Semantische Klassifizierung von 3D-Punktwolken

Christian Köhler, Marc Donner, Ralf Donner

TU Bergakademie Freiberg, Institut für Markscheidewesen und Geodäsie

ZUSAMMENFASSUNG:

Für die automatisierte Überwachung technischer Einbauten in untertägigen Bergwerksanlagen auf Basis dreidimensionaler Punktwolken wurde ein Geomonitoringverfahren, bestehend aus Datenaufnahme und –analyse, entwickelt. Es werden zwei Ansätze zur semantischen Klassifizierung von dreidimensionalen Punktwolken betrachtet, die Multi-Skalen-Feature-Extraktion und die Anwendung eines dreidimensionalen Faltenden Neuronalen Netzes. Die Methode der Multi-Skalen-Feature-Extraktion bestimmt durch festgelegte Berechnungsvorschriften Features allein aus den Koordinaten eines Punktes und seiner Nachbarn auf mehreren Längenskalen. Diese werden zu Feature-Vektoren zusammengefasst und dienen als Input für einen Random Forest-Klassifizierer. Die Anwendung eines dreidimensionalen Faltenden Neuronalen Netzes erfordert nur die Vorverarbeitung der Punktwolke zu einem Voxel-Grid und liefert dann direkt Klassifizierungsergebnisse. In einer exemplarischen Anwendung beider Ansätze zur Detektion von Stempeln, Schienen und Stößen in einer untertägigen Szene werden Klassifizierungsgenauigkeiten von über 90 % erreicht.

1 Zielstellung und UPNS4D+ Projekt

Unter den Schlagwörtern "Bergbau 4.0", "Big Data" oder "Internet of Things" werden am Institut für Markscheidewesen und Geodäsie der TU Bergakademie Freiberg zukünftig bedeutende Themen der Bergbauindustrie mit der zunehmend interdisziplinären Anwendung von Methoden der softwaretechnischen Modellierung/Simulation, Sensortechnik, Datenakkumulation und -verarbeitung bis hin zu autonomer Robotik und Methoden des maschinellen Lernens bearbeitet. Das Projekt "Untertägiges 4D+-Positionierungs-, Navigations- und Mapping System zur hochselektiven, effizienten und im höchsten Maße sicheren Gewinnung wirtschaftsstrategischer mineralischer Rohstoffe" (UPNS4D+ [1]) zielt dabei u.a. auf die Entwicklung eines Systems zur autonomen, untertägigen Navigation und Kartierung, Erkundung und kontinuierlichen Änderungsüberwachung. Dazu wird ein Multi-Sensor-Erkundungsfahrzeug, ausgestattet mit RGB- und Hyperspektralkameras, einem Laserscanner und Radarsensoren, entwickelt.

Innerhalb des UPNS4D+-Projektes wird im Teilvorhaben "Merkmalsextraktion" unter dem Aspekt der autonomen Kartierung eine automatische Erzeugung und Aktualisierung des Risswerks aus den vorhandenen Sensordaten angestrebt. Das Risswerk enthält u.a. Informationen über technische Installationen, wie z.B. Schienen oder Stempel. Die Position dieser Installationen gilt es aus den erfassten Daten automatisch zu extrahieren.

Dabei stehen in ausreichendem Maße bislang nur die geometrischen Daten des Laserscanners zur Verfügung. Dieser liefert Messwerte in Form einer Punktwolke, einer ungeordneten Menge von Punkten, welche allein durch ihre Koordinaten p = (x, y, z) gegeben sind. Die Punktwolke in Abb. 1a zeigt einen untertägigen Streckenabschnitt und enthält ca. 120.000 Punkte. Das Ziel, Positionen von technischen Installationen aus den Sensordaten zu extrahieren, übersetzt sich für die vorliegenden Punktwolken in die Aufgabe, jedem einzelnen Punkt der Punktwolke ein zusätzliches Attribut, d.h. ein Label ("Schiene", "Stempel", "Stoß", etc.) zuzuordnen. Das Ergebnis einer solchen semantischen Klassifizierung ist in Abb. 1b farblich kodiert dargestellt.

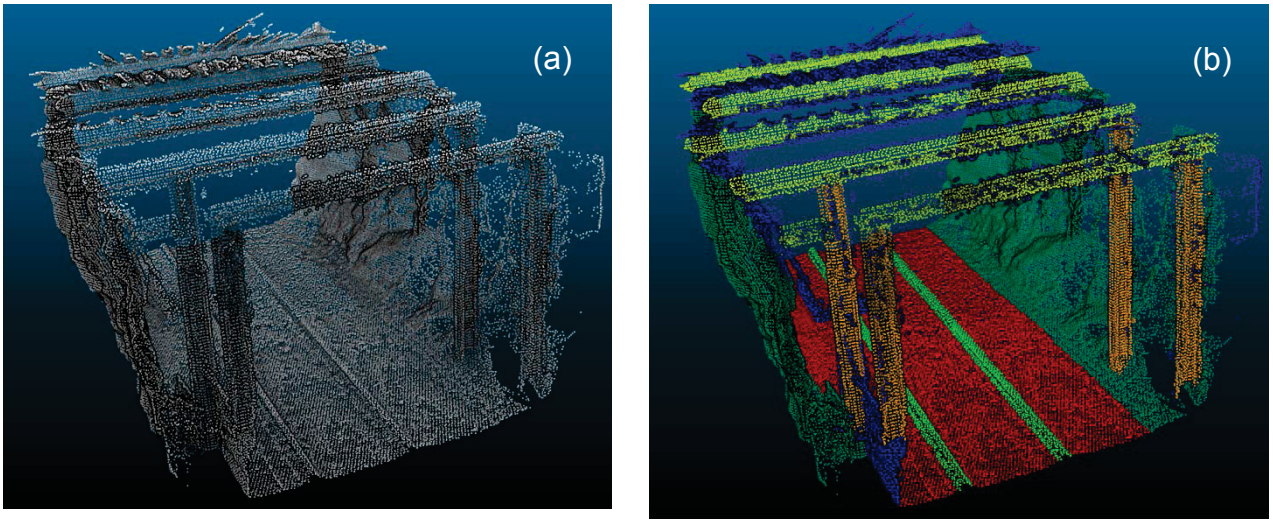


Abb. 1: (a) Punktwolke eines untertägigen Streckenabschnitts mit ca. 120.000 Punkten. (b) Jedem einzelnen Punkt ist farblich kodiert ein Label ("Schiene" (hellgrün), "Stoß" (dunkelgrün), "Stempel" (orange)) zugeordnet.

