%	0.0174	95.2%	0.629
5	0.0081	96.3%	0.0171	95.3%	0.623
6	0.0080	96.7%	0.0176	95.3%	0.638
7	0.0079	97.5%	0.0182	95.3%	0.629
8	0.0118	94.4%	0.0258	92.8%	0.549
9	0.0123	96.3%	0.0171	94.9%	0.609
10	0.0091	95.1%	0.0171	95.0%	0.620
11	0.0082	96.3%	0.0182	94.9%	0.622
12	0.0078	97.7%	0.0172	95.3%	0.633
13	0.0079	96.5%	0.0179	95.3%	0.648
14	0.0075	97.4%	0.0178	95.4%	0.635
15	0.0075	96.5%	0.0178	95.3%	0.646
16	0.0074	96.6%	0.0174	95.4%	0.616
17	0.0148	95.4%	0.0176	94.9%	0.602
18	0.0085	95.8%	0.0178	95.1%	0.625
19	0.0095	94.4%	0.0186	94.7%	0.603
20	0.0090	96.9%	0.0183	95.1%	0.625
21	0.0078	95.4%	0.0177	95.2%	0.623
22	0.0075	95.1%	0.0178	95.3%	0.643
23	0.0073	97.9%	0.0181	95.4%	0.632
24	0.0073	96.0%	0.0186	95.2%	0.612
25	0.0073	97.3%	0.0185	95.4%	0.631
26	0.0070	96.9%	0.0189	95.5%	0.632
27	0.0068	97.0%	0.0185	95.4%	0.638
28	0.0070	96.9%	0.0189	95.4%	0.623

29	0.0068	97.0%	0.0197	95.4%	0.639
30	0.0067	97.0%	0.0192	95.5%	0.644

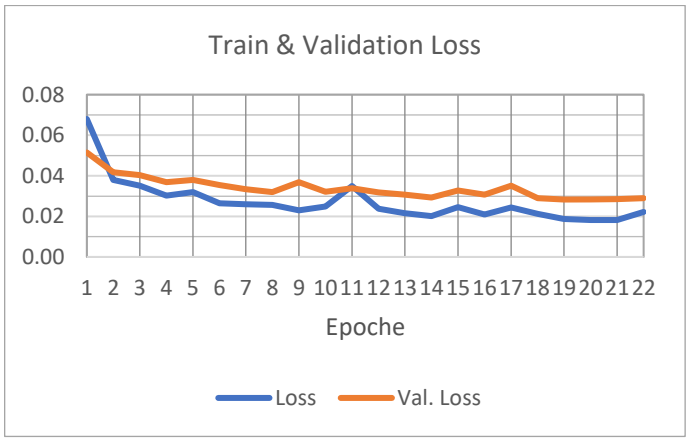


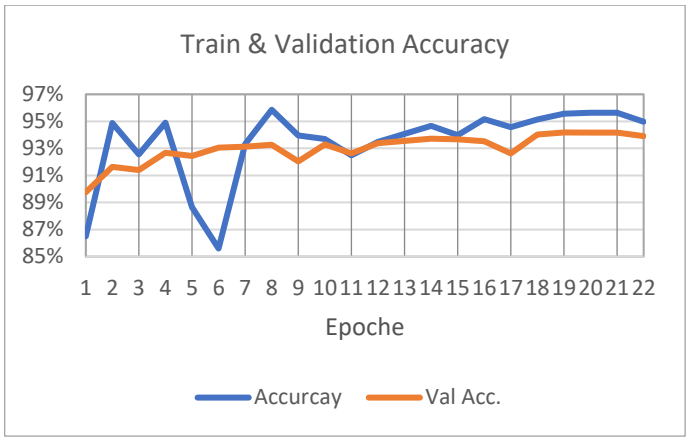
Anlage G: Training mit verschiedenen Attributteilmengen

Nachfolgend werden die Ergebnisse jeder Epoche bei Variation der verwendeten Attribute sowohl tabellarisch als auch in einem Diagramm dargestellt. Beim Training wird der Adam-Optimierer mit einer Lernrate von 0,001 und der Focal-Loss mit $\gamma = 4$ als Verlustfunktion verwendet. In Anlage E-1 werden nur die X-, Y- und Z-Koordinaten für das Training verwendet, während in Anlage E-2 zusätzlich die Farbinformationen Hue, Saturation und Value genutzt werden.

Anlage G-1: Training mit geometrischen Informationen

Epoche	Loss	Genauigkeit	Val. Loss	Val. Genauigkeit	Val. IoU
1	0.0680	86.5%	0.0515	89.8%	0.523
2	0.0379	94.9%	0.0418	91.6%	0.566
3	0.0351	92.6%	0.0403	91.4%	0.542
4	0.0303	94.9%	0.0368	92.7%	0.543
5	0.0319	88.7%	0.038	92.4%	0.546
6	0.0265	85.6%	0.0355	93.1%	0.576
7	0.026	93.3%	0.0334	93.1%	0.561
8	0.0257	95.9%	0.0319	93.3%	0.608
9	0.0229	94.0%	0.0368	92.0%	0.528
10	0.0248	93.7%	0.0321	93.3%	0.593
11	0.0349	92.5%	0.0339	92.6%	0.592
12	0.0238	93.5%	0.0318	93.4%	0.634
13	0.0215	94.1%	0.0307	93.6%	0.644
14	0.0201	94.7%	0.0293	93.7%	0.636
15	0.0246	94.0%	0.0327	93.7%	0.598
16	0.021	95.2%	0.0307	93.5%	0.617
17	0.0244	94.6%	0.0352	92.6%	0.587
18	0.0212	95.1%	0.0290	94.0%	0.637
19	0.0187	95.6%	0.0283	94.2%	0.636
20	0.0182	95.6%	0.0284	94.2%	0.649
21	0.0182	95.6%	0.0285	94.2%	0.638
22	0.0222	95.0%	0.0290	93.9%	0.588

