ALS Point Cloud Classification using PointNet++ and KPConv with Prior Knowledge

Martin Kada, Dmitry Kuramin

Technische Universität Berlin, Institute of Geodesy and Geoinformation Science, Berlin, Germany
(martin.kada@tu-berlin.de, kuramin@campus.tu-berlin.de)

KEY WORDS: Point clouds, 3D, Classification, Semantic Segmentation, City Modelling, Deep Learning.

ABSTRACT:

In the practical and professional work of classifying airborne laser scanning (ALS) point clouds, there are nowadays numerous methods and software applications available that are able to separate the points into a few basic categories and do so with a known and consistent quality. Further refinement of the classes then requires either manual or semi-automatic work, or the use of supervised machine learning algorithms. In using supervised machine learning, e.g. Deep Learning neural networks, however, there is a significant chance that they will not maintain the approved quality of an existing classification. In this study, we therefore evaluate the application of two neural networks, PointNet++ and KPConv, and propose to integrate prior knowledge from a pre-existing classification in the form of height above ground and an encoding of the already available labels as additional per-point input features. Our experiments show that such an approach can improve the quality of the 3D classification results by 6% to 10% in mean intersection over union (mIoU) depending on the respective network, but it also cannot completely avoid the aforementioned problems.

1. INTRODUCTION

Semantically interpreted and structured 3D geodata, e.g. in the form of 3D city models, but also the respective raw source data, such as aerial 3D point clouds that are enriched with additional information, are increasingly finding applications, and are as a consequence more and more collected and commissioned by geoinformation authorities and users. A comprehensive overview of applications for 3D city models is given, e.g., in (Biljecki et al., 2015). Whereas only a decade ago, the focus was primarily on distinguishing between buildings, terrain, and tall vegetation (trees), further object types such as facades, low vegetation, vehicles, overhead power lines, and poles are nowadays also of particular interest. For their automatic extraction and geometric reconstruction and modelling, airborne laser scanning (ALS) point cloud data are still the main data sources, and the classification of the collected points the first step in any processing pipeline. Besides deterministic algorithms and classical machine learning methods, neural networks have been increasingly developed and proposed for this purpose with the advent of Deep Learning in recent years.

Quite a number of neural networks have already been presented that perform point-wise classification (also known as semantic segmentation) directly on the 3D point cloud without changing the representation of the data. For these networks, it has been shown that they can lead to impressive results also for large scale ALS data with overall high accuracies of 95% to 98% (Varney et al., 2020). However, the quantitative results hide the fact that these methods also show systematic weaknesses in some areas that can be reliably classified using deterministic methods. Large objects in general, and objects that are not frequently encountered in the training data, as well as an accurate separation between ground and non-ground points, are just a few examples that often lead to problems. Figure 1, e.g., shows a Deep Learning based classification result, where the misclassification of roof points as ground points becomes visible for the large building at the top.

In the presented study, we investigate how prior knowledge from a rough semantic classification of an ALS 3D point cloud can be integrated into 3D point cloud based neural networks to produce a finer grained classification. This prior knowledge could be a classification that is the result of a deterministic algorithm with empirically set thresholds, consisting of fewer than the targeted classes, and having a lower degree of accuracy. The goal is to replace a manual post-processing step that comes after an already established automatic classification, which gives reliable results in form of a few classes with well-known and guaranteed quality properties, with a Deep Learning approach that is trained to keep the labels of already correctly classified points, if possible, and focuses on those classes that need further refinement.

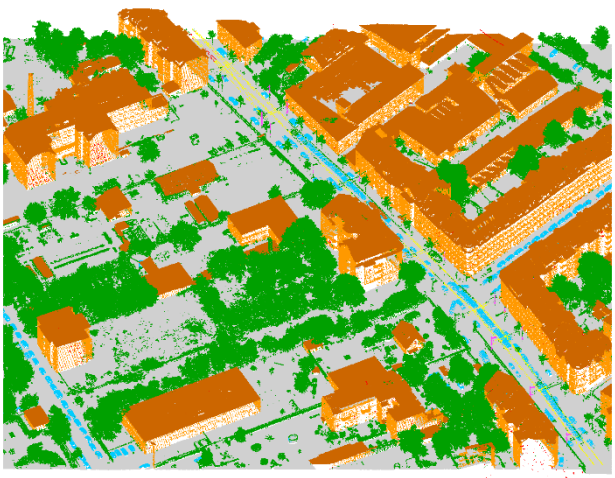

Figure 1. Classification results of an ALS 3D point cloud using KPConv showing misclassifications on large building roofs.