2 Methoden

Die semantische Klassifikation von Punktwolken kann mit zwei generellen Herangehensweisen erfolgen, der Extraktion von "handgemachten" geometrischen Merkmalen (Features) oder der Klassifizierung mit künstlichen neuronalen Netzen. Das prinzipielle Vorgehen bei der Klassifikation mit Features startet mit den Koordinaten der Punkte der Punktwolke. Aus diesen werden nach vorgegebenen Berechnungsvorschriften Features extrahiert. Alle berechneten Features werden zu einem Feature-Vektor zusammengefasst. Der Feature-Vektor dient als Argument eines Klassifizierungsalgorithmus, welcher letztendlich das Label (Kennzeichen der ermittelten Klasse) liefert. Das Vorgehen bei der Klassifizierung mit Neuronalen Netzen ist ähnlich: Die Punktkoordinaten dienen als Startwerte eines Neuronalen Netzes, welches nun die Label direkt ableitet. Beide Ansätze wurden realisiert und werden im Folgenden vorgestellt.

2.1 Klassifikation durch Extraktion geometrischer Features in verschiedenen Skalen

Die einzelnen Punkte der Punktwolke sind nur durch ihre drei Koordinaten gegeben, insbesondere liegen uns keine weiteren Eigenschaften wie z.B. Intensitäts- oder Farbwerte vor. Um nun überhaupt Features für einen einzelnen Punkt berechnen zu können, muss man dessen Umgebung und die darin enthaltenen Nachbarpunkte einbeziehen. Es hat sich erwiesen, dass die Features aussagekräftiger werden, wenn man nicht nur eine solche Umgebung betrachtet, sondern mehrere, sich durch ihre Längenskala unterscheidende [2].

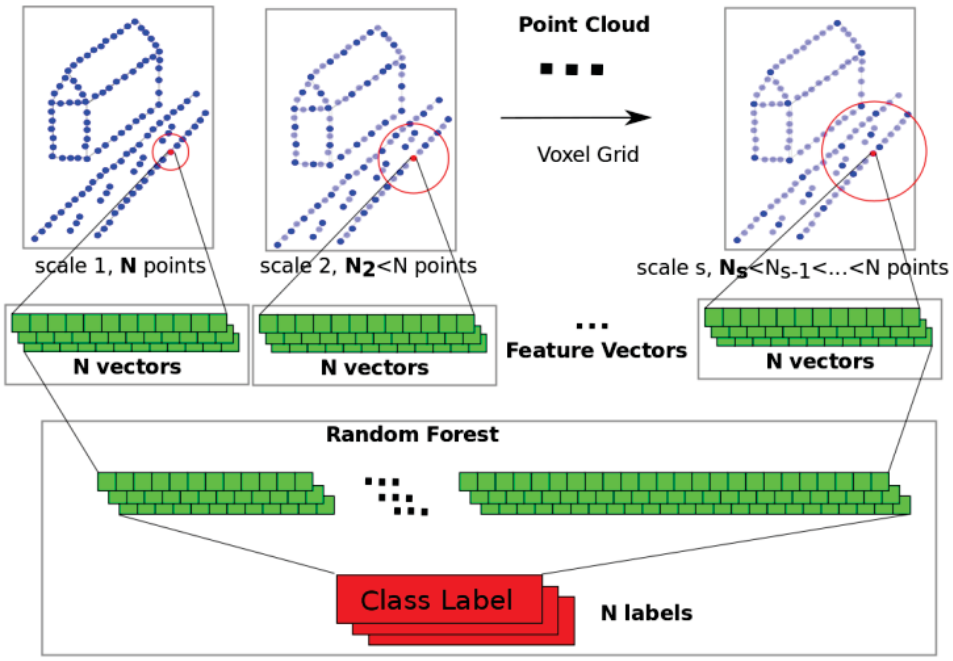


Abb. 2: Schema für den Algorithmus zur semantischen Klassifizierung durch Feature Extraktion auf verschiedenen Längenskalen. Für jeden Punkt werden auf jeder Skala unter Einbeziehung seiner jeweils zehn nächsten Nachbarn Features berechnet und zu einem Feature-Vektor zusammengefasst. Die Kombination der Feature-Vektoren aller Skalen dient als Input in einen Random Forest-Klassifizierer, welcher die Labels liefert.

Man führt diese verschiedenen Längenskalen mit Hilfe eines Voxel-Grid-Filters in die Punktwolke ein. Dabei wird die Punktwolke in regelmäßig angeordnete Würfel mit vorgegebener Kantenlänge (= Voxel[2]) eingeteilt und alle innerhalb eines Voxels gelegenen Punkte durch den Mittelpunkt des Voxels ersetzt. Man erhält also durch Anwendung des Voxel-Grid-Filters mit verschiedenen Kantenlängen Punktwolken mit unterschiedlichen Längenskalen und variierender Anzahl von Punkten, siehe Abb. 2.

Tab. 1: Verwendete Merkmale, berechnet aus den Eigenwerten $\lambda_1 \geq \lambda_2 \geq \lambda_3$ und zugehörigen Eigenvektoren $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ der Hauptkomponentenanalyse der zehn nächsten Nachbarn (NN) eines Punktes $\mathbf{p}$. k_1, k_2 sind die Hauptkrümmungen in $\mathbf{p}$.

Eigenwerte	Summe	$\lambda_1+\lambda_2+\lambda_3$
	Anisotropie	$(\lambda_1-\lambda_3)/\lambda_1$
	Linearität	$(\lambda_1-\lambda_2)/\lambda_1$
	Planarität	$(\lambda_2-\lambda_3)/\lambda_1$
	Voluminität	λ_3/λ_1
	Oberflächenvariabilität	$\lambda_3/(\lambda_1+\lambda_2+\lambda_3)$
	Omnivarianz	$(\lambda_1\lambda_2\lambda_3)^{1/3}$
	Eigenentropie	$\sum_{i=1}^{3} \lambda_i \ln\lambda_i$
	Vertikalität	$1-\langle \mathbf{e}_z, \mathbf{e}_3 \rangle$
Momente	1. Ordnung, 1. Achse	$\sum_{i\in NN} \langle p_i - p, e_1 \rangle$
	1. Ordnung, 2. Achse	$\sum_{i\in NN} \langle p_i - p, e_2 \rangle$
	2. Ordnung, 1. Achse	$\sum_{i\in NN} \langle p_i - p, e_1 \rangle^2$
	2. Ordnung, 2. Achse	$\sum_{i\in NN} \langle p_i - p, e_2 \rangle^2$
Krümmung	Gauß'sche Krümmung	$k_1 k_2$
	Mittlere Krümmung	$(k_1+k_2)/2$

Für jeden Punkt der originalen, nicht von einem Voxel-Grid-Filter ausgedünnten Punktwolke, werden nun seine zehn nächsten Nachbarn betrachtet. Diese nächsten Nachbarn werden dabei in den

[2] Voxel: zusammengesetzt aus dem englischen volume vox und el von element bezeichnet in der Computersprache ein Datenelement in einem dreidimensionalen regelmäßigen Gitter; vgl. https://de.wikipedia.org/wiki/Voxel, 22.3.2o18

gefilterten Punktwolken gesucht. Für jeden Satz von zehn Punkten wird nun eine Hauptkomponentenanalyse durchgeführt und man erhält die Eigenwerte $\lambda_1 \geq \lambda_2 \geq \lambda_3$ und zugehörigen Eigenvektoren $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ auf jeder Längenskala. Aus diesen Eigenwerten und -vektoren werden die Features berechnet; sie sind in Tab. 1 zusammengefasst [3, 4, 5].

Für jeden der Punkte der Punktwolke liegt nun auf jeder Längenskala ein Satz von 15 Features vor, welcher zu einem Feature-Vektor zusammengefasst wird. Die Kombination der Feature-Vektoren der verschiedenen Längenskalen dient als Input für einen Random Forest [6], den hier eingesetzten Klassifizierer. Dieser liefert schließlich ein Label für jeden einzelnen Punkt.

2.2 Klassifizierung durch Faltende Neuronale Netze

Im Folgenden wird die semantische Klassifizierung von Punktwolken durch Faltende Neuronale Netze beschrieben. Dazu führen wir zunächst Neuronale Netze ein, um anschließend eine spezielle Art von Neuronalen Netzen, die Faltenden Neuronalen Netze zu beschreiben. Schließlich betrachten wir die notwendige Vorbereitung der Punktwolke und stellen die Architektur des verwendeten Faltenden Neuronalen Netzes vor.

2.2.1 Neuronale Netze

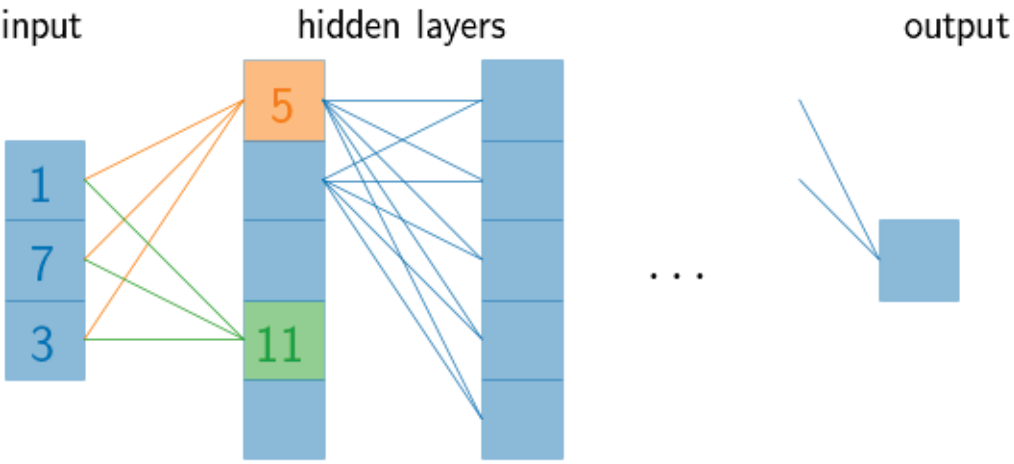


Abb. 3: Ein künstliches neuronales Netz besteht aus Knoten (als Quadrate dargestellt), welche in Layern angeordnet und durch Gewichte (Linien) verbunden sind.

Künstliche neuronale Netze (NN) bestehen aus Knoten, deren Zustand durch eine einzelne reelle Zahl definiert ist. Die Knoten sind in Layern (Schichten) angeordnet, wobei Knoten ein und derselben Layer nicht miteinander verbunden sind, aber jeder Knoten eines Layers mit jedem Knoten eines benachbarten Layers verbunden ist (siehe Abb. 3). Die Verbindungen der einzelnen Layer heißen Gewichte und werden durch reelle Zahlen repräsentiert. Ein NN besteht aus zwei oder mehr aufeinander folgenden Layern. Der erste Layer ist der Eingabe- oder Input-Layer, der letzte der Output-Layer, die dazwischenliegenden heißen Hidden Layer (verborgene Schichten).

In der normalen Anwendung (forward pass) gibt man die Werte auf den Knoten der Input-Layer vor und propagiert diese entlang der Verbindungen von Layer zu Layer bis zum Output-Layer. Die Werte der Output-Knoten stellen das Ergebnis dar. Um den Wert eines Knotens der folgenden Layer zu erhalten, berechnet man eine Linearkombination aus den Werten der Knoten der vorangegangenen Layer und den entsprechenden Gewichten, addiert ein Offset und wendet eine nichtlineare Funktion auf das Zwischenergebnis an. Aus der Wiederholung für alle Knoten eines Layers und der Wiederholung für alle Layer ergeben sich schließlich die Werte auf den Output-Knoten als Ergebnis.

Das Ergebnis ist abhängig von den Gewichten, den Offsets und den verwendeten Nichtlinearitäten der einzelnen Layer. Zusammen stellen diese die Parameter des NN dar und sind zunächst unbekannt.

In unserem konkreten Anwendungsfall möchten wir ein NN zur semantischen Klassifizierung von Punktwolken benutzen, d.h. wir nutzen die Punkte der Punktwolke als Input und wollen als Ergebnis ein Label erhalten.

Um solch ein gewünschtes Resultat zu erhalten, muss das NN zunächst trainiert werden, d.h. die Parameter des NN müssen an die entsprechende Aufgabe angepasst werden. Dazu initialisiert man die Parameter des NN (meist) zufällig und trainiert das NN anhand von Trainingsdaten (dieses Vorgehen wird als supervised learning bezeichnet). Das Training besteht aus geordneten Zuordnungen von Input-Werten und den richtigen Output-Werten. Man berechnet ausgehend von den Input-Werten den Output des NN und vergleicht diesen mit dem gewünschtem, durch die Trainingsdaten vorgegebenen Output. In der Regel (insbesondere zu Beginn des Trainings) sind diese verschieden. Anhand einer Kostenfunktion, deren Argument die Differenz zwischen berechnetem und gewünschtem Output ist, werden die Parameter des NN dahingehend angepasst, dass der Wert der Kostenfunktion sinkt. Dies wiederholt man für eine große Zahl an Trainingsdatensätzen. Wurden alle Trainingsdatensätze ein Mal zur Verbesserung der Parameter verwendet, spricht man von einer Trainingsepoche. In der Regel wird mehrere Epochen (üblich sind oft 5-50) trainiert, wobei die mittlere Differenz zwischen berechneten und gewünschten Output und damit der Wert der Kostenfunktion stetig sinken sollte. Wenn die Kostenfunktion unter einen gewünschten Wert fällt, wird das Training beendet. Nun geht man davon aus, dass auch für unbekannte (aber doch ähnliche) Daten ohne bekanntes Ergebnis das korrekte Label berechnet wird.

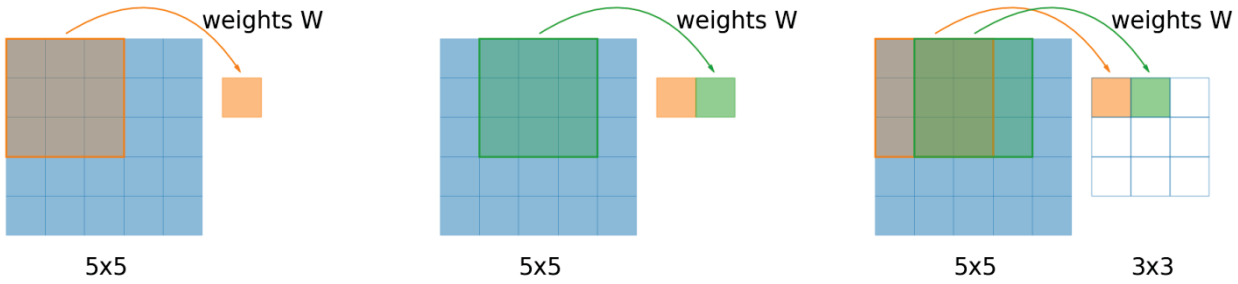
2.2.2 Faltende Neuronale Netze

Faltende künstliche neuronale Netze (FNN) sind eine spezielle Form von NN. Sie sind aus zwei generischen Layern aufgebaut, dem namengebenden Faltenden Layer und einem Pooling-Layer. In der Regel folgt auf einen faltenden Layer ein Pooling-Layer. Ein FNN besteht dann aus mehreren dieser Blöcke.

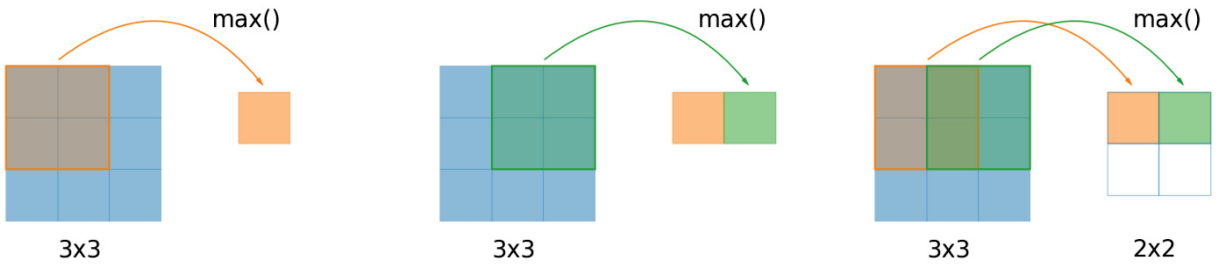
In Abb. 4 sind diese generischen Layer für den zweidimensionalen Fall dargestellt, da der zweidimensionale Fall besser visualisierbar ist und die Verallgemeinerung auf den hier verwendeten dreidimensionalen Fall einfach möglich ist.

Die Knoten der generischen Layer des FNN sind auf einem regelmäßigen zweidimensionalen Grid[3], ähnlich den Pixeln eines Bildes, angeordnet.