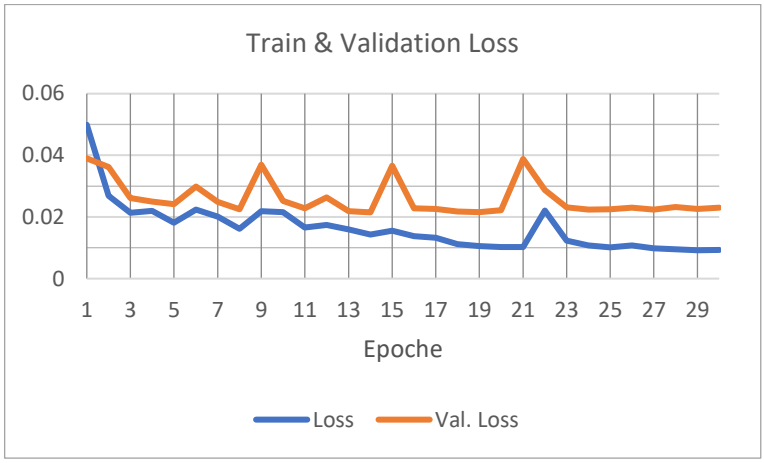


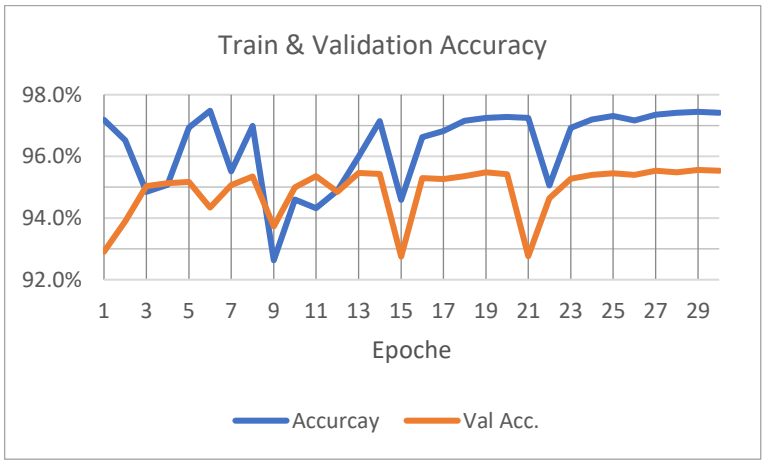


Anlage G-2: Training mit geometrischen und radiometrischen Informationen

Epoche	Loss	Genauigkeit	Val. Loss	Val. Genauigkeit	Val. IoU
1	0.0499	97.2%	0.039	92.9%	0.550
2	0.0269	96.5%	0.0362	93.9%	0.576
3	0.0214	94.9%	0.0261	95.0%	0.640
4	0.0220	95.1%	0.025	95.1%	0.625
5	0.0181	96.9%	0.0242	95.2%	0.625
6	0.0224	97.5%	0.0299	94.3%	0.635
7	0.0201	95.5%	0.0249	95.1%	0.630
8	0.0162	97.0%	0.0225	95.4%	0.650
9	0.0219	92.6%	0.0369	93.7%	0.668
10	0.0216	94.6%	0.0252	95.0%	0.599
11	0.0166	94.3%	0.0228	95.4%	0.624
12	0.0174	94.9%	0.0263	94.9%	0.590
13	0.0160	96.0%	0.0219	95.5%	0.666
14	0.0143	97.1%	0.0215	95.4%	0.692
15	0.0155	94.6%	0.0366	92.8%	0.547
16	0.0138	96.6%	0.0228	95.3%	0.612
17	0.0133	96.8%	0.0226	95.3%	0.626
18	0.0112	97.2%	0.0218	95.4%	0.635
19	0.0106	97.3%	0.0216	95.5%	0.635
20	0.0102	97.3%	0.0222	95.4%	0.678
21	0.0102	97.3%	0.0388	92.8%	0.616
22	0.0221	95.1%	0.0287	94.6%	0.604
23	0.0123	96.9%	0.0231	95.3%	0.615
24	0.0108	97.2%	0.0224	95.4%	0.626
25	0.0101	97.3%	0.0225	95.5%	0.625
26	0.0108	97.2%	0.023	95.4%	0.626
27	0.0098	97.4%	0.0224	95.5%	0.649

28	0.0095	97.4%	0.0232	95.5%	0.653
29	0.0092	97.4%	0.0226	95.6%	0.661
30	0.0093	97.4%	0.023	95.5%	0.654



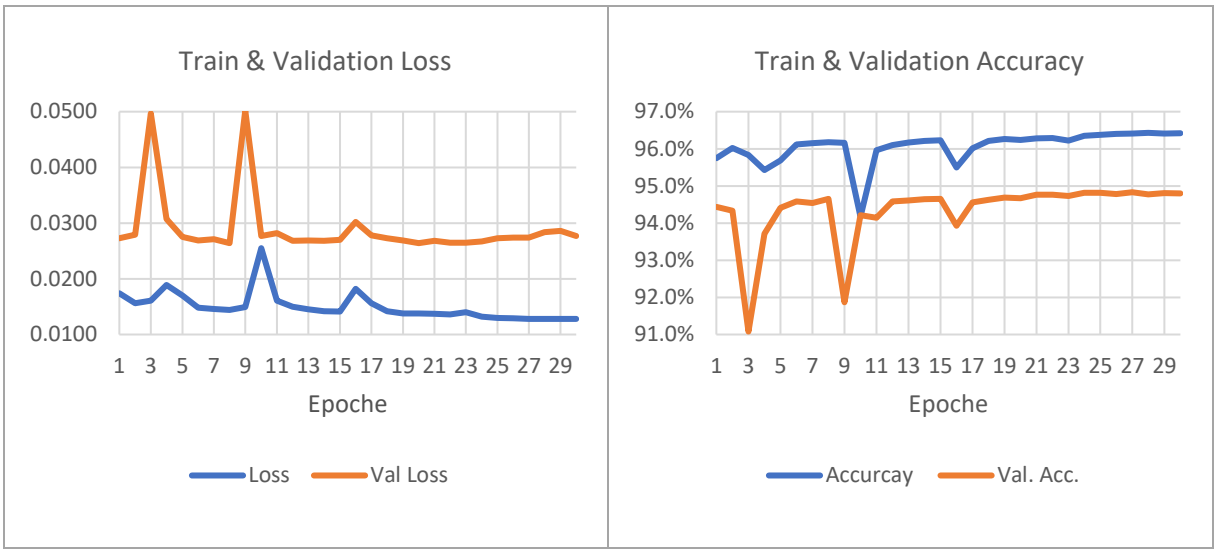


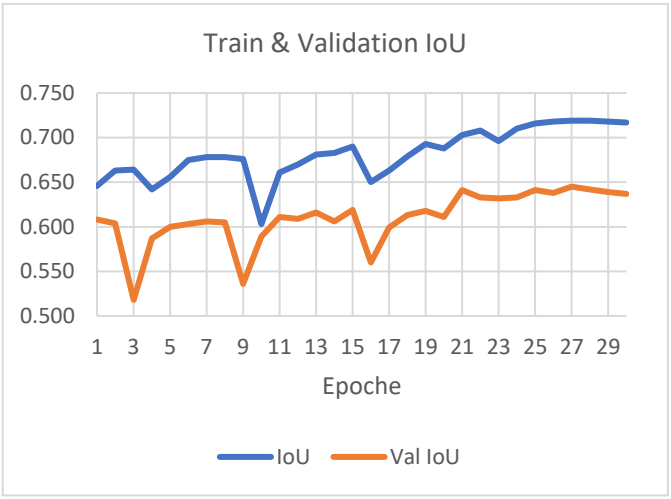
Anlage H: Trainingsprotokolle bei verschiedenen Batchgrößen

Beim Trainingsprozess werden die Batchgrößen mit $K = 50.000$ und $K = 200.000$ Epochen verwendet. Nachfolgende sind die Ergebnisse nach jeder einzelnen Trainingsepoche dargestellt. Dabei wird der Adam-Optimierer mit einer Lernrate von $0{,}001$ und der Focal-Loss mit $\gamma = 4$ als Verlustfunktion verwendet. Eine Batchgröße von $K = 100.000$ wurde in den bisherigen Experimenten näher untersucht. Die Referenz bezieht sich auf Epoche 31 in Anlage F-3: *Focal Loss mit* $\gamma = 4$. Minimale Verluste, sowie maximale Genauigkeiten werden in Rot hervorgehoben.

Anlage H-1: Batchgröße K=200.000

Epoche	Loss	Accurcay	IoU	Val. Loss	Val. Acc.	Val. IoU
Referenz	**0.0105**	**96.1%**	**-**	**0.0250**	**95.5%**	**0.646**
1	0.0174	95.8%	0.646	0.0273	94.4%	0.608
2	0.0156	96.0%	0.663	0.0279	94.3%	0.604
3	0.0161	95.8%	0.664	0.0497	91.1%	0.518
4	0.0189	95.4%	0.642	0.0307	93.7%	0.587
5	0.0170	95.7%	0.656	0.0275	94.4%	0.600
6	0.0148	96.1%	0.675	0.0269	94.6%	0.603
7	0.0146	96.2%	0.678	0.0271	94.5%	0.606
8	0.0144	96.2%	0.678	0.0264	94.7%	0.605
9	0.0149	96.2%	0.676	0.0502	91.9%	0.536
10	0.0255	94.2%	0.603	0.0277	94.2%	0.589
11	0.0161	96.0%	0.661	0.0282	94.1%	0.611
12	0.0150	96.1%	0.670	0.0268	94.6%	0.609
13	0.0145	96.2%	0.681	0.0269	94.6%	0.616
14	0.0142	96.2%	0.683	0.0268	94.6%	0.606
15	0.0141	96.2%	0.690	0.027	94.7%	0.619
16	0.0182	95.5%	0.650	0.0302	93.9%	0.560
17	0.0156	96.0%	0.663	0.0278	94.6%	0.599
18	0.0142	96.2%	0.679	0.0273	94.6%	0.613
19	0.0138	96.3%	0.693	0.0269	94.7%	0.618
20	0.0138	96.2%	0.688	0.0264	94.7%	0.611
21	0.0137	96.3%	0.703	0.0268	94.8%	0.641
22	0.0136	96.3%	0.708	0.0265	94.8%	0.633
23	0.0140	96.2%	0.696	0.0265	94.7%	0.632
24	0.0132	96.4%	0.71	0.0267	94.8%	0.633
25	0.0130	96.4%	0.716	0.0273	94.8%	0.641
26	0.0129	96.4%	0.718	0.0274	94.8%	0.638
27	0.0128	96.4%	0.719	0.0274	94.8%	0.645
28	0.0128	96.4%	0.719	0.0284	94.8%	0.642
29	0.0128	96.4%	0.718	0.0286	94.8%	0.639
30	0.0128	96.4%	0.717	0.0277	94.8%	0.637

