OResNet: Ein flexibles ResNet für Octrees

Stefan Schütte, Torsten Bertram

Lehrstuhl für Regelungssystemtechnik, Technische Universität Dortmund
Otto-Hahn-Str. 8, 44227 Dortmund
E-Mail: {stefan.schuette,torsten.bertram}@tu-dortmund.de

1 Einführung

Die lernbasierte Perzeption einer dreidimensionalen Umgebung wird nach wie vor von einer oft speicherintensiven Datenrepräsentation gehemmt. Dies beschränkt die Anwendung von Methoden des maschinellen Lernens, wie Convolutional Neural Networks (CNNs) im dreidimensionalen Raum auf kleine Anwendungsräume, geringe räumliche Auflösungen oder leistungsfähige Rechenhardware. Für übliche dreidimensionale Daten werden bei dicht besetzten Voxel-Repräsentationen große Speicherbereiche für nicht belegte Voxel aufgewendet. Alternativ können Punktwolkendaten direkt vom Netz verarbeitet werden. Hierbei berücksichtigt jedoch die Netzstruktur den räumlichen Zusammenhang der Eingabedaten nicht mehr explizit. Spärlich besetzte dreidimensionale Tensoren werden zunehmend für die Aufgabe der räumlichen Segmentierung eingesetzt. In diesen Strukturen werden Operatoren wie die aus CNNs bekannte Faltung nur auf besetzte Voxel angewendet, während bei allen unbesetzten Voxeln angenommen wird, dass alle Merkmale 0 sind. Dieses Vorgehen reduziert die Anzahl an notwendigen Operationen sowie den Speicherbedarf der Datenstruktur. Die große räumliche Ausdehnung des resultierenden Tensors beschränkt das *Receptive Field* des Netzes für eine gegebene Tiefe und Kernelgröße. Die Struktur eines Octrees kann für die weitere Datenverarbeitung nützlich sein, da Operationen wie Kollisionsprüfungen in Octrees effizient durchgeführt werden können. In der Robotik wird das OctoMap-Format [4] für verschiedene Aufgaben in statischen dreidimensionalen Umgebungen verwendet. Ein neuronales Netz, das die Belegung und semantische Klassen einzelner

DOI: 10.58895/ksp/1000151141-13 erschienen in:
Proceedings – 32. Workshop Computational Intelligence: Berlin, 1. - 2. Dezember 2022
DOI: 10.58895/ksp/1000151141 | https://www.ksp.kit.edu/site/books/m/10.58895/ksp/1000151141/

Voxel in einem Octree bestimmt, kann somit als Vorverarbeitungsschritt für die Kartenerstellung gesehen werden. Die zentrale Idee dieses Beitrags ist daher, ein flexibles Netz für die Segmentierung dreidimensionaler Daten zu entwickeln, das die räumliche Struktur der Eingangsdaten nutzen kann.

2 Verwandte Arbeiten

Für Bildsegmentierung werden häufig Architekturen wie U-Net [9] verwendet, da sie Bilder effizient pixelweise segmentieren können, indem skalierte Merkmalskarten als Zwischenrepräsentationen genutzt und fein aufgelöste Merkmale über *Skip Connections* auf die Ausgangsseite übertragen werden. Durch die begrenzte Größe üblicher Bilddaten können diese Operationen im zweidimensionalen Raum auf dicht besetzten Zwischenrepräsentationen durchgeführt werden. Dies lässt sich nicht direkt auf den dreidimensionalen Raum übertragen. Hier muss entweder die Auflösung begrenzt oder eine spärlich besetzte Darstellungsform gewählt werden, um den Speicherbedarf zu reduzieren. Beispiele für solche spärlich besetzten Repräsentationen bilden Methoden wie PointNet [8] und davon abgeleitete Architekturen [15]. Alternativ können dreidimensionale CNNs auf spärlich besetzte Tensoren angewendet werden, die die Eingabedaten in einer strukturierten Form darstellen. Dieses Vorgehen wird von Ansätzen wie Minkowski U-Net [2] oder SPVNAS [11] verwendet. Beide Ansätze bauen im Netz eine hierarchische Struktur auf, die jedoch für die Ein- und Ausgabe nicht verwendet wird. Methoden wie O-CNN [14] nutzen dagegen explizit eine hierarchische Struktur, den namensgebenden Octree, sowohl in der Eingabe als auch der Ausgabe. Die Eingabe wird auch hier durch eine U-Net-Struktur geführt. Eine entscheidende Einschränkung ist die erzwungene Verwendung der Eingabestruktur für die Ausgabe. Dies kann bei einer Segmentierung zu einem erhöhten Speicherbedarf führen, wenn größere Bereiche in der Ausgabe zur selben Klasse gehören. Weiterhin müssen auch alle Berechnungen bis zum Knoten $\mathbf{c}_{d_{\max},x,y,z}$ auf der untersten Ebene durchgeführt werden, egal ob eine frühzeitigere Klassifikation bereits möglich wäre. So werden mehr Berechnungen als zwingend notwendig durchgeführt. Daher bietet es sich an, eine Struktur wie O-CNN [14] zu erweitern, sodass eine flexible Prädiktion der Ausgabestruktur ermöglicht wird. Dieser Beitrag