Das Charakteristische der faltenden Layer ist, dass der Wert eines Knotens des folgenden Layers nicht aus den Werten aller Knoten des vorangegangenen Layers berechnet wird, sondern nur ein Teil der Knoten der vorangegangenen Layer verwendet wird. Dieser Teil ist ein zweidimensionales Teil-Grid (im Beispiel in Abb. 4a 3 x 3). Die Verbindungen zwischen den Knoten des Teil-Grids und des Knotens der folgenden Layer heißen wieder Gewichte, alle Gewichte zusammen werden Filter genannt, wie in der Bildverarbeitung auch. Wieder findet eine Linearkombination mit den Knoten des Teil-Grids und den Gewichten des Filters Anwendung, gefolgt von einem Offset und einer nichtlinearen Funktion.



(a) 2D-faltende Layer



(b) 2D-Pooling-Layer

Abb. 4: Die generischen Layer eines FNN. (a) illustriert die Verwendung der namengebenden faltenden Layer, (b) die Operation der Pooling-Layer.

Für benachbarte Knoten im Grid der folgenden Layer (grün in Abb. 4a Mitte) werden die Knoten, welche den Wert des Knotens bestimmen durch ein entsprechend verschobenes Teil-Grid festgelegt. Die Berechnung nutzt dabei dieselben Gewichte also den selben Filter. Dies ist ein entscheidender Vorteil in der Reduzierung des Rechenaufwands, denn nur eine kleine Anzahl an Gewichten pro Layer wird benötigt, verglichen mit der Anzahl an Gewichten, wenn jeder Knoten der vorangegangenen Layer mit jedem der folgenden Layer verbunden ist.

Auch im Pooling-Layer, siehe in Abb. 4b, wird ein Teil-Grid verschoben, um die Werte der Knoten des folgenden Layers zu bestimmen. Allerdings erfolgt die Berechnung nicht durch Linearkombination gefolgt von Offset und Nichtlinearität, sondern allein durch Anwendung einer i.a. nichtlinearen

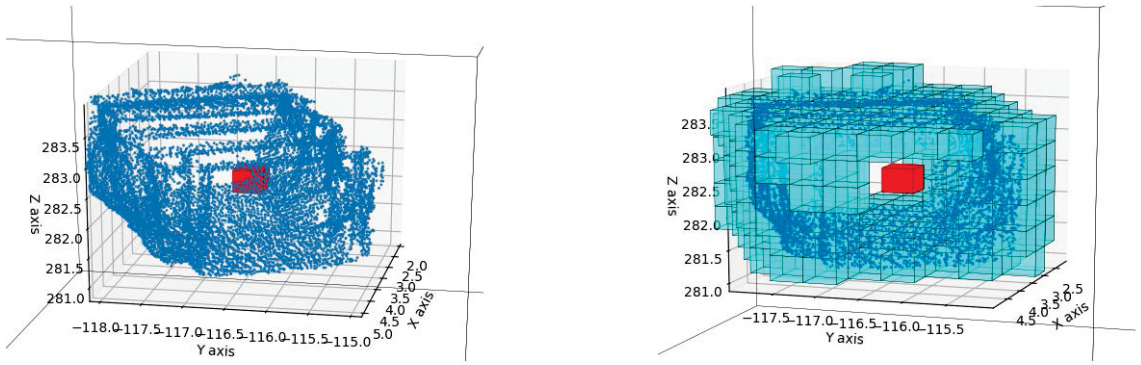
[3] Grid: aus dem Englischen für „[das] Gitter“ oder auch „Raster“, https://de.wikipedia.org/wiki/Grid; 22.3.2o18

Funktion. Wir verwenden in unserer Anwendung die max()-Funktion, welche den maximalen Wert aller vom Teil-Grid abgedeckten Knoten liefert.

2.2.3 Datenvorbereitung: Besetzungsgitter

Wie beschrieben, ist der Input für FNN ein geordnetes Grid, unsere Daten liegen allerdings als ungeordnete Punktwolke vor. Daher müssen die Daten vorbereitet werden, um ein FNN anwenden zu können. Dies erfolgt unter Verwendung eines Voxel-Grids [7].

In Abb. 5 ist dargestellt, wie die Punktwolke in dreidimensionale Pixel (= Voxel) mit fester Kantenlänge gegliedert wird. Ein Voxel wird als besetzt definiert, wenn mindestens ein Punkt der Punktwolke darin liegt. Für jedes besetzte Voxel werden nun die N x N x N umgebenden Voxel betrachtet. Diese binären (eins - besetzt, null - unbesetzt) Voxel-Grids sind der Input für das dreidimensionale FNN. Der Output des FNN wird als Label für das zentrale Voxel definiert.



(a) (b)

Abb. 5: (a) Die ungeordneten Daten der Punktwolke mit einem einzelnen, in rot dargestellten Voxel. In (b) sind alle besetzten Voxel des Voxel-Grids eingetragen. Jedes besetzte Voxel dient als zentrales Voxel eines N x N x N großen Voxelgrids, welches als Input für das FNN verwendet wird.

2.2.4 Architektur: 3D-faltendes neuronales Netz

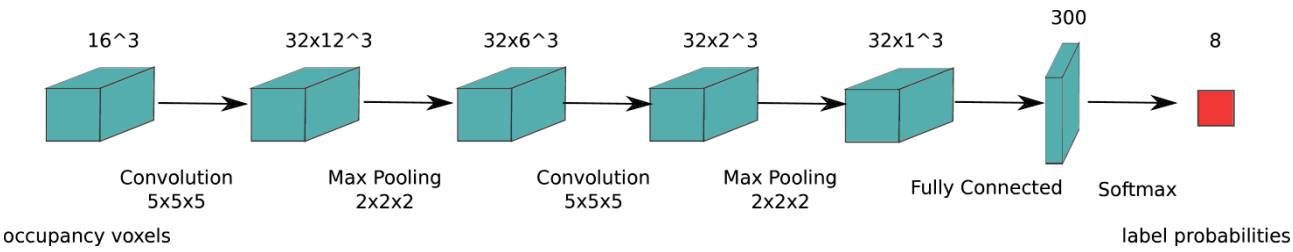


Abb. 6: Die Geometrie des verwendeten FNNs. Die Würfel symbolisieren die einzelnen Layer, repräsentiert durch dreidimensionale Grids mit den darüber angegebenen Dimensionen. Die verwendeten Filter und deren Dimensionen sind darunter angegeben. Als Input dient ein 16 x 16 x 16 großes binäres Voxel-Grid, als Ergebnis erhalten wir Wahrscheinlichkeiten für die Label.