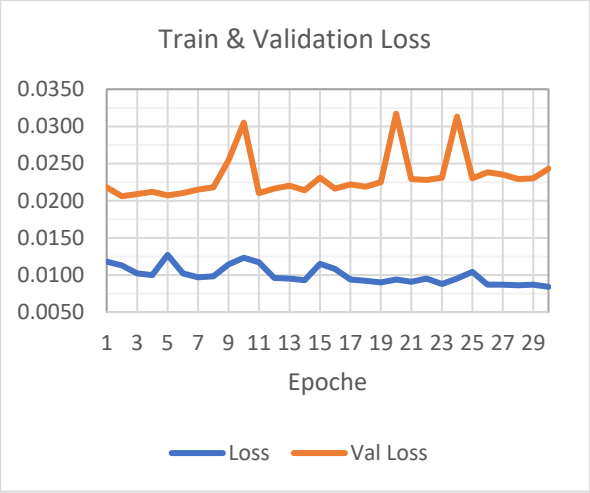


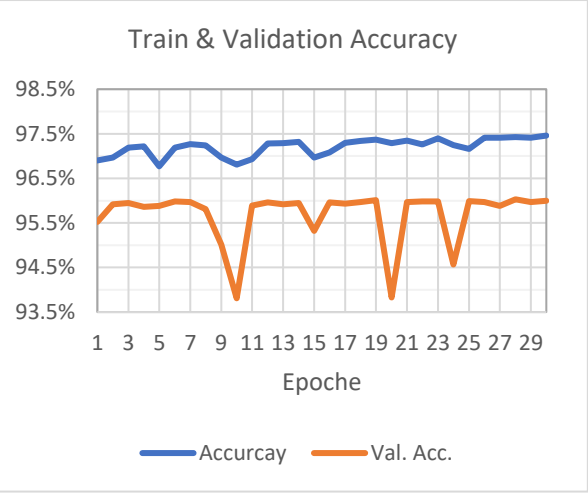


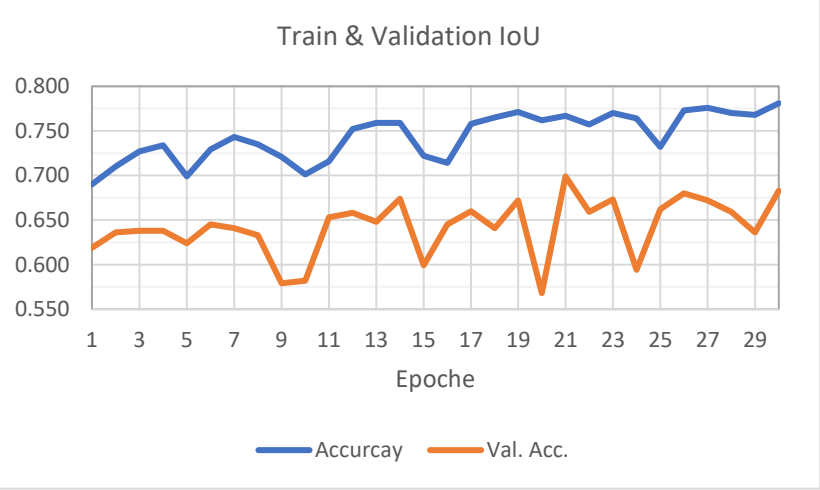
Anlage H-2: Batchgröße K=50.000

Epoche	Loss	Accurcay	IoU	Val. Loss	Val. Acc.	Val. IoU
Referenz	**0.0105**	**96.1%**	**-**	**0.025**	**95.5%**	**0.646**
1	0.0118	96.9%	0.690	0.0218	95.5%	0.619
2	0.0113	97.0%	0.710	0.0206	95.9%	0.636
3	0.0102	97.2%	0.727	0.0209	96.0%	0.638
4	0.0100	97.2%	0.734	0.0212	95.9%	0.638
5	0.0127	96.8%	0.699	0.0207	95.9%	0.624
6	0.0102	97.2%	0.729	0.0210	96.0%	0.645
7	0.0097	97.3%	0.743	0.0215	96.0%	0.641
8	0.0098	97.2%	0.735	0.0218	95.8%	0.633
9	0.0114	97.0%	0.721	0.0254	95.0%	0.579
10	0.0123	96.8%	0.701	0.0305	93.8%	0.582
11	0.0117	96.9%	0.716	0.0210	95.9%	0.653
12	0.0096	97.3%	0.752	0.0216	96.0%	0.658
13	0.0095	97.3%	0.759	0.0220	95.9%	0.648

14	0.0093	97.3%	0.759	0.0214	96.0%	0.674
15	0.0115	97.0%	0.722	0.0231	95.3%	0.599
16	0.0108	97.1%	0.714	0.0216	96.0%	0.645
17	0.0094	97.3%	0.758	0.0222	95.9%	0.660
18	0.0092	97.3%	0.765	0.0219	96.0%	0.641
19	0.0090	97.4%	0.771	0.0225	96.0%	0.672
20	0.0094	97.3%	0.762	0.0317	93.8%	0.568
21	0.0091	97.4%	0.767	0.0229	96.0%	0.699
22	0.0095	97.3%	0.757	0.0228	96.0%	0.659
23	0.0088	97.4%	0.770	0.0231	96.0%	0.673
24	0.0095	97.3%	0.764	0.0313	94.6%	0.594
25	0.0104	97.2%	0.732	0.0230	96.0%	0.662
26	0.0087	97.4%	0.773	0.0238	96.0%	0.680
27	0.0087	97.4%	0.776	0.0235	95.9%	0.672
28	0.0086	97.4%	0.770	0.0229	96.0%	0.659
29	0.0087	97.4%	0.768	0.0230	96.0%	0.636
30	0.0084	97.5%	0.781	0.0243	96.0%	0.683