The integration of prior knowledge has already been shown to significantly improve ALS point cloud classification, e.g., using normalized heights as input features as in (Lei et al., 2020), which are calculated from a digital elevation model that is built from interpolated ground points identified by the cloth simulation field algorithm (Zhao et al., 2016). We used two neural networks for our practical research: PointNet++ (Qi et al., 2017b), which is the archetype of 3D point cloud based neural networks, and KPConv (Thomas et al., 2019), which has shown current state of the art

performance. Although PointNet++ is now often outperformed in experiments by more recent networks like KPConv, ConvPoint (Boulch, 2020), or PointCNN (Li et al., 2018), it remains highly relevant, since its feature extraction mechanism and architecture is often a building block for other 3D point cloud based networks, e.g. to solve the task of object detection as described in (Qi et al., 2019). For both above mentioned networks, we study in this paper how the additional input of height above ground as well as the coded pre-classification impact the classification results into further classes.

2. RELATED WORK

Although Deep Learning convolutional neural networks (CNN) have been highly successful for all types of classification and segmentation tasks on images, the transfer of the developed methods and architectures to data without any intrinsic regular structure has proven challenging. Since 3D point clouds are sparse, unordered sets of points with highly variable density, convolutional filters cannot be directly applied on them. Point cloud data have therefore often been brought into other, regular representations, like multi-view images (Boulch et al., 2018), 2D grids (Zhao et al., 2018), 3D voxel grids (Zhou and Tuzel, 2018), octrees (Riegler et al., 2017), 2D/3D lattices (Su et al., 2018), kd-trees (Klokov and Lempitsky, 2017), etc., to which the mentioned convolutional filters can be directly applied or for which they can be defined in a similar way, and with which it is then possible to build Deep Learning architectures that are similar to the well-known frameworks for 2D image data. Many of these have also been successfully applied to the classification of ALS 3D point clouds as in (Yang et al., 2017; Schmohl and Sörgel, 2019).

In recent years, a number of neural network architectures have been proposed that operate directly on the unordered 3D point clouds. PointNet successively transforms the 3D coordinates of each input point into a higher dimensional spatial encoding from which local per-point features, and by their aggregation, global per-object features are determined that are invariant to input permutations (Qi et al., 2017a). These global per-object features that are computed from the point coordinates of small point neighbourhoods perform very similarly to hand-crafted features commonly used in the classification of 3D point clouds, and can also be used for this purpose with a fully connected neural network. In order to capture features at different scales, PointNet++ introduces hierarchical set abstraction layers based on sampling, grouping, and feature extraction, as well as feature propagation layers to upsample these features back to the original points, so that a U-Net like architecture for semantic segmentation is realized (Qi et al., 2017b). Winiwarter et al. (2019), e.g., demonstrate the applicability of PointNet++ for ALS data. Quite a few networks for 3D point cloud data have followed and extended the PointNet++ approach to implement something comparable to image based convolutional filters. PointConv, e.g., computes weights from the aforementioned point-wise spatial encoding that are then multiplied to the combined point densities and the input features of the point cloud (Wu et al., 2019). To overcome the lack of point order that is required for the use of convolutions, PointCNN learns a transformation χ from the spatial encoding – a similar concept as a spatial transformer network – and weights and permutes the point coordinates with this transformation into a canonical order (Li et al., 2018).

Another category of point cloud based Deep Learning networks define kernels that consists of a number of 3D points within a spherical space. These kernels are then overlaid on (a sampled subset of) the input points, and a weighted linear combination of input and kernel point weights are calculated (similar to image convolutions) taking into consideration the distances of input to kernel points. In KPConv, e.g., the weights and the positions of the kernel points are learned (Thomas et al., 2019), and the resulting (deformable) kernel point convolutions can be used for building architectures like the above-mentioned U-Net. Boulch (2020) uses a multi-layer perceptron to learn a continuous weighting function for the convolutional filter operations. For a performance comparison of some of the mentioned networks, see (Varney et al., 2020).