Da wir nun alle Komponenten zur semantischen Klassifizierung von Punktwolken mit FNN eingeführt haben, stellen wir im Folgenden die konkrete Architektur des verwendeten dreidimensionalen FNN vor [8, 9]. In Abb. 6 ist das Input-Layer, bestehend aus einem 16 x 16 x 16-Voxel-Grid zu sehen. Es folgen die Faltung mit 32 verschiedenen 5 x 5 x 5-Filtern und ein max()-Pooling mit entsprechend 32 Filtern der Dimension 2 x 2 x 2. Ein zweiter Block von 5 x 5 x 5-Faltung und 2 x 2 x 2 Pooling schließt sich an. Abschließend folgt noch ein voll verbundener Layer um alle Ergebnisse der 32 verschiedenen Filter zusammenzuführen und schließlich eine Softmax-Klassifizierungsschicht, welche Wahrscheinlichkeiten für die Label liefert. Das Label mit der höchsten Wahrscheinlichkeit wird als Klassifizierungsergebnis für die Punkte im zentralen Voxel des Voxelgrids angesehen.

2.3 Technische Details

Alle Berechnungen, welche Punktwolken involvieren wurden in C++ unter der Verwendung der Point Cloud Library [10] durchgeführt. Für die Programmierung der neuronalen Netze und die Klassifikation wurde Python mit Lasagne [11] und Theano [12] verwendet.

Die Klassifizierung mit Features auf verschiedenen Längenskalen erfolgte mit einem Random Forest-Klassifizierer mit 50 Bäumen zu je 30 Levels [3].

Das FNN wurde mit 123.00 handgelabelten Punkten (siehe Abb. 1b) trainiert. Wir verwendeten sechs Klassen: Boden, Stoß, Stempel, Stütze, Schiene und Sonstiges. Es fand kein balancing (= gleich viele Trainingsdatensätze für alle Klassen) statt, die Gewichte des FNN wurden mit Nesterov Momentum Update (Lernrate 0.05, Momentum 0.9) aktualisiert, die handgelabelten Punkte wurden zu 80/20 in Training- und Testdatensätze aufgeteilt.

Der Erfolg des Trainings ist in Abb. 7 dargestellt: Während der 20 Trainingsepochen sank die Kostenfunktion (loss) kontinuierlich. Die Genauigkeit als das Verhältnis von korrekt zu falsch klassifizierten Punkten erreichte Werte von über 95%. Ein overfitting (übermäßige Spezialisierung des NN auf die Trainingsdaten) konnte ausgeschlossen werden.

Das Training auf der Grundlage der 123.000 handgelabelten Punkte dauerte etwa 40 Minuten auf einem Desktop-PC, die Klassifizierung von ca. 100.000 Punkten wenige Minuten. Die Trainings- und Klassifizierungszeiten hängen dabei stark von der Anzahl der Punkte und der verwendeten Hardware ab und sollen hier nur als Anhaltspunkt für eine Größenordnung dienen.

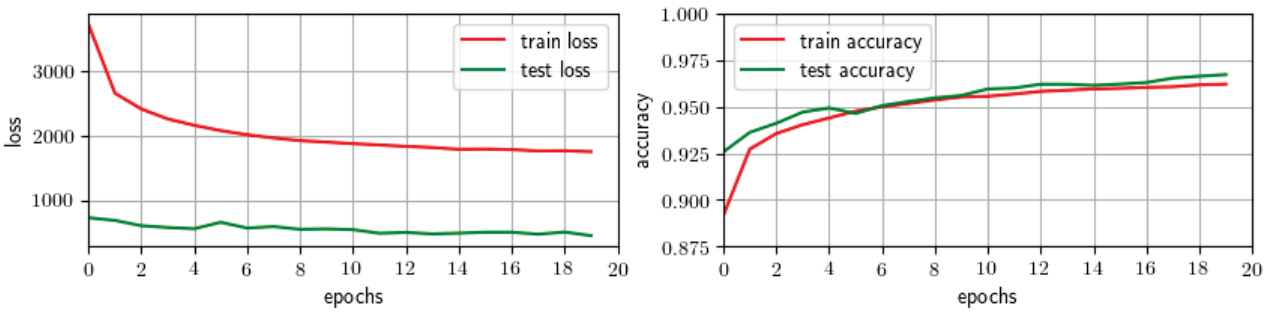
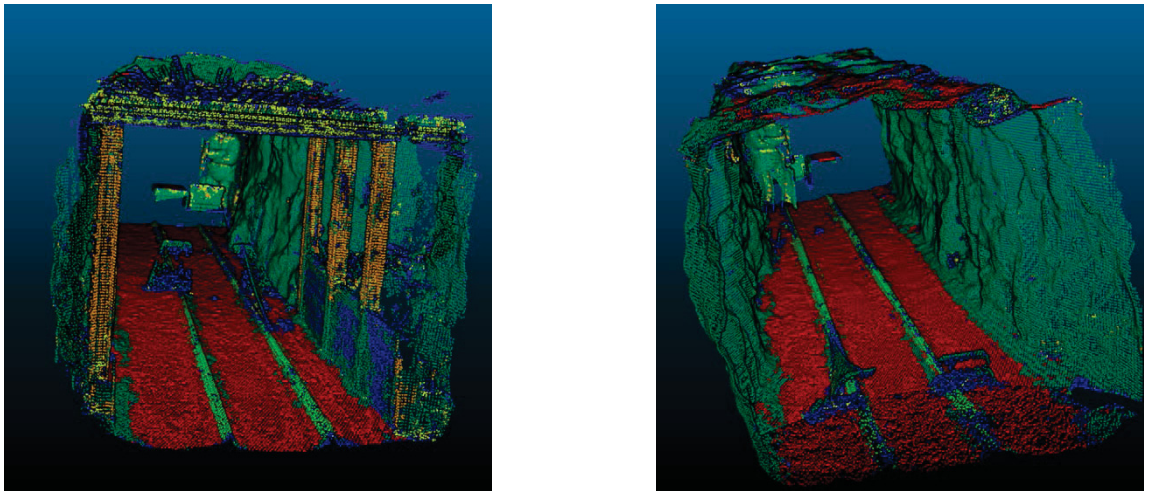