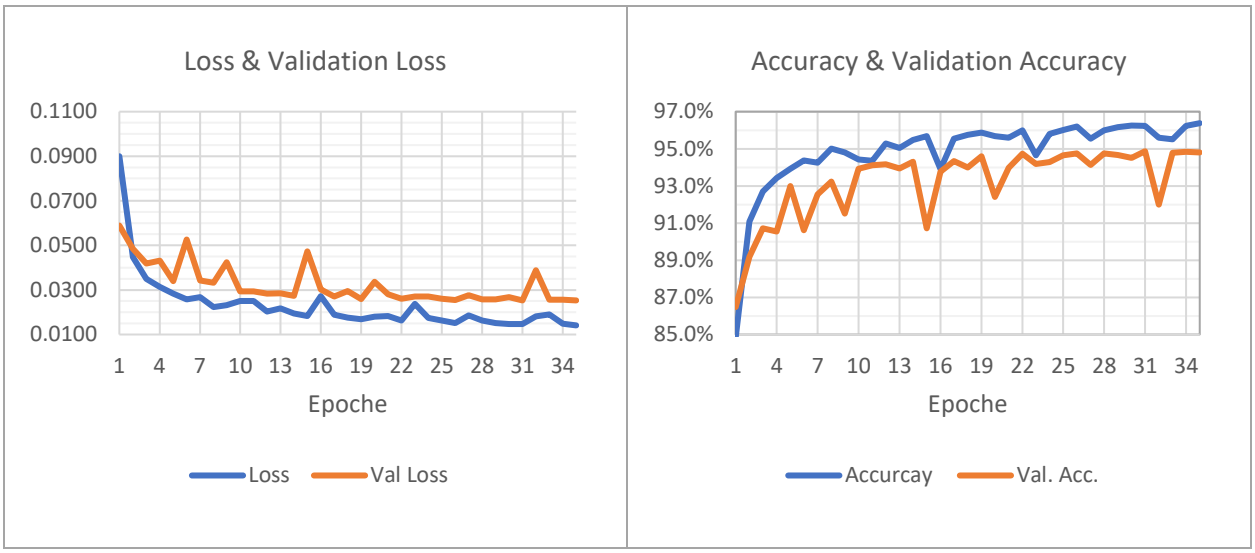


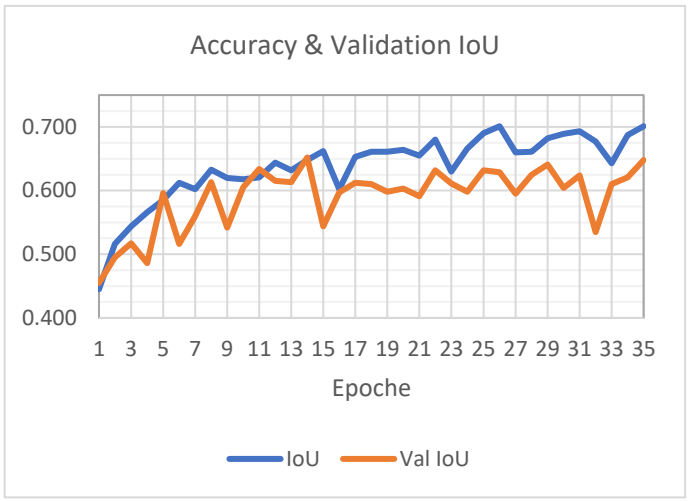


Anlage H: Trainingsprotokolle bei Variation der Netzarchitektur

In der Grundimplementierung wurden drei Set-Abstraction-Layer verwendet. Nachfolgende Ergebnisse beziehen sich auf Ergebnissen mit einem weiteren Set-Abstraction Layer. Dabei wird der Adam-Optimierer mit einer Lernrate von 0,001 und der Focal Loss $\gamma = 4$ verwendet.

Epoche	Loss	Accurcay	IoU	Val. Loss	Val. Acc.	Val. IoU
1	0.0900	84.8%	0.445	0.0589	86.5%	0.455
2	0.0447	91.1%	0.516	0.0484	89.2%	0.495
3	0.0349	92.7%	0.544	0.0419	90.7%	0.517
4	0.0313	93.4%	0.566	0.0431	90.6%	0.486
5	0.0283	93.9%	0.585	0.0339	93.0%	0.596
6	0.0257	94.4%	0.612	0.0526	90.6%	0.516
7	0.0268	94.3%	0.602	0.0342	92.6%	0.560
8	0.0223	95.0%	0.633	0.0333	93.2%	0.613
9	0.0232	94.8%	0.620	0.0425	91.5%	0.542
10	0.0250	94.4%	0.618	0.0293	93.9%	0.605
11	0.0250	94.4%	0.621	0.0294	94.1%	0.634
12	0.0203	95.3%	0.644	0.0284	94.2%	0.615
13	0.0217	95.1%	0.632	0.0285	93.9%	0.613
14	0.0194	95.5%	0.648	0.0273	94.3%	0.652
15	0.0183	95.7%	0.662	0.0473	90.7%	0.544
16	0.0272	93.9%	0.602	0.0302	93.8%	0.597
17	0.0188	95.5%	0.653	0.0270	94.3%	0.612
18	0.0176	95.8%	0.661	0.0295	94.0%	0.610
19	0.0169	95.9%	0.661	0.0259	94.6%	0.598
20	0.0180	95.7%	0.664	0.0337	92.4%	0.603
21	0.0183	95.6%	0.655	0.0281	94.0%	0.591
22	0.0163	96.0%	0.680	0.0260	94.7%	0.632
23	0.0237	94.6%	0.630	0.0271	94.2%	0.611
24	0.0174	95.8%	0.666	0.0270	94.3%	0.598
25	0.0163	96.0%	0.690	0.0261	94.7%	0.632
26	0.0152	96.2%	0.701	0.0255	94.8%	0.629
27	0.0186	95.6%	0.660	0.0276	94.1%	0.595
28	0.0163	96.0%	0.661	0.0257	94.8%	0.625
29	0.0151	96.2%	0.682	0.0257	94.7%	0.641
30	0.0147	96.3%	0.689	0.0268	94.5%	0.604
31	0.0147	96.2%	0.693	0.0253	94.9%	0.624
32	0.0181	95.6%	0.677	0.0389	92.0%	0.535
33	0.0190	95.5%	0.643	0.0256	94.8%	0.610
34	0.0148	96.2%	0.687	0.0256	94.8%	0.621
35	0.0141	96.4%	0.701	0.0253	94.8%	0.648





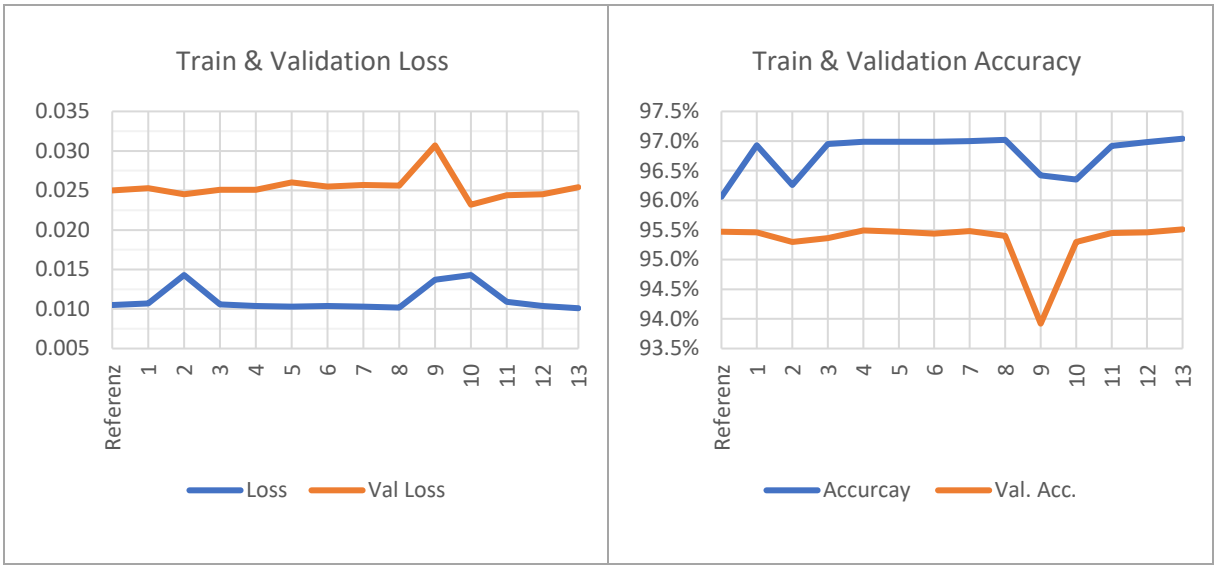
Anlage I: Fine-Tuning des PointNet++ Modells

Zur Optimierung des Modells werden verschiedene Hyperparameter getestet. Zum einen soll die Verwendung des AdamW-Optimierers untersucht werden, zum anderen die Reduktion der Lernrate im Lernprozess. Dazu wird der *Exponential Learningrate Scheduler* verwendet. Als Verlustfunktion dient der Focal Loss mit $\gamma = 4$.

Anlage I-1: Trainingsprotokoll bei der Verwendung des Optimierers AdamW

Epoche	Loss	Accurcay	IoU	Val. Loss	Val. Acc.	Val. IoU
Referenz	0.0105	96.1%	-	0.025	95.5%	0.646
1	0.0107	96.9%	0.717	0.0253	95.5%	0.627
2	0.0143	96.3%	0.676	0.0245	95.3%	0.631
3	0.0106	97.0%	0.719	0.0251	95.4%	0.628
4	0.0104	97.0%	0.721	0.0251	95.5%	0.636
5	0.0103	97.0%	0.723	0.0260	95.5%	0.634

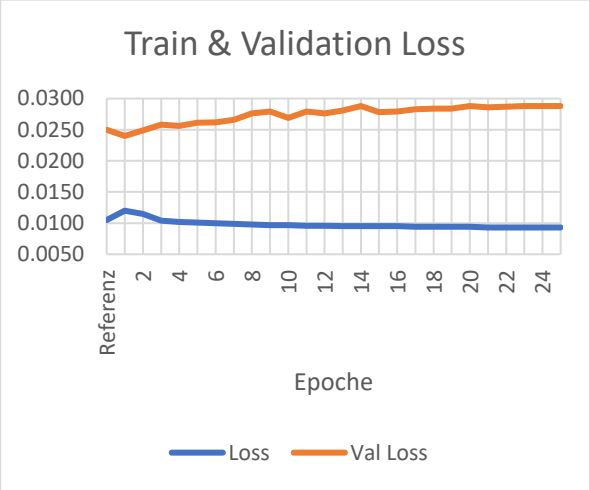
6	0.0104	97.0%	0.730	0.0255	95.4%	0.644
7	0.0103	97.0%	0.727	0.0257	95.5%	0.648
8	0.0102	97.0%	0.731	0.0256	95.4%	0.665
9	0.0137	96.4%	0.704	0.0307	93.9%	0.556
10	0.0143	96.4%	0.679	0.0232	95.3%	0.659
11	0.0109	96.9%	0.724	0.0244	95.5%	0.647
12	0.0104	97.0%	0.732	0.0245	95.5%	0.648
13	0.0101	97.0%	0.735	0.0254	95.5%	0.634