untersucht die Güte eines kombinierten Netzes zur Segmentierung und Formrekonstruktion für eine Punktwolken-Segmentierungsaufgabe.

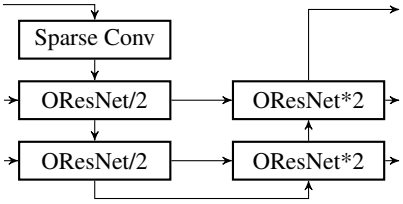
3 Ein flexibles ResNet für Octrees

Residual Neural Networks (ResNets) [29] werden für unterschiedliche Aufgaben in der Bildverarbeitung eingesetzt. Neuronale Netze werden auch in der Literatur auf Octrees angewendet [14]. Die Ausgabestruktur ist dabei jedoch häufig durch die Eingabedaten vorgegeben. Dadurch können Probleme bei der Definition von Kostentermen umgangen werden, die Flexibilität des Netzes, beliebige Strukturen in der Ausgabe darzustellen geht jedoch verloren. Im Gegensatz zu [14] wird bei dem in dem vorliegenden Beitrag beschriebenen Modell keine gemeinsame räumliche Struktur von Ein- und Ausgabe erzwungen. Stattdessen wird die Struktur, inspiriert von [12], gemeinsam mit der semantischen Information aus den Trainingsdaten gelernt und generalisiert. Das zur gleichzeitigen Inferenz von Struktur und Semantik verwendete Modell wird in Abbildung 1a dargestellt.

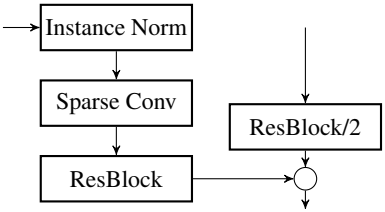
Zusätzlich zur Klassenausgabe für jede aktive Zelle auf jeder Ebene prädiziert der OResNet *2-Block eine Aktivitätsfunktion $a(\mathbf{c})$, die die Belegung der Zelle $\mathbf{c}$ beschreibt. $a(\mathbf{c})]$ besteht aus einem ResBlock und einer Sparse-Conv-Schicht, die jeweils einen 1×1-Kernel besitzen, um die räumliche Struktur des Tensors $\mathbf{C}$ nicht zu verändern.

3.1 Lernen von Segmentierung in hierarchischen Strukturen

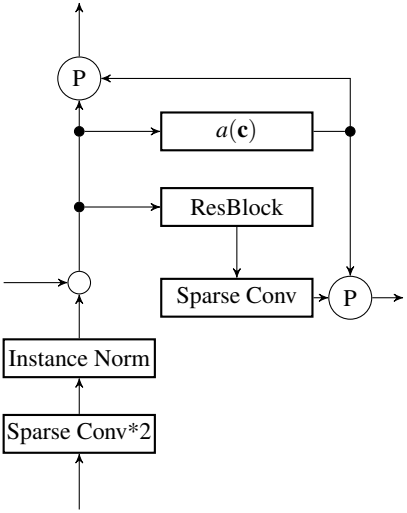
Da die hierarchische Struktur von Octrees sich mit dem Besetzungszustand von Zellen auf höheren Ebenen ändern kann, stellt das Lernen einer Segmentierung auf derartigen Daten ein Problem für die Formulierung üblicher Kostenterme dar. Sollte das Netz eine Zelle als belegt bewerten, die in der Ground Truth nicht als belegt festgelegt wird, kann die Cross Entropy hierfür keine Kosten bestimmen und damit auch keinen Gradienten, der den Fehler des Netzes in



(a) Gesamtstruktur des Netzes. Die OResNet-*2- und -/2-Blöcke führen räumliches Upsampling und Downsampling mittels Faltungsschichten mit Schrittweite und transponierten Faltungen durch. Für tiefere Octrees kann die Struktur vertikal beliebig erweitert werden.