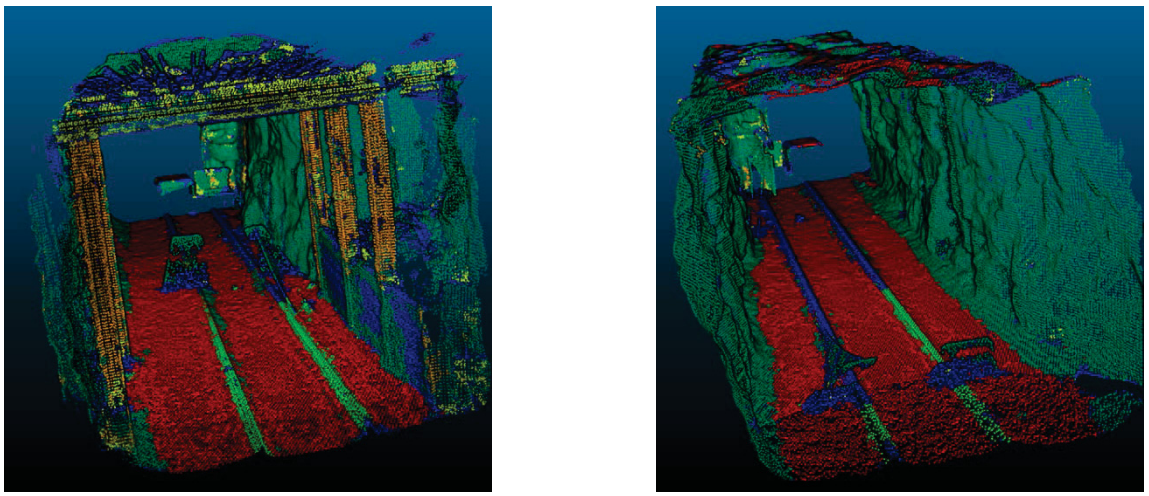
Abb. 7: Verlauf des Trainings des FNN. Während der 20 Trainingsepochen sank die Kostenfunktion für Trainings- und Testdaten kontinuierlich (links). Die Genauigkeit erreicht Werte von über 95% (rechts).

3 Klassifizierte Punktwolken

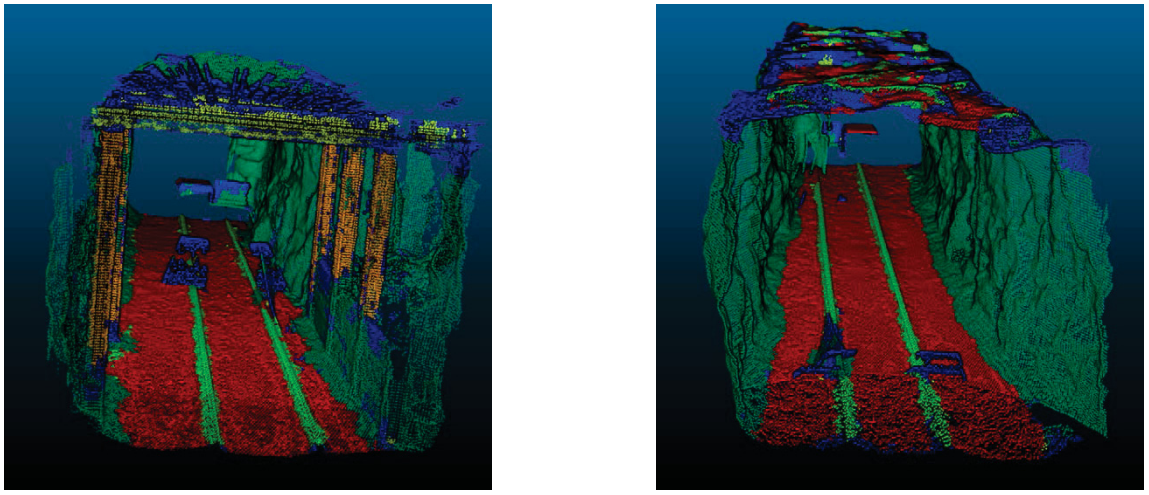
Die Präsentation und Diskussion der Ergebnisse erfolgt auf einem visuellen Level anhand der klassifizierten Punktwolken in Abb. 8.



a) Features auf zwei Skalen (2 cm, 6 cm)



(b) Features auf drei Skalen (2 cm, 6 cm und 10 cm)



(c) 3D - Faltendes Neuronales Netz

Abb. 8: Klassifizierungsergebnisse mit Multi-Skalen-Features in (a), (b) und mit 3D - Faltendem Neuronalem Netz in (c).