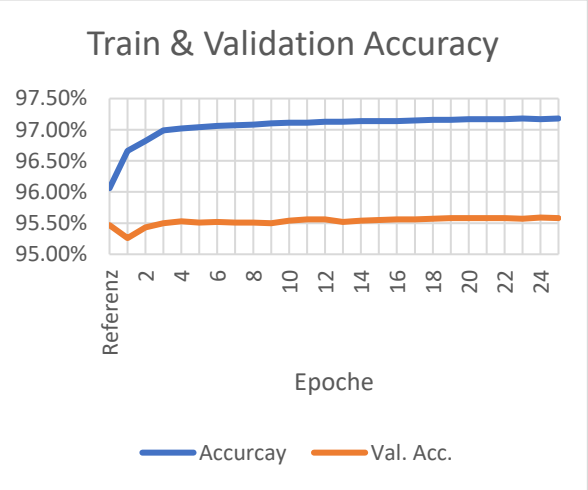


Anlage I-2: Trainingsprotokolle bei Verwendung eines Learning-Rate-Schedulers

Epoche	Loss	Accurcay	IoU	Val. Loss	Val. Acc.	Val. IoU
Referenz	0.0105	96.06%	-	0.0250	95.47%	0.6460
1	0.012	96.66%	0.706	0.0240	95.26%	0.618
2	0.0115	96.82%	0.706	0.0249	95.43%	0.638
3	0.0104	96.99%	0.727	0.0258	95.50%	0.64
4	0.0102	97.02%	0.731	0.0256	95.53%	0.653
5	0.0101	97.04%	0.734	0.0261	95.51%	0.641
6	0.01	97.06%	0.731	0.0262	95.52%	0.63
7	0.0099	97.07%	0.738	0.0266	95.51%	0.634
8	0.0098	97.08%	0.736	0.0276	95.51%	0.649
9	0.0097	97.10%	0.735	0.0279	95.50%	0.658
10	0.0097	97.11%	0.736	0.0269	95.54%	0.649
11	0.0096	97.11%	0.736	0.0279	95.56%	0.641
12	0.0096	97.13%	0.743	0.0276	95.56%	0.656
13	0.0095	97.13%	0.744	0.0281	95.52%	0.644
14	0.0095	97.14%	0.742	0.0288	95.54%	0.656
15	0.0095	97.14%	0.733	0.0278	95.55%	0.649
16	0.0095	97.14%	0.731	0.0279	95.56%	0.641

17	0.0094	97.15%	0.742	0.0283	95.56%	0.633
18	0.0094	97.16%	0.739	0.0284	95.57%	0.644
19	0.0094	97.16%	0.747	0.0284	95.58%	0.645
20	0.0094	97.17%	0.736	0.0288	95.58%	0.656
21	0.0093	97.17%	0.745	0.0286	95.58%	0.654
22	0.0093	97.17%	0.741	0.0287	95.58%	0.664
23	0.0093	97.18%	0.741	0.0288	95.57%	0.642
24	0.0093	97.17%	0.740	0.0288	95.59%	0.646
25	0.0093	97.18%	0.747	0.0288	95.58%	0.647





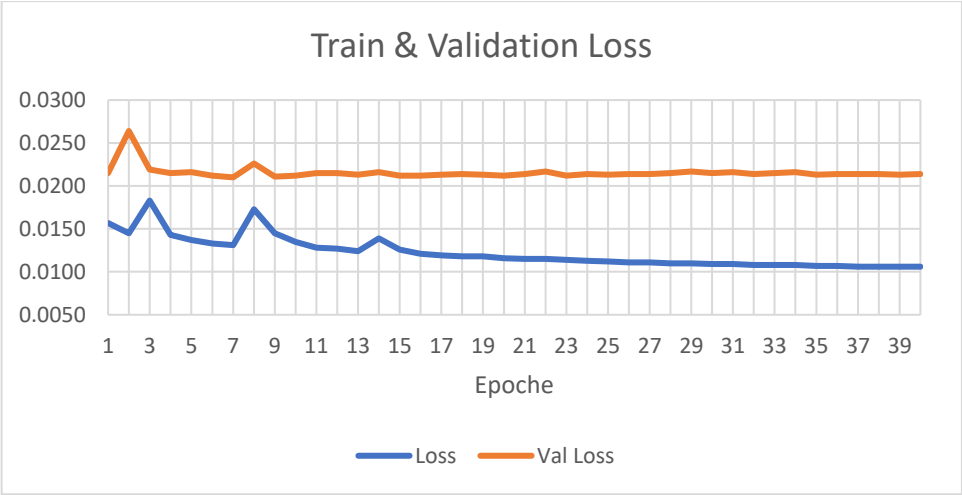
Anlage J: Finale Trainingsprozesse

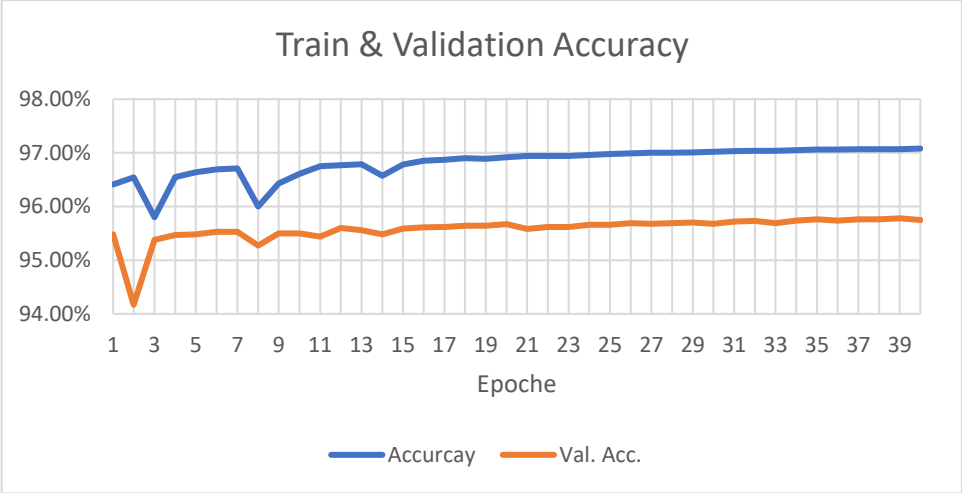
Nachfolgend sollen die Ergebnisse jeder Epoche aus den beiden finalen Trainingsprozessen dargestellt werden. Bei dem ersten Trainingsprozess werden alle acht verfügbaren Attribute verwendet (X, Y, Z, Hue, Saturation, Value, Anzahl Stereomodelle und Punktpräzision) und beim zweiten Prozess nur die Radiometrie und Geometrie (X, Y, Z, Hue, Saturation, Value). Dabei wird der Focal Loss mit $\gamma = 4$ als Verlustfunktion und der AdamW-Optimierer mit einer Lernrate von 0,001 verwendet. Ab der zehnten Epoche nimmt die Lernrate exponentiell um den Faktor $\lambda = 0{,}9$ ab. Maximale Genauigkeiten bzw. minimale Verluste werden nachfolgend in Rot hinterlegt.