(b) Struktur des OResNet/2 Encoder Blocks



(c) Struktur des OResNet*2 Decoder Blocks. *P* bezeichnet *Pruning*, das Entfernen von ausgewählten Merkmalsvektoren aus dem Tensor.

Bild 1: Struktur des verwendeten Netzes

dieser Zelle verringern würde. Umgekehrt wird auch bei einer als leer bewerteten Zelle, die in der Ground Truth eine Klasseninformation enthält, ebenso kein geeigneter Gradient bestimmt, da der Berechnungsgraph des Netzes eine Unterbrechung aufweist. In den hier gezeigten Experimenten wird für diese Fälle ein zusätzlicher Kostenterm hinzugefügt, der die Struktur des prädizierten Octrees explizit berücksichtig. Um weiterhin während des Trainings immer ein strukturell mit der Ground Truth übereinstimmendes Ergebnis zu erhalten, wird vom Netz zwar eine Strukturprädiktion vorgenommen, statt dieser jedoch die Struktur der Ground Truth auf den Ausgabe-Octree angewendet. Dies ist auch aus einem weiteren Grund notwendig: Das Netz kann, insbesondere am Anfang des Trainings, die Belegung des Ausgabe-Octrees überbewerten und dadurch einen stark erhöhten Speicherbedarf aufweisen, solange die Strukturprädiktion nicht konvergiert ist. Als Optimierer wird Adam [5] mit entkoppeltem Gewichtsverfall [6] als Regularisierungsmechanismus verwendet. Zusätzlich

findet auch Dropout [10] Anwendung. Aufgrund der spärlichen und unter Umständen stark abweichenden Besetzung der im Netz verwendeten Tensoren wird für die Normalisierung auf eine Instance Normalization [13] zurückgegriffen. So wird der Normalisierungsterm nicht stark von einzelnen Elementen des Minibatches beeinflusst.

3.2 Erzeugung des Eingabe-Octrees

Die Eingabestruktur wird aus einer dreidimensionalen Lidar-Punktwolke erzeugt, indem Zellen, die mehr als einen Punkt enthalten, weiter unterteilt werden, bis nur noch ein Punkt in einer Zelle liegt oder die maximale Tiefe des Octrees $d_{\max}$ erreicht ist. Für die Ground Truth wird die gleiche Methode angewendet, der Octree wird anschließend jedoch gekürzt. Unterteilte Octree-Zellen, die nur Punkte der gleichen Klasse enthalten, werden zusammengeführt und die neu entstandene Blattzelle erhält die gemeinsame Klasse. Dies kann die benötigte Rechenzeit des Netzes reduzieren, wenn große Bereiche der Umgebung zur gleichen Klasse gehören. Die Netzschichten der tieferen Octree-Ebenen können so die Klassifikation von typischerweise kleineren Klassen übernehmen. $d_{\max} = 12$ wird für alle Experimente in diesem Beitrag verwendet, was zu einer maximalen Voxelanzahl von $\left(2^{12}\right)^3 = 68.719.476.736$. Mit einer zusätzlichen Merkmalsdimension im Tensor, die eine Mindestgröße von 10 oder mehr Merkmalen enthält und unter Annahme von 16-Bit-Gleitkommazahlen, die in der Netzarchitektur Verwendung finden, sind dicht besetzte Tensorrepräsentationen auf aktueller Hardware nicht realisierbar. Der Speicherbedarf für die beschriebene Repräsentation läge hier bei über 80 GiB. Daher werden im Eingabe-Octree nur Voxel berücksichtigt, die belegt sind. Die Voxelanzahl auf der tiefsten Ebene des Octrees liegt mit etwa 10^4 in einer auf aktueller Hardware verwendbaren Größenordnung.

3.3 Vorgehen beim Training

Da die prädizierte Struktur des Ausgabetensors sich von der Ground Truth unterscheiden kann, muss dies von der Kostenfunktion berücksichtigt werden.

Tabelle 1: Trainingsparameter

Parameter	Lernrate	Gewichtsverfall	Batchgröße
Wert	0.001	0.0001	12

Zu diesem Zweck werden zwei Kostenterme zum Klassifikationsterm hinzugefügt:

$$\mathscr{L} = \mathscr{L}_{\text{class}} + \mathscr{L}_{\text{active}} + \mathscr{L}_{\text{inactive}} \tag{1}$$

$$\mathscr{L}_{\text{class}} = -\sum_{d=1}^{d_{\max}} \frac{1}{\left|\mathbf{C}_{\text{class},d}\right|} \sum_{\mathbf{c}_{x,y,z} \in \mathbf{C}_{\text{class},d}} \sum_{c_{x,y,z} \in \mathbf{c}_{x,y,z}} \log(c_{x,y,z}) \cdot y^{\star}_{x,y,z} \tag{1a}$$

$$\mathscr{L}_{\text{active}} = \sum_{d=1}^{d_{\max}} \sum_{c \in \mathbf{C}_{\text{pred}} \setminus \mathbf{C}_{\text{gt}}} a(\mathbf{c}_d) \tag{1b}$$

$$\mathscr{L}_{\text{inactive}} = -\sum_{d=1}^{d_{\max}} \sum_{c \in \mathbf{C}_{\text{out}}} a(\mathbf{c}_d) \tag{1c}$$

$\mathscr{L}_{\text{class}}$ ist der übliche Cross-Entropy-Kostenterm, der hier jedoch nur auf Blattknoten des Ground-Truth-Octrees angewendet wird. $\mathscr{L}_{\text{active}}$ bildet die Summe der Aktivitätsfunktion ($a(\mathbf{c}_d)$) von jeder Zelle, die vom Netz als aktiv bewertet wurde, aber nicht zur Ground Truth gehört. Im Gegensatz dazu bestehen die Inaktivitätskosten $\mathscr{L}_{\text{inactive}}$ aus der Summe der Aktivitätsausgabe von jeder Zelle, die in der Ground Truth aktiv ist, unabhängig von ihrem Zustand in der Prädiktion des Netzes. Diese Kostenterme bestrafen somit Strukturunterschiede zur Ground Truth.

Während der Validierung in den ersten Trainingsepochen bleibt die vorgegebene Ground-Truth-Struktur in Verwendung, da vor Konvergenz der Strukturprädiktion die Anzahl an aktiven Zellen zu rapide anwachsendem Speicherbedarf führen kann.

Für die Implementierung des Netzes werden Pytorch [7] und Minkowski Engine [2] verwendet. Tabelle 1 stellt die verwendeten Trainingsparameter dar. Die hierarchische Struktur des Octrees $O = (\mathbf{C}_1, \ldots, \mathbf{C}_{d_{\max}})$ mit der maximalen Tie-

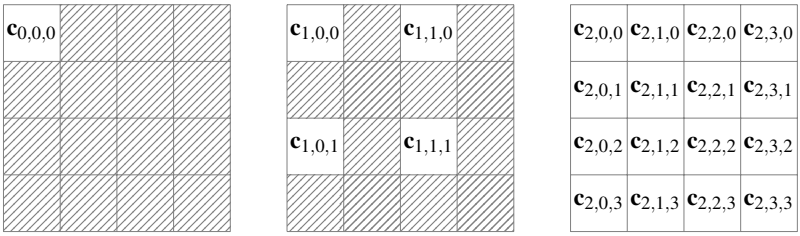


Bild 2: Struktur des Minkowski-Engine-Tensor-Tupels zur Darstellung eines Quadtrees der Tiefe 3. Schraffierte Zellen werden aufgrund der Schrittweite des betreffenden Tensors niemals belegt.

fe $d_{\max}$ wird als $d_{\max}$-Tupel von spärlich besetzten vierdimensionalen Tensoren $\mathbf{C}_d \in \mathbb{R}^{X\times Y\times Z\times F_d}$ mit $F_d = \min(2^{d_{\max}-d}, 256)$ dargestellt. Der in der Minkowski Engine üblichen Vorgehensweise entsprechend werden die räumlichen Dimensionen als spärlich besetzt behandelt, während der Merkmalsvektor an den besetzten Stellen eine dichte Darstellung besitzt. Die Darstellung des Tensors $\mathbf{C}_d$ im Speicher ist daher ein Tupel der Koordinaten der n aktiven Zellen $\mathbf{K} \in \mathbb{R}^{n\times 4}$ und der zugehörigen Merkmalsvektoren $\mathbf{F}^{n\times F_d}$. Die einzelnen spärlich besetzten Tensoren der Hierarchieebenen teilen sich in der hier verwendeten Implementierung ein gemeinsames diskretisiertes Koordinatensystem, besitzen jedoch abhängig von der Hierarchieebene unterschiedliche Schrittweiten. Für jede höhere Hierarchieebene wird die Schrittweite s des Tensors um den Faktor zwei erhöht. Abbildung 2 zeigt die Struktur der Tensoren am Beispiel eines Quadtrees. Für die dreidimensionalen Faltungsschichten gelten die gleichen Beschränkungen, was effektiv zu einer Dilatation der Kernels führt. Eine Faltungsschicht, die auf einen Tensor mit einer Schrittweite $s_{\mathrm{T,in}} \neq 1$ angewendet wird, muss nicht zwangsweise eine Schrittweite $s_{\mathrm{conv}} \neq 1$ aufweisen, verändert die Schrittweise des Tensors jedoch entsprechend $s_{\mathrm{T,out}} = s_{\mathrm{T,in}} \cdot s_{\mathrm{conv}}$. So entsteht ein hierarchisches Netz, das den in der Bildverarbeitung üblichen U-Nets [9] ähnelt. Von der Architektur in [2] unterscheidet sich das Netz durch die Möglichkeit der Ausgabe von semantischen Informationen auf höheren Hierarchieebenen. Das so entstehende Octree ResNet (OResNet) kann als Kombination von *Shape Reconstruction* [12] und semantischer Segmentierung verstanden werden. Es kann für zusätzliche Aufgabenfelder erweitert werden.

4 Ergebnisse der Lidar-Segmentierung

Um die Güte des beschriebenen Modells zu bewerten, wird eine Auswertung bei einer Punktwolkensegmentierung vorgenommen. Der nuScenes-Lidarseg-Datensatz [1] ermöglicht die Untersuchung des Netzes auf Realdaten. Der Datensatz enthält Punktwolken von Straßenszenen, die mit einem Lidarsensor auf dem Dach eines Fahrzeugs aufgenommen wurden. Die Anwendung eines Netzes, das Octree-Strukturen segmentiert, bildet die Aufgabe der dreidimensionalen Kartierung einer Umgebung ab. Die Güte des Netzes wird, um die für nuScenes-Lidarseg übliche Metrik der *Intersection over Union* (IoU) anwenden zu können, durch anschließende Projektion der Eingabepunktwolke in den Ausgabe-Octree bestimmt. Insgesamt werden die Punktwolken in 32 Klassen eingeteilt, da aber von einigen Klassen nur wenige Beispiele vorhanden sind, werden diese Klassen für die Auswertung der Lidarseg-Challenge entsprechend zu 16 Klassen zusammengeführt. Um die vom Netz prädizierten voxelgenauen Klassen auf die Punktwolke abzubilden, wird diese in den Octree projiziert. Die Klassenprädiktion für einen einzelnen Punkt entspricht dabei der Klasse, die das Netz für den Blattknoten, in dem dieser Punkt liegt, bestimmt hat. Die Klassifikationsergebnisse werden dann auf dem nuScenes LidarSeg-Validierungsdatensatz bewertet, da für das Testset keine Ground Truth vorliegt. Tabelle 2 zeigt die Ergebnisse der punktweisen Segmentierung auf dem nuScenes LidarSeg-Datensatz für aktive und inaktive Strukturinferenz.

Hier zeigt sich ein deutlicher Einfluss der Strukturinferenz durch das Netz auf das Endergebnis der Segmentierung. Die integrierte Strukturschätzung zeigt im Vergleich zur Vorgabe der räumlichen Struktur der Ground Truth verringerte Werte für die IoU. Die räumliche Ausdehnung der zu detektierenden Klasse beeinflusst die Detektionsgüte ebenfalls. Klassen, die üblicherweise ein größeres Volumen einnehmen, werden vom Netz zuverlässiger korrekt klassifiziert. Für einzelne Klassen von kleineren Objekten wie der Klasse *Motorcycle* zeigt sich dagegen eine verbesserte Prädiktionsgüte bei der Strukturprädiktion. Dies deutet auf Vorteile durch die Verwendung der weiteren dem Netz zur Verfügung stehenden Faltungsschichten für die Klassifikation hin.

Tabelle 2: Intersection over Union für die unterschiedlichen Klassen des nuScenes-LidarSeg-Datensatzes

Klasse	Inferenzmethode	
	GT-Struktur	Strukturinferenz
Barrier	0,45	0,31
Bicycle	0	0
Bus	0,24	0,19
Car	0,49	0,45
Constr. Veh.	0,04	0,04
Motorcycle	0,06	0,12
Pedestrian	0,08	0,08
Traffic Cone	0,16	0,12
Trailer	0,23	0,24
Truck	0,36	0,25
Driv. Surf.	0,91	0,75
Flat	0,60	0,43
Sidewalk	0,59	0,47
Terrain	0,62	0,51
Manmade	0,74	0,63
Vegetation	0,72	0,71
mIoU	0,39	0,33

5 Zusammenfassung

Dieser Beitrag stellt eine Architektur für ein neuronales Netz vor, das durch Kombination von semantischer Segmentierung und Shape Reconstruction im Decoder einen Octree mit semantischen Informationen prädizieren kann. Die hierarchische Struktur der Eingabe- und Ausgabedaten enthält zwar nützliche Informationen für nachfolgende Aufgaben, die Erstellung der Eingabestruktur ist jedoch mit einem hohen Berechnungsaufwand verbunden. Dies bedeutet, dass das Verfahren für Online-Aufgaben nur eingeschränkt geeignet ist. Das muss jedoch keine Beschränkung bei Offline-Aufgaben wie der Erstellung dreidimensionaler semantischer Karten von großen Szenen sein. Da das Netz mit dem Ziel trainiert wird, eine möglichst effiziente Repräsentation

der Ausgabe zu generieren, muss hier der erhöhte Rechenaufwand bei der Vorverarbeitung mit den Vorteilen dieser Ausgabeform abgewogen werden.

Darüber hinaus bringt die hierarchische Struktur des Netzes einige Einschränkungen mit sich. Das Netz führt zwangsläufig ein Downsampling bis zur Wurzel des Octrees durch, was für die semantische Segmentierung nicht benötigt wird. Diese Berechnungen müssen jedoch durchgeführt werden, um eine Einschränkung des effektiven Receptive Field durch die üblichen 3×3-Kernelgröße der Residual Layer zu vermeiden.

Da das Netz häufige Klassen zuverlässiger bestimmen kann als seltene, könnten Trainingsmethoden wie CutMix [16] auf diesen Ansatz angewendet werden. Um CutMix sinnvoll auf Octrees anwenden zu können, müsste die Methode auf die Octree-Struktur angepasst werden, da ein wiederholtes Berechnen des Octrees während des Trainings zu einer deutlichen Verlängerung der Trainingsdauer führen würde.

Literatur

[1] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan und O. Beijbom. „nuscenes: A multimodal dataset for autonomous driving“.

[2] C. Choy, J. Gwak und S. Savarese. „ 4d spatio-temporal convnets: Minkowski convolutional neural networks“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019.

[3] K. He, X. Zhang, S. Ren und J. Sun. „Deep residual learning for image recognition“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. 2016.

[4] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss und W. Burgard. „OctoMap: an efficient probabilistic 3d mapping framework based on octrees“. 34(3):189–206.

[5] D. P. Kingma and J. Ba. „Adam: A method for stochastic optimization“.

[6] I. Loshchilov and F. Hutter. „ Decoupled weight decay regularization“. In: *International Conference on Learning Representations*. 2019.

[7] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai und S. Chintala. „Pytorch: An imperative style, high-performance deep learning library“. In: H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox und R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc. 2019.

[8] C. R. Qi, H. Su, K. Mo und L. J. Guibas. „Pointnet: Deep learning on point sets for 3d classification and segmentation“.

[9] O. Ronneberger, P. Fischer und T. Brox. „U-net: Convolutional networks for biomedical image segmentation“. In: *Lecture Notes in Computer Science*, pages 234–241. Springer International Publishing. 2015

[10] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever und R. Salakhutdinov. „Dropout: a simple way to prevent neural networks from overfitting“. *The journal of machine learning research*, 15(1):1929–1958. 2014.

[11] H. Tang, Z. Liu, S. Zhao, Y. Lin, J. Lin, H. Wang und S. Han. „Searching efficient 3d architectures with sparse point-voxel convolution“. In: *European Conference on Computer Vision*. 2020.

[12] M. Tatarchenko, A. Dosovitskiy und T. Brox. „Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs“. In: *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2107–2115. 2017.

[13] D. Ulyanov, A. Vedaldi und V. Lempitsky. „Instance normalization: The missing ingredient for fast stylization“.

[14] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun und X. Tong. „O-cnn: Octree-based convolutional neural networks for 3d shape analysis“.

[15] Y. Xie, J. Tian und X. X. Zhu. „Linking points with labels in 3d: A review of point cloud semantic segmentation“. *IEEE Geoscience and Remote Sensing Magazine*, 8(4):38–59. 2020.

[16] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe und Y. Yoo. „Cutmix: Regularization strategy to train strong classifiers with localizable features“. In: *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032. 2019.