In Abb. 8a und 8b sind je zwei Szenen eines untertägigen Streckenabschnitts des Lehr- und Forschungsbergwerks "Reiche Zeche" der TU Bergakademie Freiberg dargestellt, welche mit dem Ansatz der Multi-Skalen-Feature-Extraktion klassifiziert wurden. Der mittlere Punktabstand der 3D-Punktwolke beträgt etwa 2 cm. Die zugeordneten Label sind farblich kodiert: rot - Boden, dunkelgrün - Stoß, hellgrün - Schiene, orange - Stempel, gelb - Stütze und blau - Sonstiges. In Abb. 8a wurden zwei Längenskalen zu 2 cm und 6 cm betrachtet, in 8b kam eine weitere Skala von 10 cm hinzu. Insgesamt wurden die Punkte der Szenen zum größten Teil korrekt klassifiziert, insbesondere die großskaligen Klassen Boden und Stoß sind nahezu vollständig korrekt gelabelt. Kleinteiligere Elemente wie Schiene, Stempel und Stütze wurden korrekt identifiziert, weisen aber eine einem Rauschen ähnliche Fehlklassifizierung einzelner Punkte und Punktgruppen auf. Auffällig und entgegen der Erwartung, verbessert die Einführung der dritten Längenskala in Abb. 8b die Ergebnisse nicht. Im Gegenteil, die Identifizierung der kleinskaligen Schiene in Abb. 8b rechts, scheint durch die Zusatzinformation der großen Längenskala von 10 cm eher gestört und führt zu einer ausgeprägteren Fehlklassifizierung.

Zum Vergleich finden sich in Abb. 8c die Ergebnisse der Klassifikation durch das FNN unter Verwendung eines Voxel-Grids mit einer Kantenlänge eines einzelnen Voxels von 3.125 cm. Groß- und kleinskalige Elemente wurden nahezu vollständig korrekt gelabelt, speziell die rauschartige Fehlklassifizierung von einzelnen Punkten innerhalb eines Elements hat deutlich abgenommen. Das heißt, die hohen Genauigkeitswerte aus Abschnitt 2.3 spiegeln durchaus die Performance des trainierten FNN wieder.

Insgesamt funktionieren beide Ansätze hinreichend gut, um sie in einer Anwendung zur Klassifikation von technischen Einbauten unter Tage einsetzen zu können. Dabei schneidet das FNN in der Klassifizierungsgenauigkeit und Performance etwas besser ab und wäre die Wahl für eine Implementierung in einer Anwendung.

4 Ausblick

Folgende Schritte könnten in der weiteren Optimierung der Parameter des FNN durch cross validation oder der Erhöhung der Code-Performance bestehen.

Conditional Random Fields können unter Einbeziehung der Label benachbarter Punkte für eine glattere Klassifizierung sorgen und die Genauigkeit nochmals etwas (1-3 %) erhöhen.

Reizvoll ist auch die Möglichkeit, zusätzliche Informationskanäle der Punktwolke zur Klassifizierung zu nutzen, z.B. RGB Farbkanäle und hyperspektrale Daten.

Denkbar wären Ansätze des unüberwachten Lernens: Restricted Boltzmann Machines und Deep Belief Networks könnten zur Extraktion von Features eingesetzt werden, welche durch ein Recurrent Net kombiniert werden.

PointNet [13], ein weiterer Ansatz, realisierte ein NN zur Klassifizierung, welches die Vorbereitung der Daten mit einem Voxelgrid obsolet macht und direkt ungeordnete Daten als Input akzeptiert.

5 Förderhinweis

Das Verbundforschungsprojekt "Untertägiges 4D+ Positionierungs-, Navigations- und Mapping-System zur hochselektiven, effizienten und im höchsten Maße sicheren Gewinnung wirtschaftsstrategischer Rohstoffe" der BMBF-Bekanntmachung "R4 - Innovative Technologien für Ressourceneffizienz - Forschung zur Bereitstellung wirtschaftsstrategischer Rohstoffe" wird durch das Bundesministerium für Bildung und Forschung gefördert. Förderkennzeichen: 033R126.

LITERATUR

[1] UPNS4D+ → https://www.r4-innovation.de/de/upns4d.html.

[2] N. Brodu; D. Lague: 3d terrestrial lidar data classification of complex natural scenes using a multi scale dimension-ality criterion:Applications in geomorphology. *ISPRS Journal of Photogrammetry and Remote Sensing*, 68:121 - 134, 2012.

[3] Timo Hackel; Jan D Wegner; Konrad Schindler: Fast semantic segmentation of 3d poinclouds with strongly varying density. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 3(3), 2016.

[4] Martin Weinmann; Boris Jutzi; and Clément Mallet: Feature relevance assessment for the semantic interpretation of 3d point cloud data. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 5:W2, 2013.

[5] Martin Weinmann; Boris Jutzi; Stefan Hinz; Clément Mallet: Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. *ISPRS Journal of Photogrammetry and Remote Sensing*, 105:286-304, 2015.

[6] Leo Breiman: Random forests. *Machine Learning*, 45(1):5-32, 2001.

[7] Timo Hackel; Nikolay Savinov; Lubor Ladicky; Jan Dirk Wegner; Konrad Schindler; Marc Pollefeys: Seman-tic3d.net: A new large-scale point cloud classification benchmark. *CoRR*, abs/1704.03847, 2017.

[8] Jing Huang; Suya You: Point cloud labeling using 3d convolutional neural network. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2670-2675, 2016.

[9] Daniel Maturana; Sebastian Scherer: Voxnet: A 3d convolutional neural network for real-time object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Pittsburgh, PA, 2015.

[10] Radu Bogdan Rusu; Steve Cousins: 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 2011.

[11] Sander Dieleman; Jan Schlüter; Colin Raffel; Eben Olson; Søren Kaae Sønderby; Daniel Nouri; Daniel Maturana; Martin Thoma; Eric Battenberg; Jack Kelly; Jeffrey De Fauw; Michael Heilman; Diogo Moitinho de Almeida; Brian McFee; Hendrik Weideman; Gábor Takács; Peter de Rivaz; Jon Crall; Gregory Sanders; Kashif Rasul; Cong Liu; Geof-frey French; Jonas Degrave: Lasagne: First release. 2015.

[12] Theano Development Team: Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, 2016.

[13] Charles R Qi; Hao Su; Kaichun Mo; Leonidas J Guibas: Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv*:1612.00593, 2016.