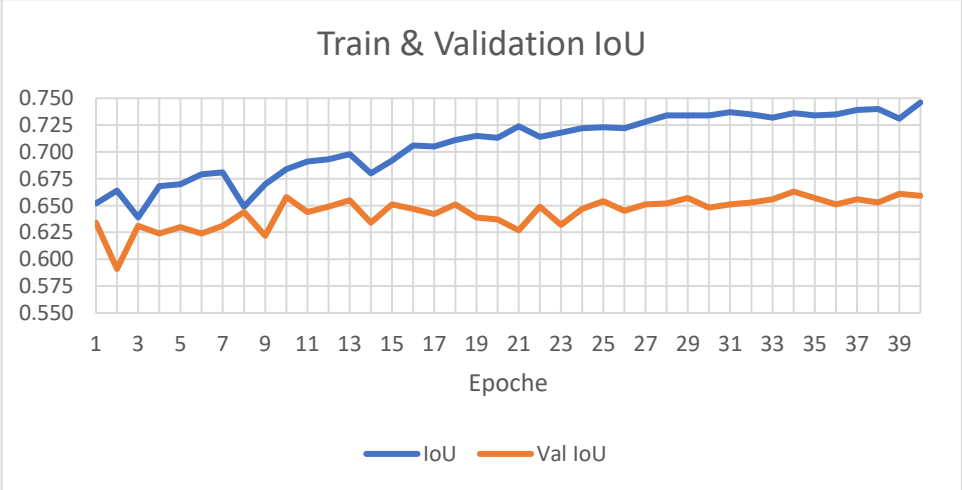
Anlage J-1: Training mit allen Attributen

Epoche	Loss	Accurcay	IoU	Val. Loss	Val. Acc.	Val. IoU
1	0.0157	96.41%	0.652	0.0215	95.48%	0.634
2	0.0145	96.54%	0.664	0.0264	94.17%	0.591
3	0.0183	95.80%	0.639	0.0219	95.38%	0.631
4	0.0143	96.55%	0.668	0.0215	95.47%	0.624
5	0.0137	96.64%	0.670	0.0216	95.48%	0.630
6	0.0133	96.69%	0.679	0.0212	95.53%	0.624

7	0.0131	96.71%	0.681	0.0210	95.53%	0.631
8	0.0173	96.00%	0.649	0.0226	95.27%	0.644
9	0.0145	96.43%	0.670	0.0211	95.50%	0.622
10	0.0135	96.61%	0.684	0.0212	95.50%	0.658
11	0.0128	96.75%	0.691	0.0215	95.44%	0.644
12	0.0127	96.77%	0.693	0.0215	95.60%	0.649
13	0.0124	96.79%	0.698	0.0213	95.56%	0.655
14	0.0139	96.57%	0.680	0.0216	95.48%	0.634
15	0.0126	96.78%	0.692	0.0212	95.59%	0.651
16	0.0121	96.85%	0.706	0.0212	95.61%	0.647
17	0.0119	96.87%	0.705	0.0213	95.62%	0.642
18	0.0118	96.90%	0.711	0.0214	95.64%	0.651
19	0.0118	96.89%	0.715	0.0213	95.64%	0.639
20	0.0116	96.92%	0.713	0.0212	95.67%	0.637
21	0.0115	96.94%	0.724	0.0214	95.58%	0.627
22	0.0115	96.94%	0.714	0.0217	95.62%	0.649
23	0.0114	96.94%	0.718	0.0212	95.62%	0.632
24	0.0113	96.96%	0.722	0.0214	95.66%	0.647
25	0.0112	96.98%	0.723	0.0213	95.66%	0.654
26	0.0111	96.99%	0.722	0.0214	95.69%	0.645
27	0.0111	97.00%	0.728	0.0214	95.68%	0.651
28	0.0110	97.00%	0.734	0.0215	95.69%	0.652
29	0.0110	97.01%	0.734	0.0217	95.70%	0.657
30	0.0109	97.02%	0.734	0.0215	95.68%	0.648
31	0.0109	97.03%	0.737	0.0216	95.72%	0.651
32	0.0108	97.04%	0.735	0.0214	95.73%	0.653
33	0.0108	97.04%	0.732	0.0215	95.69%	0.656
34	0.0108	97.05%	0.736	0.0216	95.74%	0.663
35	0.0107	97.06%	0.734	0.0213	95.76%	0.657
36	0.0107	97.06%	0.735	0.0214	95.74%	0.651
37	0.0106	97.07%	0.739	0.0214	95.76%	0.656
38	0.0106	97.07%	0.740	0.0214	95.76%	0.653
39	0.0106	97.07%	0.731	0.0213	95.78%	0.661
40	0.0106	97.08%	0.746	0.0214	95.75%	0.659



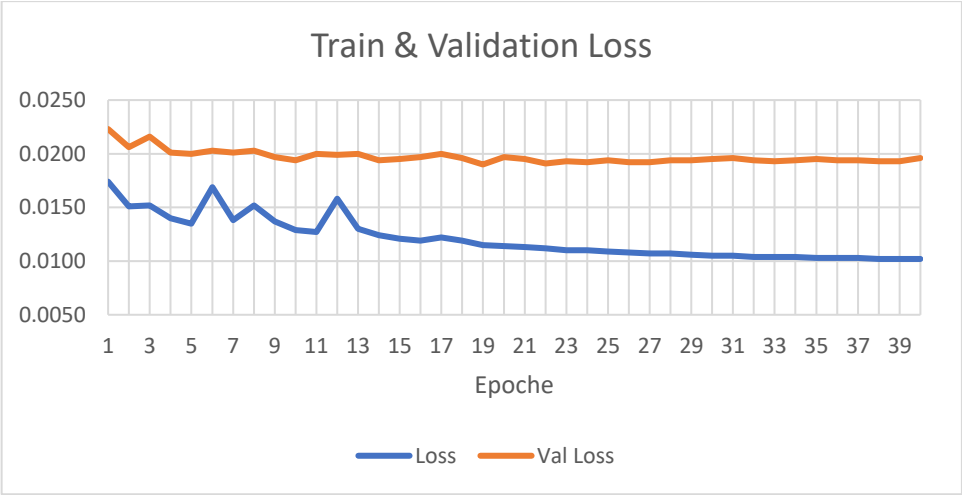


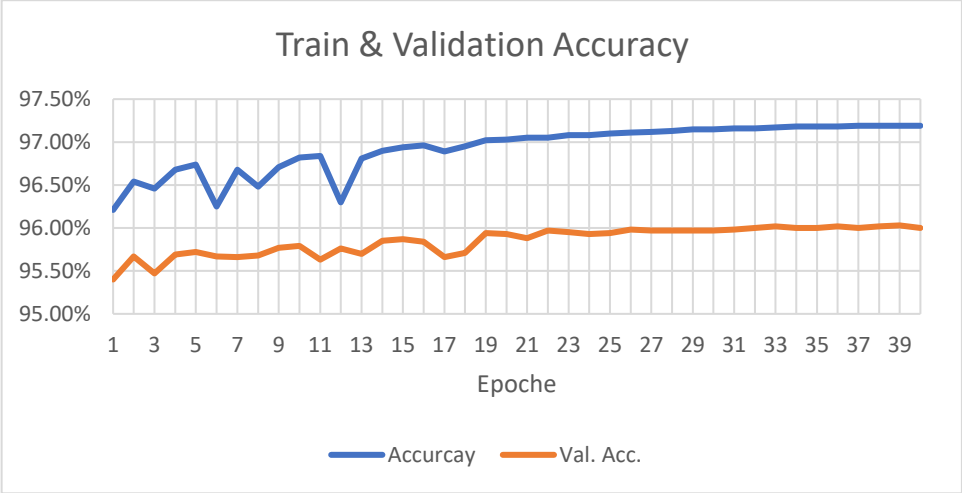


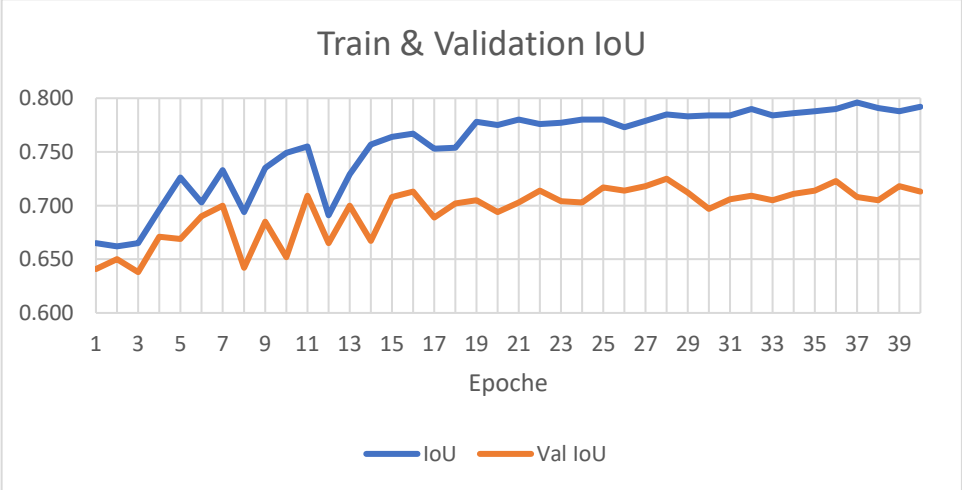
Anlage J-2: Training mit radiometrischen und geometrischen Attributen