3. METHODOLOGY

In this practical study, we used both PointNet++ and KPConv as the most prominent representatives of the above-mentioned categories of Deep Learning networks for 3D point clouds. The ALS data we performed our experiments with has a rather high point density of about 110 points per m², and this obviously has implications regarding network hyperparameters, such as the number of layers or the specifics of local point neighborhood queries. These details, as well as the method by which prior knowledge, given in the form of a rough pre-classification, is fed into the networks, is described in this section.

3.1 Prior Knowledge as Input Features

In the proposed approach, a pre-existing rough classification of the ALS 3D point cloud data is fed as prior knowledge into the neural networks in the form of two additional point properties: height above ground, and an encoding of the classes. Assuming that the identification of ground points from deterministic algorithms is already pretty accurate, a digital terrain model is constructed from those by triangulation, and a height above ground computed for each point. To provide the prior classes, a vector is produced in a one-hot-encoding, where one value is stored for each class, all of which have a value of zero except for the value representing the actual class, which is given a value of one. For our pre-classified ALS 3D point cloud featuring five classes, e.g., we end up with six additional scalar input values per point, which are then fed into the neural networks along with intensity, number of returns, and return number; besides the 3D coordinates.

Using PointNet++, we also experimented with injecting the one-hot-encoded class priors further towards the end of the network right before the segmentation head. However, despite using an increased number of network layers and units there, this was still a too strong of a prior that the network seemed unable to ignore, and the output classes hardly differed from the input classes.

3.2 PointNet++

The PointNet++ architecture we used follows to a large extent the one described in (Qi et al., 2017b). For feature learning, four point set abstraction layers with multi-resolution grouping are employed, subsampling from their input 8192, 4096, 2048, and 1024 points. For these points, a number of 16, 64, 64 and 32 neighboring points are found in the respective layer using ball queries with radius 0.5m, 1m, 5m, and 15m, from which features are calculated using a multilayer perceptron with an identical setup of 128, 128, and 256 units in all set abstraction layers. Only if 2D features are not used, see sub-section 3.3 for more details, then the number of units in the first layer are halved to 64, 64, and 128 units. In contrast to Winiwarter et al. (2019), we observed a significant improvement when using four set abstraction layers instead of three, which is most likely attributed

to the high point density of the input 3D point cloud and the consequential use of the small ball radius (0.5m) in the first set abstraction layer. The learned point features are gradually interpolated back to the original number of input points by four feature propagation layers, with the first three layers using a multi-layer perceptron with 256 and 256 units, and the last one with 128 and 64 units. The head of the network consists of a dense layer with 128 units, a dropout layer with keep probability of 0.5, and another dense layer with 9 units (equal to the number of output classes). ReLU activations and max-pooling are used wherever appropriate (but not average pooling). For more details on PointNet++ and its application for ALS 3D point cloud classification, please refer to (Qi et al., 2017b) and (Winiwarter et al., 2019).

3.3 PointNet++ with 2D Point Neighborhoods

While working with PointNet++ on ALS data, we noticed that the network has problems when there are a lot of points on top of each other, especially in the areas of facades or high vegetation that reaches over roofs. Our explanation is that it cannot capture the needed vertical spatial context in order to predict the correct classes. PointNet++ is one of the architectures that computes for each input point a number of features from the 3D coordinates of the points that are located in a query neighborhood, which can be seen somewhat similar to the hand-crafted features as described in (Weinmann et al., 2015). In its original implementation, PointNet++ computes its features in a hierarchical way from spherical point neighborhoods, which might not be sufficient for ALS point clouds that have a mainly horizontal extent. Based on these considerations, we added another branch in the first (of the four) set abstraction layers that calculates further features from cylindrical (2D) neighborhoods, so that all points from the lowest (ground) point to the highest (roof, vegetation) point are always taken into consideration. And at the end of the first set abstraction layer, all features from both neighborhoods are concatenated. Because we are now extracting twice as many features, from both 2D and 3D neighborhoods, we doubled the number of multi-layer perceptron units that follow in the first layer. In general, we did not see any considerable improvement when we increased the number of units in a different layer, so that the capacity of the network appears sufficient enough to learn from the given data.

3.4 KPConv

Our experiments with KPConv were performed without any changes to the network architecture, only the hyperparameter were adjusted and the described point properties were used as additional input. The hyperparameters were found using a grid search from a set of values based on previous experiments also conducted on other ALS datasets. The reason is to have a more general set of hyperparameter values that work well on a range of ALS datasets. Therefore, they might not be ideal for datasets with such high point density as described here; especially the cell size for the first subsampling seems too high in retrospect. For our experiments, we used the following hyperparameter values: 0.5m for the cell size of the first subsampling, 15m for the radius of batching spheres, 1m for the convolution radius, 0.6m for the (repulse and kernel point) extent, and 15 for the number of kernel points. The regularization loss for deformable kernels is added to the overall output loss with a multiplicative factor of 0.5.

Since KPConv does not extract features from the 3D point coordinates, but uses the 3D coordinates to relate the input points to the filter points in order to weight the input features, an extension to 2D neighborhoods as described for PointNet++ did not seem meaningful.

4. EXPERIMENTS AND RESULTS

We conducted our experiments on an ALS 3D point cloud dataset with 42.5 million points and a point density of about 110 points per m², collected from an inner-city area of Leipzig, Germany, which includes dense scenes with many large building blocks and close-by objects. Figure 2 shows the dataset. The point cloud was split in 34.8 million points for training and 7.7 million points for testing. Since bridge objects and overhead lines are spatially separated in the dataset, and we wanted to keep training and test areas as compact as possible, with a rectangular extent, and have similar built-up, the test data does unfortunately not include any bridge objects. We made this decision, because bridges did not have such a high practical relevance for the study. However, we still trained with the bridge class.

Besides the described prior knowledge, we included intensity, the number of returns, and the return number as input features to both networks in addition to point coordinates. The prior classification includes the five classes of ground, roof, facade, error and outlier, and vegetation (or other), with the latter including many other objects such as cars, trams, road construction sites, fences, etc. (besides vegetation). The networks are trained to output the classes of vehicles (cars, buses, trucks, trains), overhead lines, (lamp or other) poles, and bridges, in addition to the five input categories. In the process, classes that belong to both input and output are to be improved, and the vegetation class is to be further refined, so that other object types no longer appear within it.

4.1 PointNet++

For all experiments with PointNet++, we generated 651 circular patches with 200,000 points each from the training point cloud as input for the network, so that each point is included in three to four patches on average. We conducted experiments with more points per patch, larger search radii for the neighborhood ball query, and a fifth set abstraction layer, but noticed no significant improvement. For network training, we use the Adam optimizer, a learning rate of 0.001, and reserved 15% of the training patches for validation. Training was performed for 50 epochs using the remaining patches in random order. For testing, we generated 169 patches from the test data, predicted the class probabilities for every patch individually, and merged the predictions as described in (Winiwarter et al., 2019). Although we had good experience in previous experiments with rotating the 3D point cloud patches around the upward (z) axis for data augmentation, we chose not to do so due to already long training times of over 1h per epoch.

Our experience with training PointNet++ with ALS point cloud data is that the network makes quite some jumps in the quality metrics (1% to 3% in overall accuracy) that are used to observe the training process, which is also noticeable in the classification results. We therefore used for predictions the weights resulting from one of the last five training epochs that showed the highest overall accuracy on the validation patches. It is interesting to note that these performance jumps are generally less severe when using input classes as prior knowledge. The performance metrics for the PointNet++ without any prior knowledge (or architectural modifications) and with the described prior knowledge (height above ground and the five input classes) are given in the first part of Table 1. While the overall accuracy only increases by 3%, the mean intersection over union (mIoU) goes up by almost 9%. (Note that we did not include the bridge class in the calculation of mIoU, nor report the IoU for this class, since there are no points in the test data that belong to it.) See also Figure 3 for a visual presentation of the classification results using PointNet++.

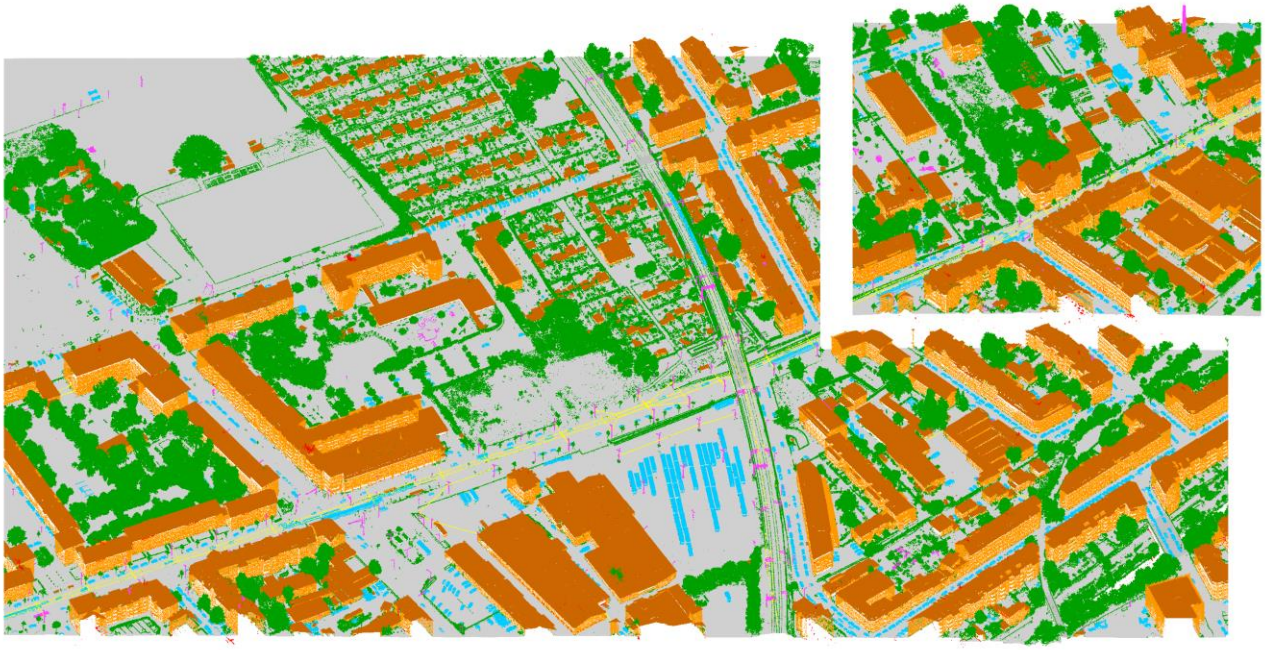

Figure 2. Training and test data of Leipzig, Germany, with the semantic classes: ground (gray), vegetation(green), building roof (brown), building facade (orange), vehicle (light blue), overhead line (yellow), pole (purple), bridge (dark gray), error and outlier (red). The test tile seamlessly adjoins the training data, but is presented with a horizontal offset for better visualization.

	Prior			IoU								
Method	HaG	Cls	OA	*mean*	*ground*	*vegetation*	*roof*	*facade*	*vehicle*	*line*	*pole*	*error*
PointNet++			0.931	0.605	0.919	0.836	0.903	0.703	0.291	0.707	0.170	0.310
	×	×	0.964	0.691	**0.981**	0.919	0.932	0.721	0.432	0.852	0.184	0.506
PointNet++ (+2D)	×		0.935	0.615	0.925	0.837	0.918	0.716	0.135	**0.877**	0.189	0.322
	×	×	**0.968**	**0.702**	0.980	**0.921**	**0.946**	**0.782**	**0.434**	0.814	**0.224**	**0.516**
KPConv			**0.955**	0.624	0.942	0.893	0.945	0.793	0.502	0.577	0.069	0.268
	×		0.949	0.684	0.932	0.891	0.921	0.792	0.498	**0.898**	0.223	0.319
		×	0.954	0.669	**0.948**	0.882	**0.948**	0.773	0.377	0.878	**0.238**	0.309
	×	×	0.953	**0.686**	0.938	**0.898**	0.930	**0.779**	**0.506**	0.878	**0.238**	**0.321**

Table 1. Overall accuracy (OA), mean and class-specific intersection over union (IoU) for the two reported Deep Learning networks of this study without and with height above ground (HaG) and input classes (Cls) as priors. Best scores are highlighted per network.

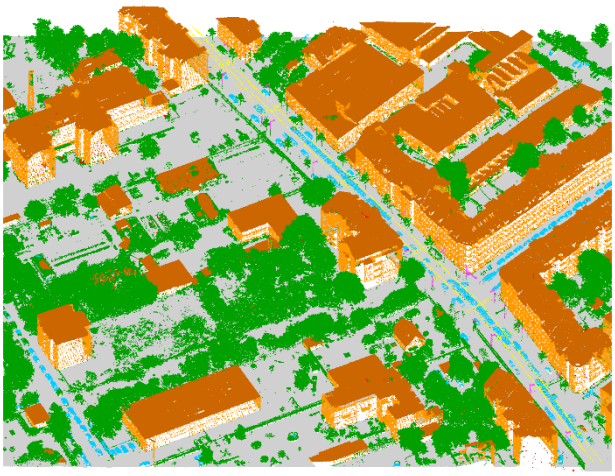

Figure 3. Classification results from PointNet++ using prior knowledge (height above ground and input classes).

4.2 PointNet++ with 2D Point Neighborhoods

To study and demonstrate the effectiveness of extracting point features using their cylindrical (2D) neighborhoods in the first set abstraction layer of PointNet++, we trained the neural network without any priors on the dataset of five classes. The differences can be seen in Figure 4. In general, the edges between building roofs and facades are better defined, vegetation points are more homogeneously assigned to the same class, and vehicles seem more complete (although they are in the same class as vegetation in that dataset). But on the downside, there are misclassification problems where points of larger ground areas are being classified as roof points, which could be the result of outliers located below the ground that are now taken into consideration in the cylindrical neighborhood. Using height above ground as a prior knowledge tends to reduce this problem.

In the quantitative evaluation with the dataset of nine classes given in the second part of Table 1, one also sees an improvement of about 3% in mIoU when going from the original version of PointNet++ to adding 2D features in combination with height above ground. Such a setup could be interesting when ground

points are known, but no further classification is available. The difference going from a setup with both types of prior knowledge (height above ground and input classes) to adding 2D features is not as pronounced and is merely a 1% increase in mIoU. The increase seems to be primarily in the better separation between buildings roofs and facades. This is, however, also accompanied with some IoU changes in the classes of overhead lines and poles.

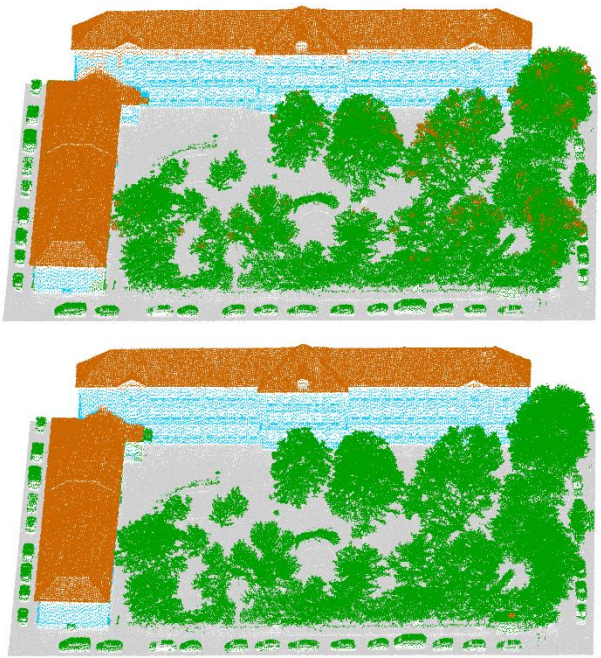

Figure 4. Comparison of classification results between original PointNet++ (top) and PointNet++ using additional cylindrical (2D) point neighborhoods for feature extraction (bottom).

Another important aspect in the study was the separability of points that belong to vehicles and ground points, which improved from the original PointNet++ with the use of prior knowledge and further more by using 2D features (see Figure 5).

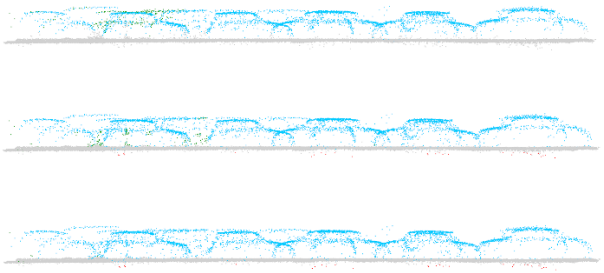

Figure 5. Separability of vehicle and ground points given by PointNet++ without any changes (top), with prior knowledge (middle) and with additional 2D features (bottom)**.**

4.3 KPConv

Since the implementation of KPConv (from the authors) already contains various functionalities, such as subsampling, patching, selection of patches per epoch, merging of prediction results, etc., hardly any preprocessing, data preparation, or modifications had to be performed. We trained the network as described in (Thomas et al., 2019) for 30 epochs using 300 steps per epoch.
KPConv already shows great performance out of the box (cp. Table 1). However, even though the overall accuracy does not change much between experiments, there is again a notable increase in mIoU of around 6% between omitting and using the described priors. Table 1 gives further details how the individual classes improve. When inspecting the point cloud colorized by class labels, however, it becomes obvious that although some classes like vehicles, overhead lines, and poles visually improve, that also points of some larger ground areas are now misclassified as roof, which we cannot explain. However, the systematically occurring gaps on roofs almost no longer appear (see Figure 1 and Figure 6).

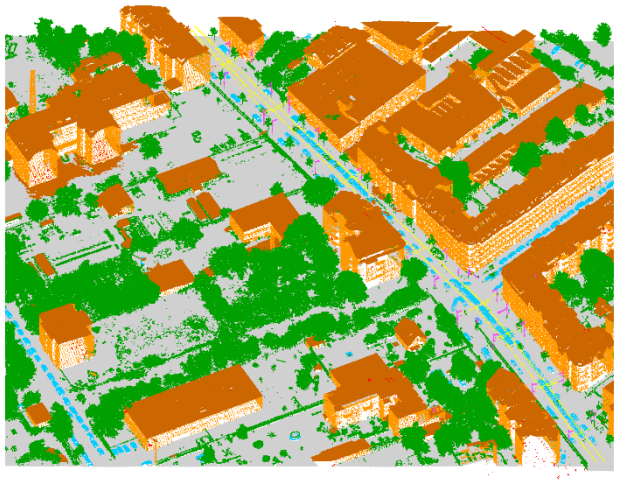

Figure 6. Classification results of an ALS 3D point cloud using KPConv with height above ground and input classes as priors.

A close-up view of the classification results of KPConv is given in Figure 7. Although improvements in KPConv are visible through the use of prior knowledge, we want to be cautious about making a final assessment, as we do not want to rule out the possibility that the more generally determined hyperparameters are not best suited when using KPConv with this dataset.

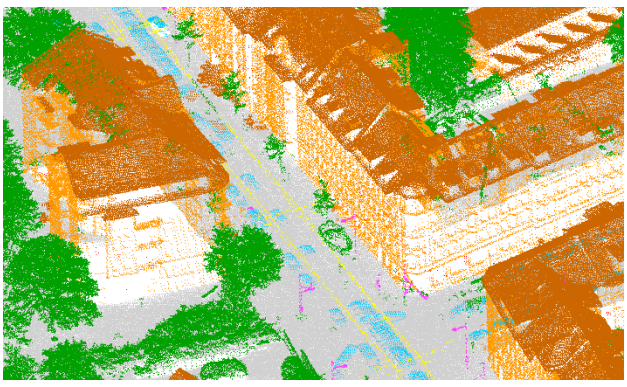

Figure 7. Close-up view of classification results generated with KPConv using prior knowledge.

5. CONCLUSIONS

In this study, we demonstrate that integrating prior knowledge in the form of height above ground and pre-existing class labels can

improve the classification results of ALS 3D point clouds in point cloud based Deep Learning networks, when it comes to refining already available class labels, but also to overcome often seen misclassifications. Our results indicate that for both networks, PointNet++ and KPConv, only the combination of the described two input priors (height above ground and class labels with fewer categories) leads to the best results. Both networks showed fewer issues with misclassifications of roof points for larger buildings, eliminating it almost completely for PointNet++. Although the input of prior knowledge improves the results of KPConv, there still remain areas where the original problems continue to occur, especially noticeable in misclassifications in the middle of roofs on larger buildings, which could be due to the choice of hyperparameters.

In general, PointNet++ benefited much more from the prior knowledge and was able to eliminate its original problems quite reliably. By integrating height above ground, PointNet++ does not confuse ground with building roof or vehicle points as often even when class labels are not given as input, providing a more pronounced separation between classes. To result in sharper edges between facades and roof points, PointNet++ could be improved by adding feature extraction with cylindrical point neighborhoods in the first set abstraction layer.

ACKNOWLEDGEMENT

The authors would like to express their sincere thanks to BFS Swissphoto for providing the dataset of Leipzig at two different classification levels.

REFERENCES

Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., Çöltekin, A., 2015. Applications of 3D City Models: State of the Art Review. *ISPRS International Journal of Geo-Information*, 4(4), 2842-2889.

Boulch, A., Guerry, J., Le Saux, B., Audebert, N., 2018. SnapNet: 3D Point Cloud Semantic Labeling with 2D Deep Segmentation Networks. *Computers & Graphics*, 71, 189-198.

Boulch, A., 2020. ConvPoint: Continuous Convolutions for Point Cloud Processing. *Computers & Graphics*, 88, 24-34.

Klokov, R., Lempitsky, V., 2017. Escape From Cells: Deep Kd-Networks for the Recognition of 3D Point Cloud Models. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 863-872

Lei, X., Wang, H., Wang, C., Zhao, Z., Miao, J., Tian, P., 2020. ALS Point Cloud Classification by Integrating an Improved Fully Convolutional Network into Transfer Learning with Multi-Scale and Multi-View Deep Features. *Sensors*, 20(23), 6969.

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B., 2018. PointCNN: Convolution On χ-Transformed Points. *Advances in Neural Information Processing Systems*, 31.

Qi, C.R., Su, H., Mo, K., Guibas, L.J., 2017a. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 652-660.

Qi, C.R., Yi, L., Su., H., Guibas, L.J., 2017b. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *Advanced ion Neural Information Processing Systems*, 30.

Qi, C.R., Litany, O., He, K., Guibas, L.J., 2019. Deep Hough Voting for 3D Object Detection in Point Clouds. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 9277-9286.

Riegler, G., Ulusoy, A.O., Geiger, A., 2017. OctNet: Learning Deep 3D Representations at High Resolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 3577-3586*.

Schmohl, S., Sörgel, U., 2019. Submanifold Sparse Convolutional Networks for Semantic Segmentation of Large-Scale ALS Point Clouds. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, IV-2/W5, 77–84.

Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.-H., Kautz, J., 2018. SPLATNet: Sparse Lattice Networks for Point Cloud Processing. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2530-2539.

Thomas, H., Qi, C.R., Deschaud, J.-E., Marcotegui, B., Goulette, F., Guibas, L.J., 2019. KPConv: Flexible and Deformable Convolution for Point Clouds. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 6411-6420.

Varney, N., Asari, V.K., Graehling, Q., 2020. DALES: A Large-scale Aerial LiDAR Data Set for Semantic Segmentation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 717-726.

Weinmann, M., Jutzi, B., Hinz, S., Mallet, C., 2015. Semantic Point Cloud Interpretation Based on Optimal Neighborhoods, Relevant Features and Efficient Classifiers. *ISPRS Journal of Photogrammetry and Remote Sensing*, 105, 286–304.

Winiwarter, L., Mandlburger, G., Schmohl, S., Pfeifer, N., 2019. Classification of ALS Point Clouds Using End-to-End Deep Learning. *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 87, 75-90.

Wu, W., Qi, Z., Fuxin, L., 2019. PointConv: Deep Convolutional Networks on 3D Point Clouds. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yang, Z., Jiang, W., Xu, B., Zhu, Q., Jiang, S., Huang, W., 2017. A Convolutional Neural Network-Based 3D Semantic Labeling Method for ALS Point Clouds. *Remote Sensing*, 9(9), 936.

Zhang, W., Qi, J., Wang, P., Wang, H., Xie, D., Wang, X., Yan, G., 2016. An Easy-to-Use Airborne LiDAR Data Filtering Method Based on Cloth Simulation. *Remote Sensing*, 8(6), 501.

Zhao, R., Pang, M., Wang, J., 2018. Classifying Airborne LiDAR Point Clouds via Deep Features Learned by a Multi-Scale Convolutional Neural Network. *International Journal of Geographical Information Science*, 32(5), 960-979.

Zhou, Y., Tuzel, O., 2018. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4490-4499.