Post-print of: Kowalczuk Z., Szymański K.: Classification of objects in the LIDAR point clouds using Deep Neural Networks based on the PointNet model. IFAC-PapersOnLine. Vol. 52, iss. 8 (2019), pp. 416-421. DOI: 10.1016/j.ifacol.2019.08.099

Classification of objects in the LIDAR point clouds using Deep Neural Networks based on the PointNet model

Zdzisław Kowalczuk * **Karol Szymański** **

* *Department of Robotics and Decision Systems*
Faculty of Electronics, Telecommunications and Informatics
Gdansk University of Technology
Narutowicza 11/12, 80-233 Gdansk
(email: kova@pg.edu.pl).
** *Department of Robotics and Decision Systems*
Faculty of Electronics, Telecommunications and Informatics
Gdansk University of Technology
Narutowicza 11/12, 80-233 Gdansk
(email: karszyma@pg.edu.pl).

Abstract: This work attempts to meet the challenges associated with the classification of LIDAR point clouds by means of deep learning. In addition to achieving high accuracy, the designed system should allow the classification of point clouds covering an area of several dozen square kilometers within a reasonable time interval. Therefore, it must be characterized by fast processing and efficient use of memory. Thus, the most popular approaches to the point cloud classification using neural networks are discussed. At the same time, their shortcomings are indicated. A developed model based on the PointNet architecture is presented and the way of preparing data for classification is shown. The model is tested on a cloud coming from the 3D Semantic Labeling competition, achieving a good result, confirmed by the high quality of the system, i.e. a high rate of categorization of objects.

Keywords: aircraft operation, classification, neural networks

1. INTRODUCTION

One of the most popular tasks of unmanned aircraft is mapping and recognizing new areas, which should be scanned, e.g. by a flying vehicle using LIDAR. As a result of data processing an appropriate point cloud is created. Another task is the classification of the cloud points. Point cloud classification consists in assigning an object class to each point (which means connecting each point with a specific type of object). Of course, the number and names of classes may vary (this depends mainly on the number of different objects to be recognized). The most common classes are ground, low vegetation, medium vegetation, high vegetation and buildings, and unclassified facilities that are not qualified for the defined classes of objects.

To classify a point cloud that maps a large area (up to several hundred square kilometers), you must use the appropriate software. Such software is based on analytical algorithms that isolate each class separately. Unfortunately, the results are far from perfect (there are many classification errors). To achieve an accuracy exceeding 95%, the classification obtained should be corrected manually. Such a process is time-consuming and expensive. An opportunity to improve the classification accuracy and processing time is to use deep learning for this purpose.

In connection with the above, we present a model that can facilitate the classification of huge point clouds with high accuracy and in a short time using a deep neural network.

2. RELATED WORKS

The main difficulty in using neural networks for point cloud classification is the lack of regularity in the space distribution of cloud points. Popular convolutional networks operate on data whose subsequent samples are mutually shifted by a constant time step in the case of sound signals (Salamon and Bello (2017), Piczak (2015)) or a fixed amount of space in the case of digital images (Krizhevsky et al. (2012), He et al. (2016)). In the case of a point cloud, the distance between adjacent points is variable. On the other hand, the use of weight sharing is possible in text processing (Zhang et al. (2015), dos Santos and Gatti (2014)), because the order of words can always be indicated. However, this sort of ordering is generally unsuitable for point clouds. What's more, this issue is even more complex due to its large dimensionality and variable density in space. For these reasons, the cloud of points can be represented as an unsorted list of coordinates with additional features. That is why the vast majority of solutions that use neural networks to classify point clouds rely on converting a point cloud into a 2D image, projecting the cloud to a horizontal plane or 3D image.

In several works, it is proposed to classify clouds using a point-by-point approach, generating a 2D image for each point (Hu and Yuan (2016), Yang et al. (2017), Zhao et al. (2018a)). For each classified location, we get an image with pixels having certain features that can be associated with specific channels (of this image) assigned to each pixel (like the intensity of RGB colors). Each pixel maps a specific part of the analyzed area (therefore the features are determined on the basis of surrounding points mapping this area). Such features may be, for example, the average height above the ground, shape coefficients calculated on the basis of the eigenvalues of the coordinates of the points, the intensity of the return beam, the histogram of normal vector directions for each point, etc. However, this approach is not suitable for the classification of large areas, since the values of all functions must be calculated as many times as the pixels in the image. What's more, there are as many 2D images as there are points in the cloud. This makes it time to classify the points of the entire cloud unacceptably long, although the generated images can be classified directly by convolutional neural networks.

The considered problem of 3D clouds can be better solved using the 3D domain (instead of the above 2D mapping). So, let us extend the 2D image into 3D and introduce the idea of a voxel instead of a pixel. This approach is called voxelization (Jing Huang and Suya You (2016), Zeng Wang and Posner (2015), Engelcke et al. (2016), Zhou and Tuzel (2017), Song and Xiao (2014), Song and Xiao (2016), Li (2016)). As in the 2D image approach, a specific space is assigned to each mapping voxel. However, in contrast to the 2D method, quantization of space takes place only once during cloud analysis, which is a great advantage of this approach. The voxel can store information about space occupancy or other features based on the characteristics of points assigned to this voxel. After converting a point cloud into a 3D image, it can then be classified using a convolutional 3D network. However, this approach also has disadvantages. Namely, in this sort of aggregate description, many points are 'loaded' into one voxel (with a label), thus the exact location is lost (quantization introduces an error). What's more, when a LIDAR point cloud is classified, which contains high objects, such as cranes, buildings, high voltage lines, the vast majority of voxels remain empty. This makes this approach and representation very inefficient in storage (it requires memorizing a large number of unnecessary data).

Qi et al. (2016) proposed a solution for direct classification of point clouds using neural networks. The key idea of the PointNet model is to obtain information about the neighborhood of classified points. Instead of using multiple convolutional layers, maxpooling is applied to the features of the points in the cloud. This solution allows you to speed up the time-consuming process of processing raw data and reduce the size of stored data, while maintaining high classification accuracy. The first version of the model was adapted only to the classification of a cloud with a relatively small number of points (counted in thousands). Therefore, a few months later, in the next work (Qi et al. (2017)) the authors proposed a few additional solutions that adapted the PointNet model to objects in complex scenes. Among other things, there were proposed ways of extracting features in different scales of the neighborhood. However, the improved model, which has been tested only on small point clouds, is not adapted to the classification of point clouds covering tens of square kilometers.

The novelty of this work is the adaptation of the PointNet model for fast classification of large LIDAR cloud points.

3. THE PROPOSED METHOD

The following subsections show the stages of data processing in the proposed method. First, the algorithm for preparing data before entering them to the inputs of the neural network (NN model) is described. Next, the architecture of the neural network that classifies the analyzed points is discussed. Finally, the authors suggest a way to enlarge the training data set.

3.1 Pre-processing of data

Using professional LIDAR, point clouds are generated with a density of 10 to 120 points per square meter. To classify an area of a few square kilometers, sometimes you have to designate up to one billion labels. It is not possible to simultaneously classify all points of such a large point cloud. Therefore, the cloud must be divided into smaller parts, and the way this partition is executed is crucial to the performance of the system.

We suggest dividing the cloud into cuboids with the sides of the base with a length of r and the height to cover all points of the cloud. To assign each of the points to the appropriate cuboid, it is necessary to round the x and y coordinates. In addition to the point features that are directly accessible from the LIDAR, such as coordinates, intensity and the number of LIDAR returns, we introduce some additional features. Namely, for each point normal vectors are calculated, and the angle between the vector and the z axis is treated as another feature of the point. In addition, the x and y coordinates are normalized to the center of the cuboid in which they lie, and the z coordinate is normalized to the lowest point in the cuboid.

In total, 5 features are given to the input of the NN model, which together with the single-byte information about the point class takes only 21 bytes in the memory. Thanks to that, you can store data containing over 5 billion points in 100 GB memory. Such number of points can map the area from 40 to 500 square kilometers.

To capture the structure of the cloud at various scales, not only the features of the points inside the cuboid selected for classification are input to the neural network. To the analyzed area are also attached features of points coming from 8 nearest neighboring cuboids (second scale) or from 24 (weaker) neighboring cuboids (third scale). The three smallest processing scales are shown in figure 1. Classified cuboid is marked with navy crosses. Red lines mean a larger range of analysis (the second scale therefore includes 9 cuboids). And the largest area of analysis (third scale) includes 25 cuboids, which is marked in yellow.

To process each cuboid in the same way, it is convenient to have a fixed number of points (H) in them. If fewer points occur, existing points are simply copied to reach the fixed number H. However, if there are more than H

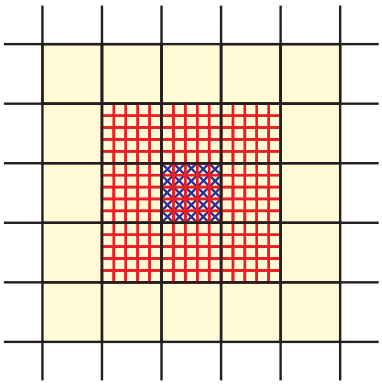

Fig. 1. Exemplary processing scales

points, then additional cuboids are created in the same location, so that each point is in at least one of them.

3.2 The DNN model

The PointNet model has been adapted to the classification of properly pre-processed LIDAR data. The block diagram of the whole Deep Neural Network (DNN) system is shown in figure 2, where sharp rectangles point to a data block, and rounded rectangles indicate data operations. The input data of the system is a cloud of raw points, which is a list of points with their initial geometrical and physical features. The output of the system is a matrix $H \times C$. The rows correspond to H classified points, and the positions in the individual columns indicate the probability with which each point belongs to each of the adopted C classes. At the beginning, the cloud of original (raw) points is pre-processed, as described in the previous section.

Pre-processed data is fed to the input of a deep NN (DNN) network based on the PointNet model. This network has 4 sets of inputs, which are initially processed independently (in parallel). In the first branch from the left we are looking for a different representation of the input geometrical and physical features. For this purpose, a four-fould multi-layer perceptron (MLP) is used. In figure 2, the number of neurons in subsequent layers is written in brackets, and each of the MLP layers uses a diode activation function (ReLU). So in this branch, each layer has the same number (64) of neurons. We assume that 4 layers are sufficient to determine some purified individual deep features (IDF) of each point (without any information about neighboring points). The input data of this branch is H points from the classified cuboid, which have (initially) F functions. The output of this branch is a matrix $H \times 64$, so for each of the H points of the analyzed cuboid there are 64 IDFs.

The second branch determines the extended IDFs in the sense of the order/complexity of the neural network layers. From this kind of 'noise' information represented by a large number of IDFs, we select only the most strong attributes (256) of all points (by maxpooling), called extended deep features (EDF), which are common to all cuboid points. In connection with the above, in order to restore all (H) cuboid points, we perform the multiplication phase, assigning the same EDFs to all the newly generated points.

The third branch performs the same operation, but instead of H points from the classified cuboid, $4 \cdot H$ points representing the second scale are created (processing 3x3 = 9 'boxes'). Similarly, $9 \cdot H$ points enter the fourth branch, taking into account the third scale (processing of 25 'boxes', the classified cuboid and 24 neighboring ones).

After concatenating all four data blocks, MLP processing is called again, allowing you to calculate the target C deep features (DF), where C is the number of desired target classes. By using softmax type activation in the last layer of MLP, the values of these features can be interpreted as the probability with which each point belongs to a particular class (one of C).

3.3 Data augmentation

It is worth to have as much training data as possible to train neural networks. A cloud of LIDAR measurement often contains many points and there is usually no problem finding important sections that should be classified. It is much more difficult to find data already well classified (with high accuracy), which can be used as training data for the model being taught. This problem can be solved, for example, by generating new data based on existing data.

In the case of the discussed method, two simple transformations can be performed for this purpose. The first of these is the rotation of the entire cloud. In this case, the created cuboids are differently oriented in the analyzed cloud. The second method is to move each point in the cloud by a certain vector. Then the coordinates of the cuboid vertices are different. Both transformations lead to new sets of points, which are in each cuboid. As a result, training data can be broadly expanded.

4. EXPERIMENTS

The applied data processing algorithm has been implemented in Python. The PointNet model used was programmed using the Keras library with the TensorFlow backend. All experiments were carried out using the Nvidia GTX 1080 Ti graphics card.

Model tests were performed on data from the ISPRS 3D Semantic Labeling Contest. The goal of the competition is to classify the LIDAR point cloud collected from the city of Vaihingen on the basis of an already classified cloud from the same town. As part of the competition, each item should be classified as one of 9 categories (in color):

(1) power line (blue)
(2) pow vegetation (lime)
(3) impervious surfaces (aegean)
(4) car (canary)
(5) fence/Hedge (orange)
(6) roof (red)
(7) facade (yellow)
(8) shrub (pear)
(9) tree (green).

Figure 3 shows the training cloud containing 412 thousand points in which different detected classes are marked with different colors. The reference labels have been taken over from the authors Niemeyer et al. (2014). The evaluation of the results is based on the average value of the F1 index (Nancy Chinchor (1992)) for each class.

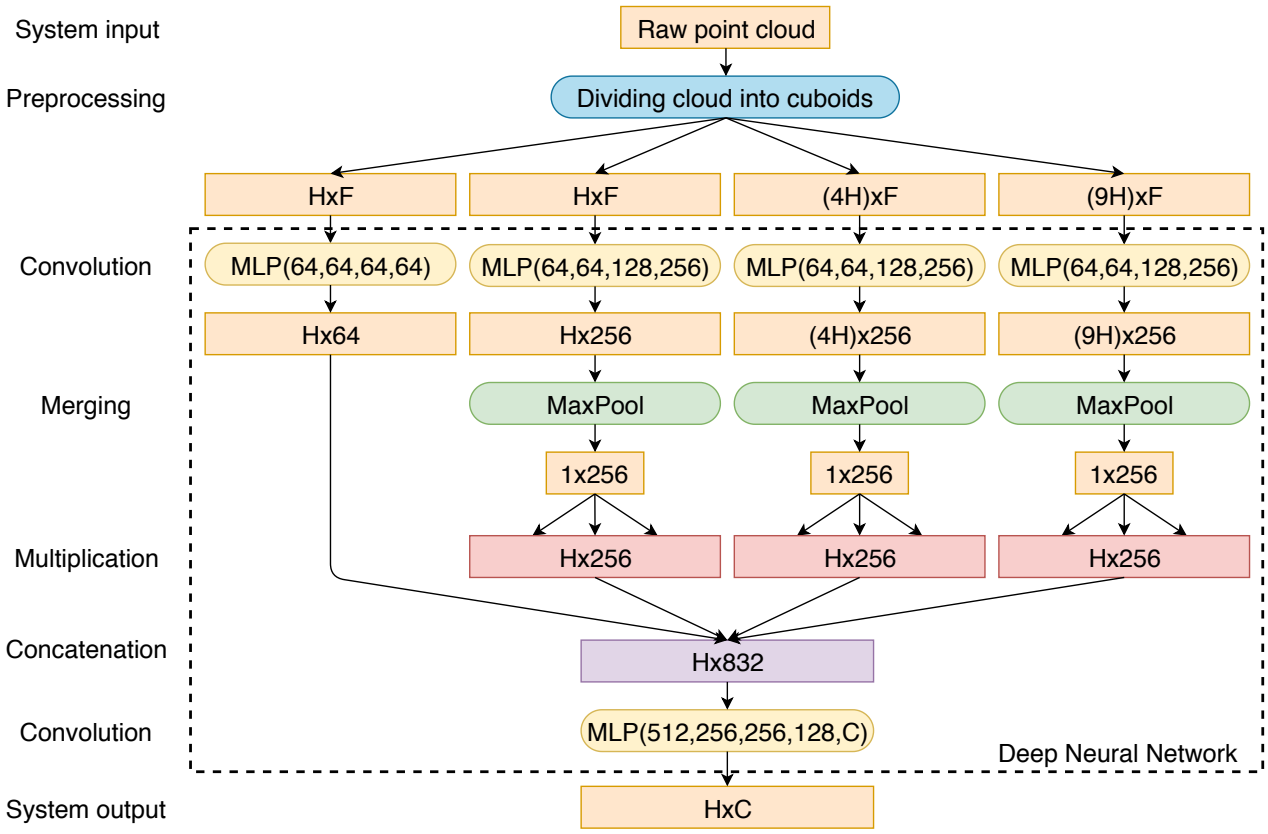

Fig. 2. Block diagram of the whole DNN system

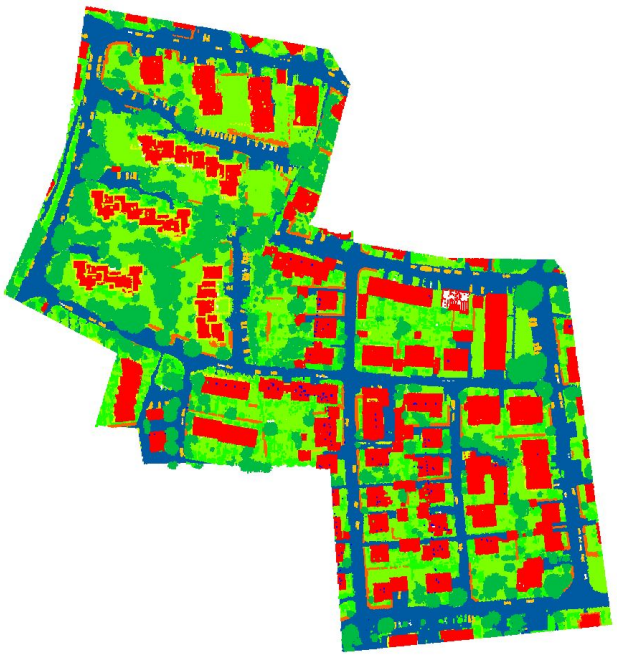

Fig. 3. A view of a cloud of training points indicating the reference class for each point

4.1 Tuning of hyperparameters

To ensure that the training of the model will bring the expected results, it is necessary to adjust the model's hyperparameters. The first parameter is the learning speed, which determines how dynamically the weighing of the model changes during training. To strengthen system generalization, we use the L2 regularization Schmidhuber (2014). The scaling factor with which the loss L2 is controlled in the applied loss function is the second hyper parameter to be set. The last two hyperparameters are the length of the side of the cuboid base (r) and the number of points in each of them (H).

To properly select the hyperparameters of the algorithm, we performed 100 model training cycles, each of which lasted 5 epochs. In each training cycle, the training data set consisted of 90% cuboids generated from training data without any data enlargement, and the validation set contained the remaining 10%. For each training cycle, hyperparameters were randomly selected from the following ranges:

- learning rate: $(1 \cdot 10^{-1}; 1 \cdot 10^{-6})$
- L2 regularization factor: $(1 \cdot 10^{-1}; 1 \cdot 10^{-6})$
- r: $(1m; 6m)$
- H number: $(50; 150)$.

In the experiments conducted, the highest accuracy of classification (84.8%) for validation data was obtained with the following values of the hyperparameters:

- learning rate: $1.49 \cdot 10^{-3}$
- L2 regularization factor: $7.94 \cdot 10^{-4}$
- r: $2.55m$
- H: 116

and these values were chosen for the final training of the considered NN model.

4.2 Model training

The final training data set contained 90 extended and original training data sets. Sets created from cloud rotation by one of ten angular values around the z axis and from a shift by one of the three selected values along the x or y axis. In total, over 900,000 cuboids containing different sets of points were used in the final training. 5% cuboids served as validation data. Final training lasted 50 epochs. The model achieved 94.86% accuracy of classification on training data and 93.25% on validated data.

4.3 Evaluation of the model on test data

Using the trained model, a test point cloud was classified. Figure 4 shows a sample view obtained for the test cloud, after being classified by the trained model. The same colors are used to mark individual classes, as in the case of the training cloud. As can be seen in the figure, the proportions of points belonging to each class are well approximated. Detailed results of this classification are presented in Table 1. It can be noticed there that underrepresented classes are much worse identified (clasified) than those with more representatives. One reason may also be that points belonging to an underrepresented class are much more difficult to classify due to the small size of the objects to which they belong.

Table 2 presents a comparison of the results obtained on the test data with the results achieved by means of three other methods that took part in the ISPRS Semantic Labeling Contest (3D). NANJ2 (Zhao et al. (2018b)) and WhuY3 (Yang et al. (2017)) are methods in which 2D feature images are created for each point. In addition, NANJ2 generates images at various scales. Next, the images are classified using convolutional neural networks. The IIS7 method (Ramiya et al. (2016)) uses geometric and spectral features in conjunction with segmentation based on voxels and and a growing color region. Even the best NANJ2 method has achieved only an 8.7% bigger F1 score in the classification of test data than our method, which means that our method has made 60% more mistakes on test data than the winner of ISPRS Semantic Labeling Contest (3D). Our method is not only better than the IIS7 method, but it is also better than the results of half of all participants. This means that the proposed approach is quite accurate at a relatively high speed of operation (classification).

Data preparation and the test cloud classification, which maps the area of approximately 6 hectares and has 412 thousand points, lasted 22 seconds. You can therefore calculate that the system can process over 10 square kilometers per hour and over one million points per minute. Such data processing speed is sufficient to classify LIDAR point clouds mapping huge areas.

5. CONCLUSION

The proposed NN model met the expected requirements: a good result was achieved with the three-dimensional semantic marking; the process of data preparation and classification has been accelerated; and at the same time generated training data take up not much space.

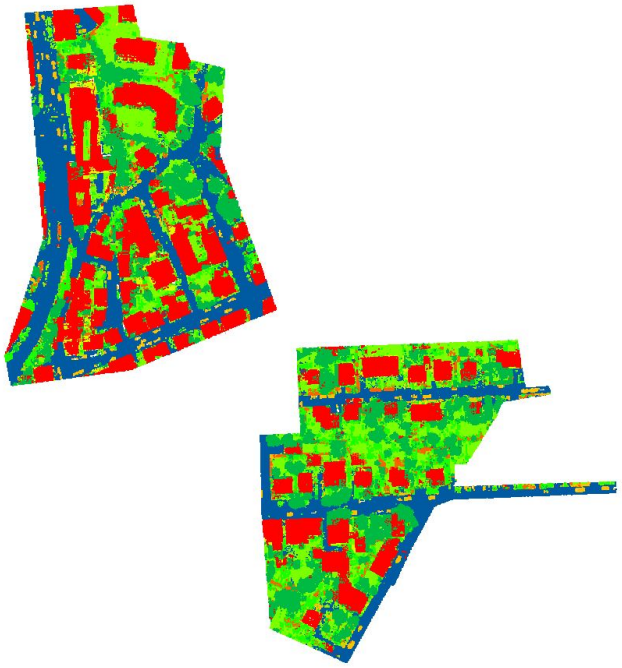

Fig. 4. A sample view of the test point cloud classified by the trained DNN model

Experience shows that the assessment of height above the ground has a big impact on the accuracy of the classification. Therefore, in the future, we would like to divide the classification process into two stages: basic and other. Then, after proper classification of the ground, precise determination of the height of objects above the ground should be easier.

ACKNOWLEDGEMENTS

This research was partially implemented as part of the project "Development of a self-learning IT analytical platform supporting the management of forests based on image data from photogrammetric cameras and point clouds of a multi-spectral laser scanner", Contract No. UDA-RPWM.01.02.01-28-0012/16-00 of on 07/11/2017 Project co-financed by the European Union from the European Regional Development Fund under the Regional Operational Program of the Warmińsko-Mazurskie Voivodeship for 2014-2020, Priority axis 1 Intelligent economy of Warmia and Mazury, Measure 1.2 Innovative companies, Sub-measure 1.2.1 R&D activities of enterprises.

REFERENCES

dos Santos, C. and Gatti, M. (2014). Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, 69–78.

Engelcke, M., Rao, D., Wang, D.Z., Tong, C.H., and Posner, I. (2016). Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. *CoRR*, abs/1609.06666. URL `http://arxiv.org/abs/1609.06666`.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings*

class names	precision	recall	f1-score	support
Powerline	0.0%	0.0%	0.0%	600
Impervious surfaces	87.8%	90.3%	89.1%	101986
Tree	62.7%	85.7%	72.4%	54226
Shrub	32.0%	35.5%	33.7%	24818
Low vegetation	81.2%	75.0%	78.0%	98690
Facade	41.6%	31.1%	35.6%	11224
Car	51.4%	64.8%	57.3%	3708
Fence/Hedge	24.0%	26.6%	25.2%	7422
Roof	92.6%	78.6%	85.0%	109048
average / total	77.9%	76.5%	76.8%	411722

Table 1. Precision, recall and F1 score for every class for the test dataset

method	precision	recall	f1-score
Ours	77.9%	76.5%	76.8%
NANJ2	86.4%	85.2%	85.5%
WhuY3	81.6%	82.3%	81.6%
IIS7	78.4%	74.7%	74.2%

Table 2. Comparison of the results obtained by the proposed DNN method with the results achieved by three other methods participating in the Semantic Labeling Contest

of the IEEE conference on computer vision and pattern recognition, 770–778.

Hu, X. and Yuan, Y. (2016). Deep-learning-based classification for DTM extraction from ALS point cloud. *Remote Sensing*. doi:10.3390/rs8090730.

Jing Huang and Suya You (2016). Point cloud labeling using 3D Convolutional Neural Network. In *2016 23rd International Conference on Pattern Recognition (ICPR)*. doi:10.1109/ICPR.2016.7900038.

Krizhevsky, A., Sutskever, I., and Hinton, G.E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Li, B. (2016). 3d fully convolutional network for vehicle detection in point cloud. *CoRR*, abs/1611.08069. URL `http://arxiv.org/abs/1611.08069`.

Nancy Chinchor, P. (1992). Muc-4 evaluation metrics. In *FOURTH MESSAGE UNDERSTANDING CONFERENCE (MUC-4), Proceedings of a Conference Held in McLean, Virginia, June 16-18, 1992*. URL `http://aclweb.org/anthology/M92-1002`.

Niemeyer, J., Rottensteiner, F., and Soergel, U. (2014). Contextual classification of lidar data and building object detection in urban areas. *ISPRS journal of photogrammetry and remote sensing*, 87, 152–165.

Piczak, K.J. (2015). Environmental sound classification with convolutional neural networks. In *Machine Learning for Signal Processing (MLSP), 2015 IEEE 25th International Workshop on*, 1–6. IEEE.

Qi, C.R., Su, H., Mo, K., and Guibas, L.J. (2016). Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593. URL `http://arxiv.org/abs/1612.00593`.

Qi, C.R., Yi, L., Su, H., and Guibas, L.J. (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *CoRR*, abs/1706.02413. URL `http://arxiv.org/abs/1706.02413`.

Ramiya, A.M., Nidamanuri, R.R., and Krishnan, R. (2016). Object-oriented semantic labelling of spectral–spatial lidar point cloud for urban land cover classification and buildings detection. *Geocarto International*, 31(2), 121–139. doi:10.1080/10106049.2015.1034195. URL `https://doi.org/10.1080/10106049.2015.1034195`.

Salamon, J. and Bello, J.P. (2017). Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3), 279–283.

Schmidhuber, J. (2014). Deep learning in neural networks: An overview. *CoRR*, abs/1404.7828. URL `http://arxiv.org/abs/1404.7828`.

Song, S. and Xiao, J. (2014). Sliding shapes for 3D object detection in depth images. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. doi:10.1007/978-3-319-10599-4-41.

Song, S. and Xiao, J. (2016). Deep sliding shapes for amodal 3d object detection in rgb-d images. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 808–816.

Yang, Z., Jiang, W., Xu, B., Zhu, Q., Jiang, S., and Huang, W. (2017). A convolutional neural network-based 3D semantic labeling method for ALS point clouds. *Remote Sensing*. doi:10.3390/rs9090936.

Zeng Wang, D. and Posner, I. (2015). Voting for Voting in Online Point Cloud Object Detection. In *Robotics: Science and Systems XI*. doi:10.15607/RSS.2015.XI.035.

Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, 649–657.

Zhao, R., Pang, M., and Wang, J. (2018a). Classifying airborne LiDAR point clouds via deep features learned by a multi-scale convolutional neural network. *International Journal of Geographical Information Science*. doi:10.1080/13658816.2018.1431840.

Zhao, R., Pang, M., and Wang, J. (2018b). 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. doi:10.1080/13658816.2018.1431840. URL `https://doi.org/10.1080/13658816.2018.1431840`.

Zhou, Y. and Tuzel, O. (2017). Voxelnet: End-to-end learning for point cloud based 3d object detection. *CoRR*, abs/1711.06396. URL `http://arxiv.org/abs/1711.06396`.