Epoche	Loss	Accurcay	IoU	Val. Loss	Val. Acc.	Val. IoU
1	0.0174	96.21%	0.665	0.0223	95.40%	0.641
2	0.0151	96.54%	0.662	0.0206	95.67%	0.650
3	0.0152	96.46%	0.665	0.0216	95.47%	0.638
4	0.0140	96.68%	0.696	0.0201	95.69%	0.671

5	0.0135	96.74%	0.726	0.0200	95.72%	0.669
6	0.0169	96.25%	0.703	0.0203	95.67%	0.690
7	0.0138	96.68%	0.733	0.0201	95.66%	0.700
8	0.0152	96.48%	0.694	0.0203	95.68%	0.642
9	0.0137	96.71%	0.735	0.0197	95.77%	0.685
10	0.0129	96.82%	0.749	0.0194	95.79%	0.652
11	0.0127	96.84%	0.755	0.0200	95.63%	0.709
12	0.0158	96.30%	0.691	0.0199	95.76%	0.665
13	0.0130	96.81%	0.729	0.0200	95.70%	0.700
14	0.0124	96.90%	0.757	0.0194	95.85%	0.667
15	0.0121	96.94%	0.764	0.0195	95.87%	0.708
16	0.0119	96.96%	0.767	0.0197	95.84%	0.713
17	0.0122	96.89%	0.753	0.0200	95.66%	0.689
18	0.0119	96.95%	0.754	0.0196	95.71%	0.702
19	0.0115	97.02%	0.778	0.0190	95.94%	0.705
20	0.0114	97.03%	0.775	0.0197	95.93%	0.694
21	0.0113	97.05%	0.780	0.0195	95.88%	0.703
22	0.0112	97.05%	0.776	0.0191	95.97%	0.714
23	0.0110	97.08%	0.777	0.0193	95.95%	0.704
24	0.0110	97.08%	0.780	0.0192	95.93%	0.703
25	0.0109	97.10%	0.780	0.0194	95.94%	0.717
26	0.0108	97.11%	0.773	0.0192	95.98%	0.714
27	0.0107	97.12%	0.779	0.0192	95.97%	0.718
28	0.0107	97.13%	0.785	0.0194	95.97%	0.725
29	0.0106	97.15%	0.783	0.0194	95.97%	0.712
30	0.0105	97.15%	0.784	0.0195	95.97%	0.697
31	0.0105	97.16%	0.784	0.0196	95.98%	0.706
32	0.0104	97.16%	0.790	0.0194	96.00%	0.709
33	0.0104	97.17%	0.784	0.0193	96.02%	0.705
34	0.0104	97.18%	0.786	0.0194	96.00%	0.711
35	0.0103	97.18%	0.788	0.0195	96.00%	0.714
36	0.0103	97.18%	0.790	0.0194	96.02%	0.723
37	0.0103	97.19%	0.796	0.0194	96.00%	0.708
38	0.0102	97.19%	0.791	0.0193	96.02%	0.705
39	0.0102	97.19%	0.788	0.0193	96.03%	0.718
40	0.0102	97.19%	0.792	0.0196	96.00%	0.713







Anlage K: Klassifizierte Punktwolke aus dem Validierungsdatensatz

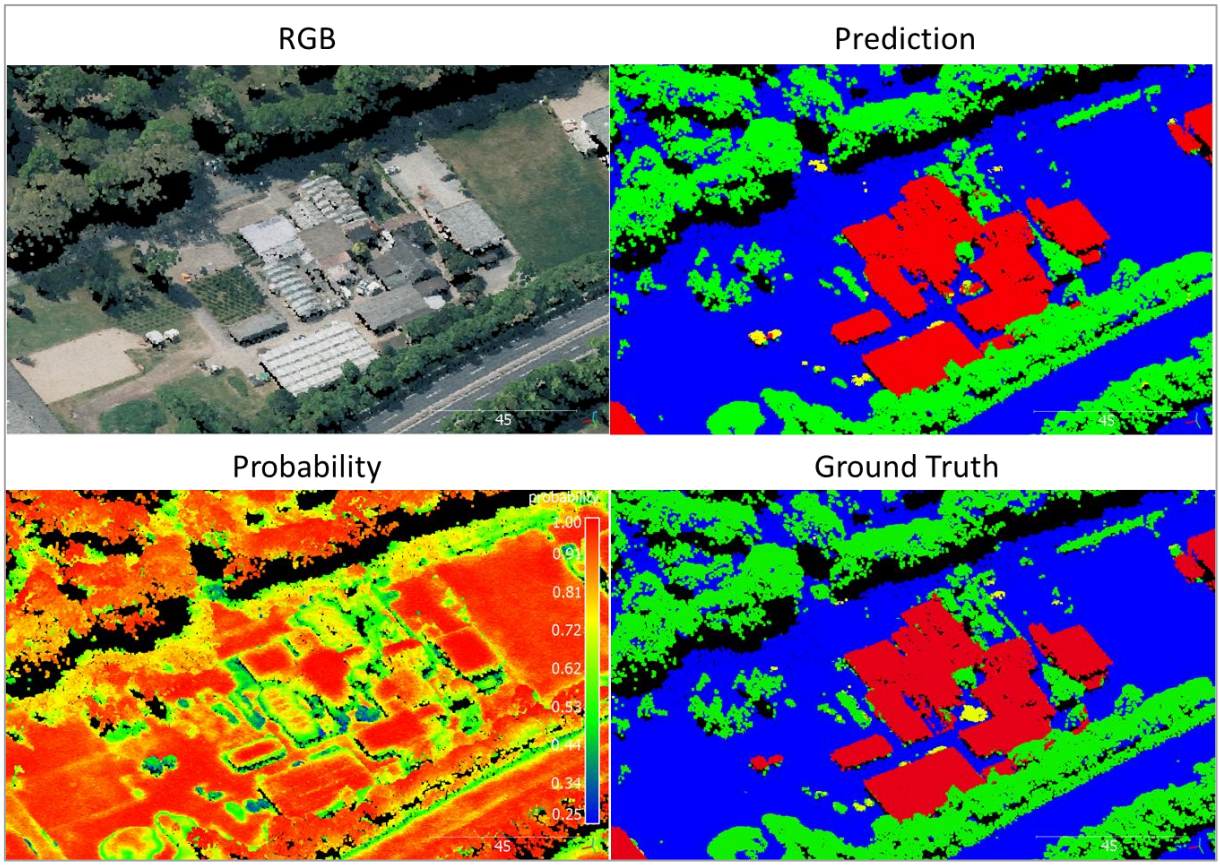
Zur Validierung des Modells wurde der gesamte Validierungsdatensatz (Patch 555000_5800000) klassifiziert. Unter dem nachfolgenden Link sind die klassifizierten Punktwolken herunterladbar. Nachfolgende Daten sind zu finden:

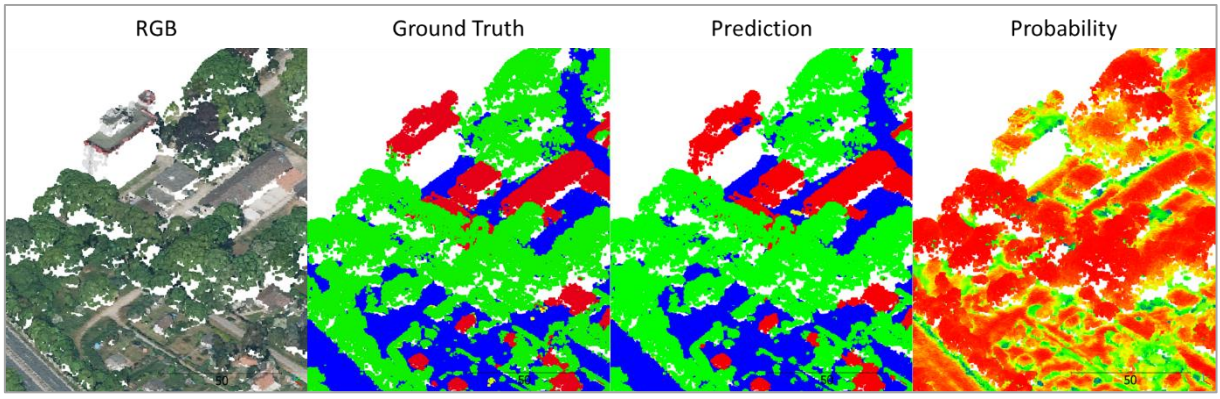
- **555000-5800000_6-atts.laz**: Klassifizierte Punktwolke durch die Verwendung der sechs Attribute XYZ + HSV.
- **555000-5800000_all-atts.laz**: Klassifizierte Punktwolke durch die Verwendung aller Attribute

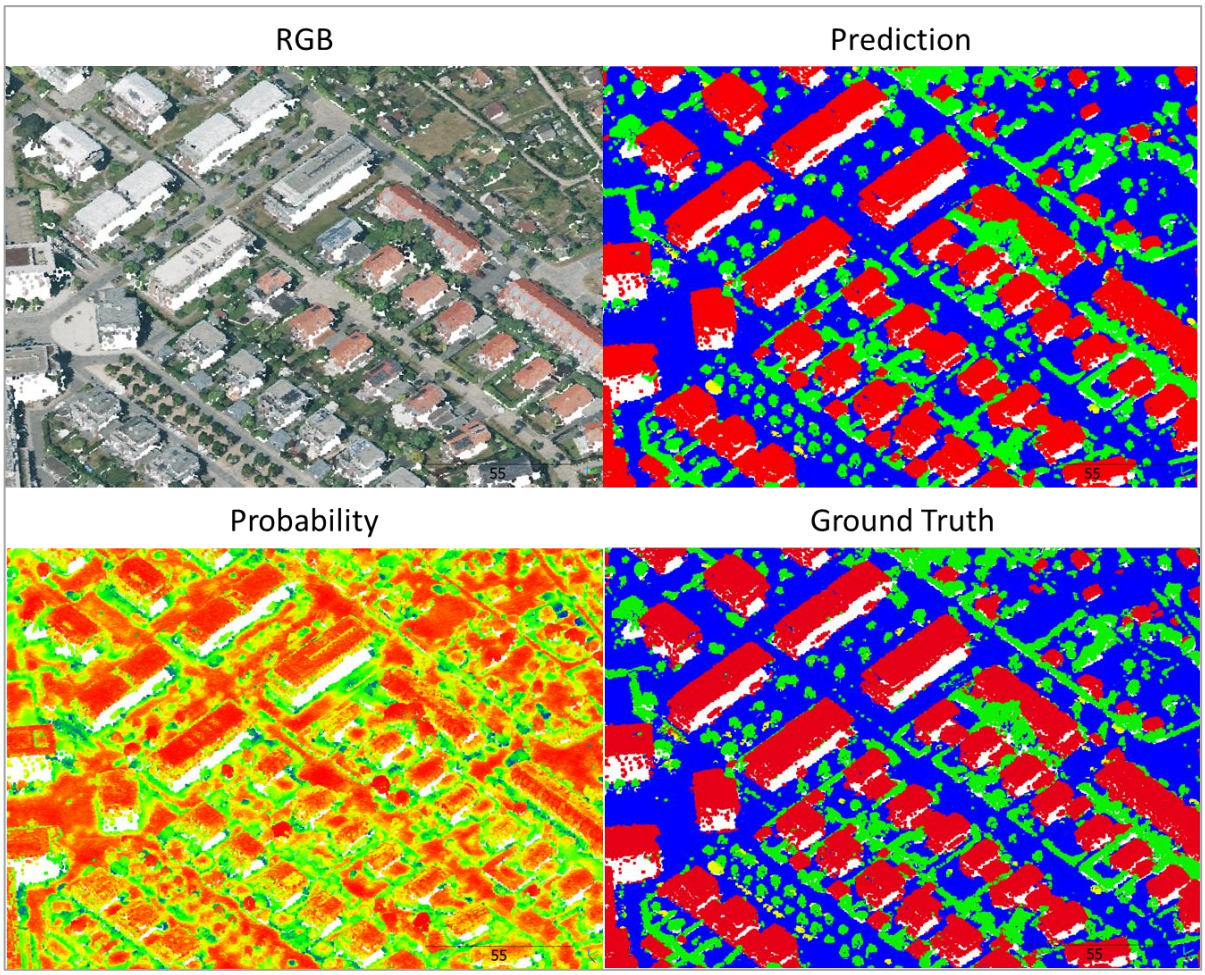
Link: https://s.gwdg.de/kBluTC

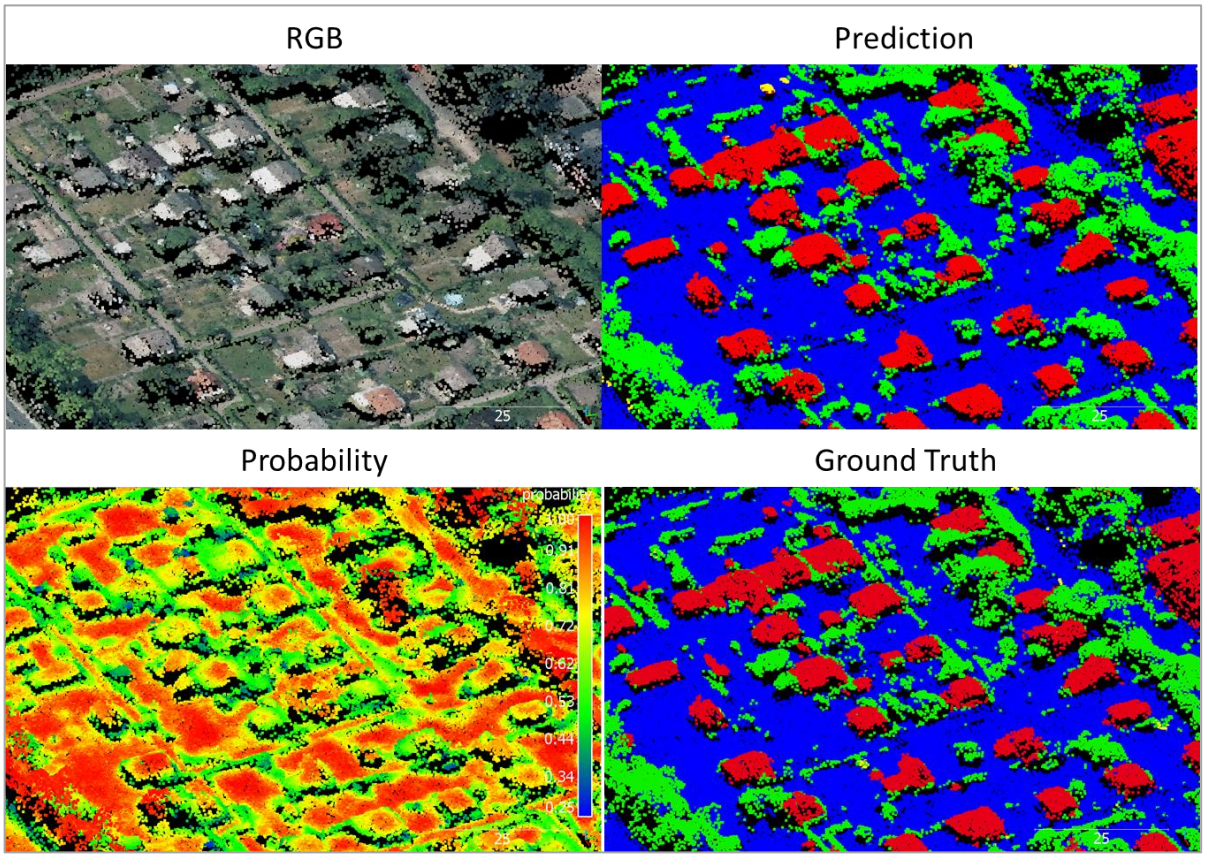
Anlage L: Beispielhafte Darstellungen der klassifizierten Punktwolke

Klassifikation mit allen Attributen:






RGB
Prediction
Probability
Ground Truth


RGB
Prediction
Probability
Ground Truth

Klassifikation bei Verwendung von sechs Attributen: