Exploring Explainability and Robustness of Point Cloud Segmentation Deep Learning Model by Visualization

Floris Verburg
University of Twente
PO Box 217, 7500 AE Enschede
the Netherlands
f.m.verburg@student.utwente.nl

ABSTRACT

Modern deep learning techniques are very suitable for point cloud segmentation of catenary arches of railway systems. The downside of deep learning models is that they have a low explainability. In this paper we explore the explainability of a PointNet++ model that is used for segmenting point clouds of catenary arches. The exploring of explainability is done by creating a pipeline for adapting, segmenting and visualizing the point clouds. By adapting the point clouds the robustness of the model is also tested. From this research it follows that the PointNet++ model mainly relies on the location of the objects in order to segment them. Changing the shape of an object does not have a significant impact on the performance of the model.

Keywords

Point Cloud Segmentation, Deep Learning, Visualization, Open3d, Explainability, Robustness

1. INTRODUCTION

About 11% of the people in the Netherlands rely on the efficient railway infrastructure the country has to offer [1]. To keep the trains arriving on time, renovation, construction and repairs of this railway infrastructure must happen regularly. The inspection of the catenary arches is part of the maintenance and is very time consuming. Railway infrastructure company Strukton wants to automize the detection of the different objects of these catenary arches to efficiently inventorize which catenary arch components are used where. To find out if this automatization is possible Strukton employed a research group to see if they can create a 3D CAD model of the railway environment using point cloud segmentation and deep learning.

The railway environment is scanned and stored as a point cloud using LiDaR. Deep learning models are used for semantic segmentation of the catenary arches in this point cloud. A modified PointNet++ model has shown good results. However, as with every deep learning model, it is unknown how the model precisely does the segmentation. In this research the so called 'black box' of the deep learning model will be opened to explain how the model works and to see how robust the model is. We also provide architecture and implementation of a pipeline for creating such visualizations.

36th Twente Student Conference on IT, Febr. 4nd, 2022, Enschede, The Netherlands.

In section 2 the relevant background of this research is covered. Followed by the related work in section 3. In section 4 the problem statement is defined, containing the research questions that will be answered in this paper. The methodology used to answer these questions is covered in section 5. The results can be found in section 6.

2. BACKGROUND

To better understand the research, definitions for the used terms and tools are given. First the definition and application fields of point clouds is given. Then semantic segmentation is explained, followed by the open source software Open3D-ML. Lastly, explainability of deep learning is elaborated on.

2.1 Point Clouds & LiDaR

The railway environment has been captured in a 3D representation using 3D LiDaR (Laser Imaging Detection and Ranging). LiDaR uses a fast-rotating laser to determine the range of objects 360° around the laser. It measures the time of flight for the light of the laser to reflect on an object and calculates the distance to the object using the measured time. The objects are represented in large amount of points, each having an X, Y and Z coordinate. The collection of these points is called a Point Cloud. The use of LiDaR to create Point Clouds for semantic segmentation has been proven successful in various fields such as the segmentation of tree crowns [2] and vegetable crops [7]. Furthermore, LiDaR is widely used in the automotive industry for autonomous driving [8].

2.2 Semantic Segmentation

The goal of semantic segmentation in a 2D environment is to classify each pixel belonging to a particular label. Image segmentation is applied in various fields, ranging from medical [13] to military [18].

Semantic segmentation of point clouds is very similar to semantic segmentation of images. The only difference is that it is in 3D instead of 2D. Instead of each pixel getting classified, each point is classified. Semantic segmentation of point clouds is widely used in automotive to classify objects [4].

Several techniques exist to perform the semantic segmentation [3]. One of the first techniques used for image segmentation was split-and-merge. This technique mainly relies on hardcodes rules and is very outdated for this application (for details see [6]). In recent years the focus has shifted towards deep learning. One of the deep learning methods used for semantic segmentation on point sets is PointNet [11]. An example of Point Cloud Segmentation of a catenary arch using PointNet can be found in Figure 1, this picture is taken from [9]. This deep learning model was later further developed into PointNet++, which uses a hierarchical neural network that applies PointNet recursively [12].

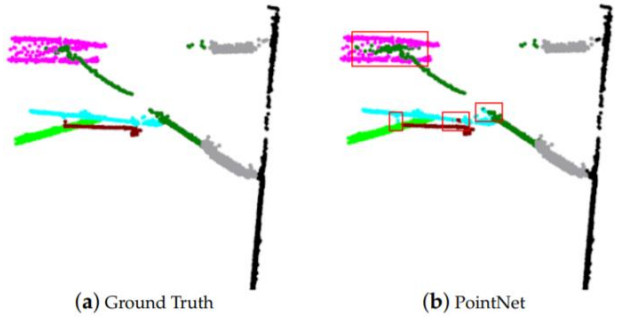

Figure 1 Example of Point Cloud Segmentation by using PointNet

One of the limits of these deep learning models is their low explainability. The model receives an input set, it does a lot of calculations in the so called "black box", and then returns the output. Increasing the explainability will make this black box more transparent [5].

2.3 Open3D-ML

Open3D-ML is a modern library that can be used for 3D machine learning tasks. It can be used to visualize labeled point clouds and it is built on the Open3D core library [10]. An example of a 3D representation of a crossroad can be found in Figure 2. In this figure the road is labeled in purple, trees and other vegetation in green and buildings in blue.

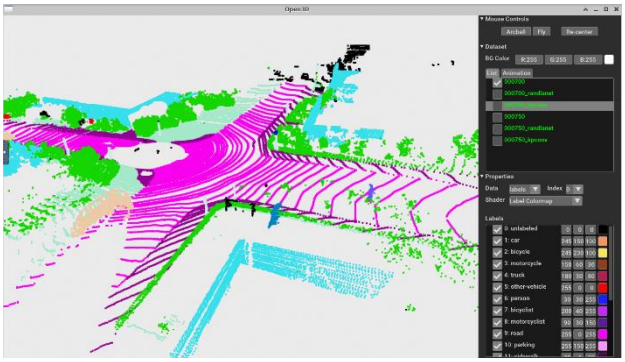

Figure 2 Example Open3D-ML [1]

2.4 Explainability

In this section we will look at the definition of explainability, its importance and available techniques to explore explainability.

2.4.1 Definition and Importance

To be able to make a deep learning model more explainable, we first need to define what explainability means. More specifically, what is the difference between explainability and interpretability? Interpretability is the degree to which a human can understand the cause of a decision [15]. For a model with a high interpretability it is easier for humans to understand the decisions than for a model with low interpretability. Explainability is the extent to which the internal mechanics of a model can be explained in human terms.

When a model has a high explainability, the trust in this model will increase [14]. Furthermore, when a model has a high explainability, its performance can increase. If you know why decisions were made, you can tweak the model to increase its performance.

2.4.2 Available techniques

Before listing the available techniques to improve model explainability, we first explain the types of techniques available.

First of all, the technique can be global or local. Global techniques look at the whole model and local techniques study a small portion of the network [15]. The second distinction that can be made between the techniques is model-specific versus model-agnostic. A model specific technique is specifically made for one model and cannot be used on other models. A model-agnostic technique can be used for multiple models.

In this research the focus will be on a model-specific and local technique.

3. RELATED WORK

As mentioned before, the railway environment has been captured using LiDaR and stored in a point cloud. A PointNet++ model was trained to segment and label the point cloud. One of these labeled arches visualized using the Open3D-ML library can be seen in Figure 3. This work and the results are all covered in [16].

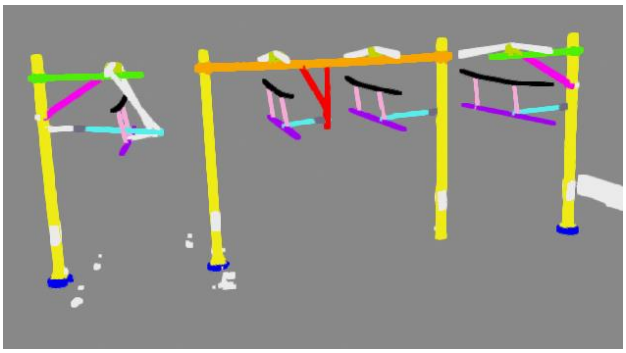

Figure 3 Labeled Arch Open3D-ML

The accuracy of the model is quite high, as it has a mIoU (mean Intersection over Union) of 0.86. However, the explainability of the model is not discussed. Furthermore, the robustness of the model is not tested. In this paper we analyze the effect of a permutation to the shape and location of an object in the arch to the performance of the model.

4. PROBLEM STATEMENT

Since the model has a low explainability, the trust in the model might not be that high. Exploring the explainability of the model will help us understand how the model works. To explore the explainability some research questions were formulated.

4.1 Research Questions

This paper aims to answer the following research question:

RQ: How to visualize point clouds to leverage them for model explainability and model robustness?

The following sub-questions are used to answer the main research question:

SRQ1: How to create an end-to-end pipeline to adapt, segment and visualize point clouds?

SRQ2: Does de model rely on the (relative) location of objects for the semantic segmentation?

SRQ3: Does the model rely on the shape of the objects for the semantic segmentation?

[1] source: http://www.open3d.org/docs/release/open3d_ml.html

5. METHODOLOGY

This section describes the steps that have been taken to answer the research questions.

5.1 Flowchart of Pipeline

This section explains the pipeline that has been created to adapt, segment and visualize the point clouds. For this research only the point cloud of one catenary arch is used. This is used as the input of the pipeline. A flowchart of the pipeline can be found in Figure 4. The details of each step are discussed later in sections 5.2, 5.3, 5.4 and 6.

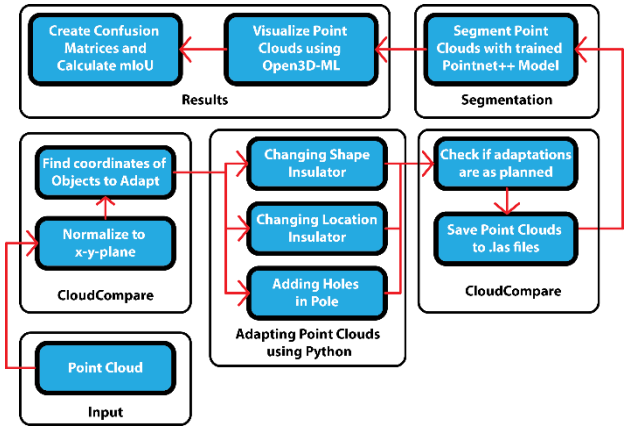

Figure 4 Flowchart of the Pipeline

5.2 Visualization using Open3D-ML

To visualize the segmented point clouds the open source library Open3D-ML is used. Since the visualization of multiple point clouds at the same time is quite resource intensive, the Open3D-ML library was installed on the Jupyter Notebook environment of the University of Twente, which has more than enough computing power for this task.

5.3 Adapting Point Clouds

A point cloud of a catenary arch that has been used for training and testing the deep learning model was adapted in several ways to find out if the performance of the model decreased by doing so. The arch used is displayed in Figure 5. By altering the arch and then letting the trained deep learning model segment the arch, the robustness of the model can be tested. Furthermore, if the performance decreases after modifying aspects of the arch it can be assumed that the deep learning model relies on those aspects during the segmentation. This will help us understand how the model works, thus increasing its explainability.

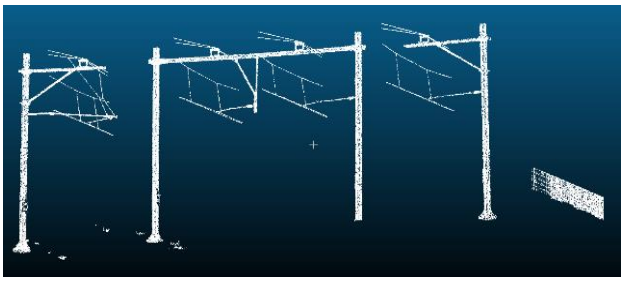

Figure 5 Original Point Cloud in CloudCompare

The two objects that have been adapted to test the robustness of the model are the second pole from the left and the third insulator from the left. To be able to add the adjustments in the point cloud, it first had to be normalized such that the front view would be seeing the x-y-plane. This was done in the point cloud editing software called CloudCompare[2]. The result of this transformation can be seen in Figure 6.

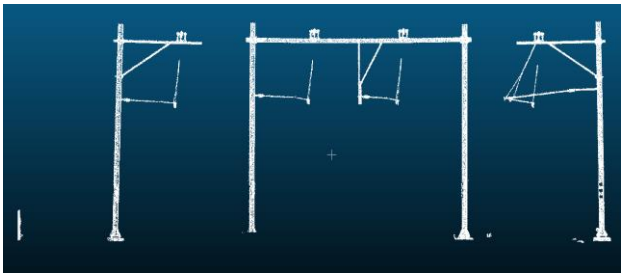

Figure 6 x-y-view of Normalized Point Cloud in CloudCompare

The normalized point cloud was saved to a .csv file and is now easy to process using Python scripts. To find out what aspects of the point cloud have influence on the performance of the model several different kinds of transformations were done. The location of the insulator was changed, the shape of the insulator was changed and different amounts of holes were added in the pole.

5.3.1 Changing Insulator Shape

To find out if the model relies on the shape of the different objects in the point cloud in order to segment them, one of the insulators was replaced by a cylinder.

In order to achieve this a Python script was written. In this Python script, the points forming the original insulator were removed. To find out which points formed the original insulator CloudCompare was used. This software has an option that allows you to select points in the point cloud and save the coordinates to a .csv file, as shown in Figure 7. By selecting six points, each with either the minimum or maximum value of the x-, y- or z-coordinate, the boundaries of the insulator were found.

A for loop was used to remove the points with coordinates between these minimum and maximum x-, y-, and z-coordinates.

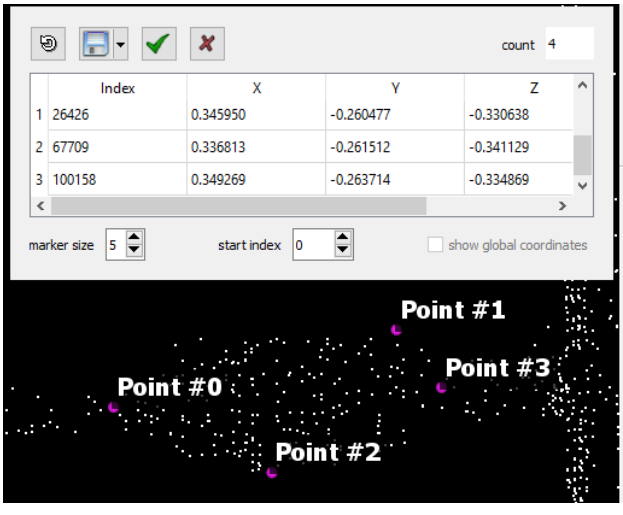

Figure 7 CloudCompare Point Selection

Using the same option in CloudCompare two points were chosen in between which the cylinder should be placed. First of all the formula for the coordinates for the line between these points was created. The equation used to find the x-, y- and z-coordinates of the points that form the line between point 1 $[x_1, y_1, z_1]$ and point

[2] https://www.danielgm.net/cc/

2 $[x_2, y_2, z_2]$ can be found in Equation 1. In this equation d gives the distance between point 1 and the new point.

Equation 1

$$x, y, z \rightarrow (x_2 - x_1) \cdot d + x_1, (y_2 - y_1) \cdot d + y_1, (z_2 - z_1) \cdot d + z_1$$

This equation was used to create 100 equally spaced points between point 1 and point 2. Around each of these points a circle in the y-z-plane was made using the formula in Equation 2. This equation was used to transform the cartesian coordinates to cylindrical coordinates. In this equation the letter n determines the amount of points for each circle. The value of n was 50. The letter i increments from 0 to n. The letter r determines the radius of the circle. Multiple radiuses were used to see if the size of the cylinder had impact on the performance of the model.

Equation 2

$$[x_c, y_c, z_c] \rightarrow [x, y + \cos\left(\frac{360}{n} \cdot i\right) \cdot r, z + \sin\left(\frac{360}{n} \cdot i\right) \cdot r$$

The points that form the circles were added to the point cloud and the points that form the centerline were not added. The newly added cylinder consisted of 5000 points, which is much more than the original insulator consisted of, namely 217 points. Therefor points of the new cylinder were randomly removed until it had the same amount of points as the original insulator.

The label belonging to the insulator was added to the cylinder to be able to compare the true label with the predicted label later.

The original and replaced insulator can be seen in Figure 8 and Figure 9.

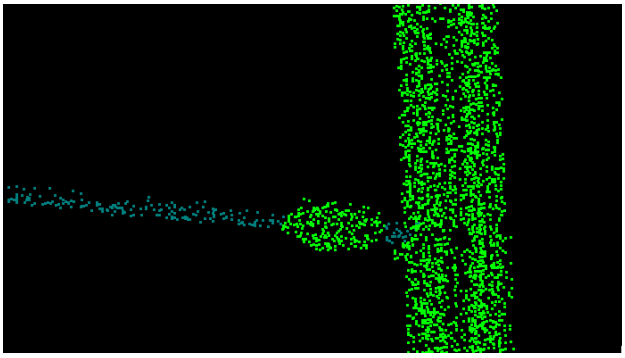

Figure 8 Original Insulator

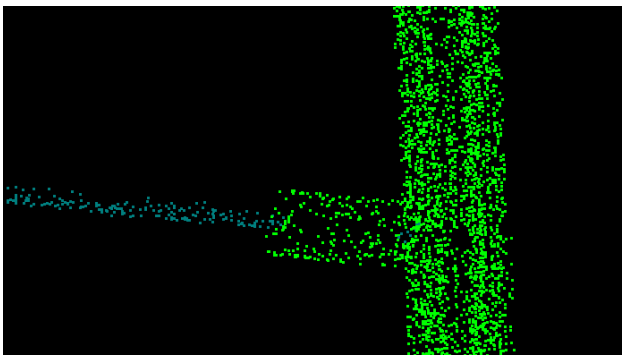

Figure 9 Replaced Insulator

5.3.2 Changing Insulator Location

To find out if the model relies on the location of the objects in the point cloud in order to segment them, one of the insulators was moved in various directions with various distances.

To move an insulator another Python script was written. Two points from the point cloud were selected using CloudCompare, one point in the middle of the insulator to be moved $[x_1, y_1, z_1]$, and one randomly in the point cloud $[x_2, y_2, z_2]$. These two points form a line on which the insulator is moved.

Changing the location of the insulator was done as follows: A for loop loops through all the points of the normalized point cloud and checks for each point if their coordinates are in the boundaries that were defined earlier. If this is the case it means that the point is part of the insulator that has to be moved. The points forming the insulator get a linear transformation with the formula from Equation 1. In this equation the letter d determines the distance that the insulator is moved. Several point clouds were created, each with another distance.

Besides several distances, multiple directions were also used to move the insulator. This was done by selecting a new point 2 $[x_2, y_2, z_2]$. The moved insulator kept the original label to later compare it to the predicted label. A picture showing the moved insulator can be found in Figure 10.

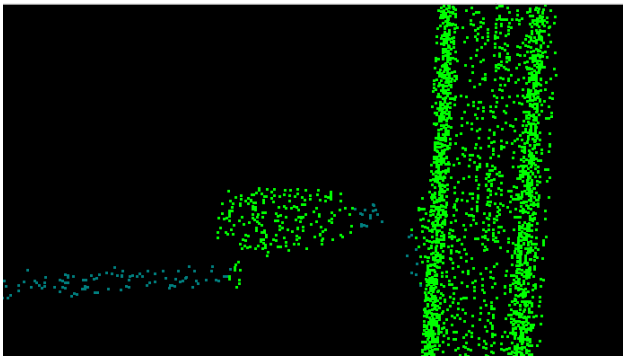

Figure 10 Moved Insulator

5.3.3 Changing Pole Shape

To find out if the model relies on the shape of the pole to classify it, different amounts of holes were added in one of the poles. Yet another Python script was created to achieve this. Again two points were selected using CloudCompare, one at the bottom of the pole (point 1 $[x_1, y_1, z_1]$) and one at the top of the pole (point 2 $[x_2, y_2, z_2]$).

A list containing the coordinates that define n squares on the line from point 1 to point 2 was made. For each square the following values were stored:

$$x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max}$$

By looping through all the points from the normalized point cloud and checking if their coordinates are between the extremes of any of the squares a new point cloud was created. Several point clouds with holes in one of the poles were made, ranging from 5 holes up to 80 holes. In the pole with 80 holes the holes are connected to each other resulting in a gap all along the pole.

Both a normal pole and a pole containing holes is shown in Figure 11.

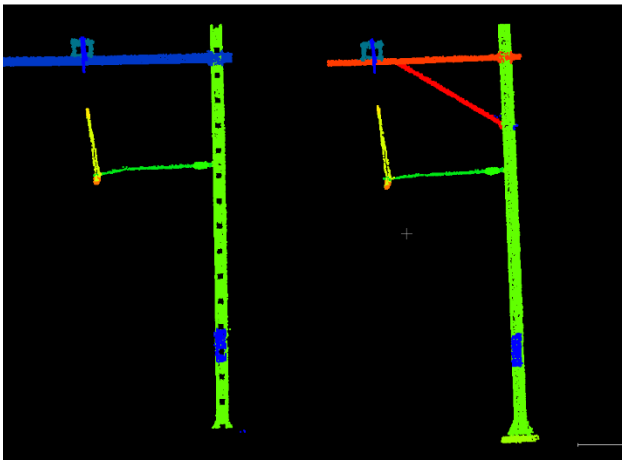

Figure 11 Left: Pole with 18 holes. Right: Normal Pole

5.4 Semantic Segmentation of new Point Clouds

Now that the new point clouds with alternations have been created, they can be segmented using the trained deep learning model (PointNet++) that was also used on the original point clouds [16].

The segmentation was done by using the UT-JupyterLab environment which has enough computing power to perform the segmenting in a short amount of time [17].

By visualizing the segmented point clouds with Open3D-ML and comparing them with the original point cloud we can see if the model performs equally well or not. The adapted objects in the new point clouds should have gotten the same label as in the original point cloud. If this is not the case, we can conclude that the model relies on the adapted parameter to do the segmentation. For example, if the point cloud where the insulator was moved does not receive the correct labels, we can conclude that the PointNet++ model relies on the location of the insulator with respect to the other components in order to segment it.

To get a measurable performance of the model, a confusion matrix was created for each adapted point cloud. The confusion matrix clearly shows for each object which label it should have gotten and which label it has received. The confusion matrix of the original point cloud can be found in Figure 12. As you can see in the matrix, 90% of the insulators were predicted as insulators. The mIoU corresponding to this matrix is 0.86. The mIoU is also calculated for all the matrices of the adapted point clouds to compare the performance with the original point cloud.

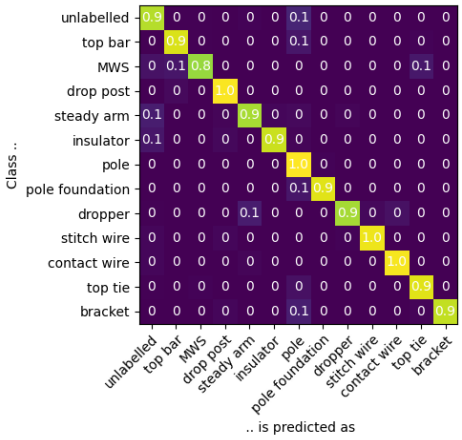

Figure 12 Confusion Matrix Original Point Cloud

6. RESULTS

The results are divided into the three adaptations that have been made. In section 6.1 we discuss the results of the point clouds in which the insulator was replaced by a cylinder. In section 6.2 we discuss the results of the point clouds in which the insulator was moved and in section 6.3 we discuss the results of the point clouds in which holes were added in one of the poles.

6.1 Changed shape of insulator

We will first look at the results of segmented the point cloud in which an insulator was replaced with a cylinder with a small radius. The point cloud with the true labels can be seen in Figure 13 and the point cloud with the predicted labels can be seen in Figure 14.

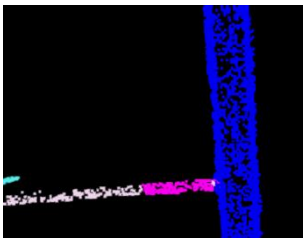

Figure 13 Small Cylinder True

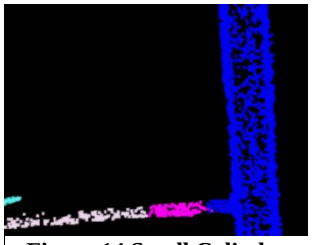

Figure 14 Small Cylinder Predicted

The model labeled most of the cylinder as an insulator. The part that is connected to the pole has been labeled as a pole, this probably happened because the normal insulator is not directly connected to the pole so the cylinder was made a bit too long.

In Figure 15 and Figure 16 the results with two slightly larger cylinders are shown. The same behavior as with the small cylinders is observed.

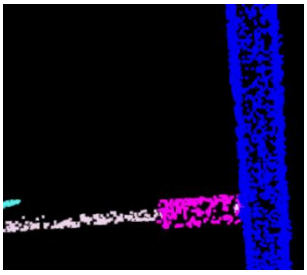

Figure 15 Medium Cylinder True

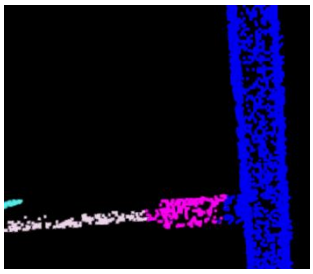

Figure 16 Medium Cylinder Predicted

Lastly, we take a look at the cylinder with a large diameter. The results can be seen in Figure 17 and Figure 18. Once again the model performed the same as before.

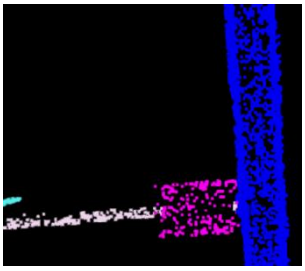

Figure 17 Large Cylinder True

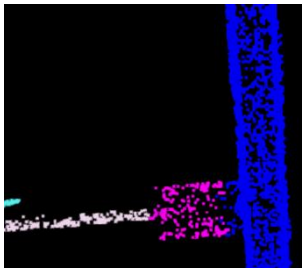

Figure 18 Large Cylinder Pred

Since the model segmented the point cloud just as good as it did with the original point cloud there is no difference in the confusion matrix with respect to the original confusion matrix in Figure 12. The mIoU of each of the adapted point clouds varied

between 0.85 and 0.86, which is very similar to the mIoU of the original point cloud (0.86).

6.2 Moved location of insulator

6.2.1 Visualization

Let us first look at the prediction versus the true labels of a point cloud where the insulator was only moved a small amount. The cloud with the true labels can be found in Figure 19 and the cloud with the predicted labels can be found in Figure 20.

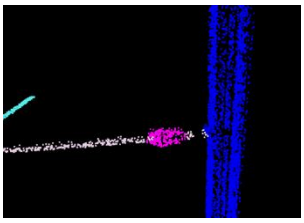

Figure 19 Small Movement True Labels

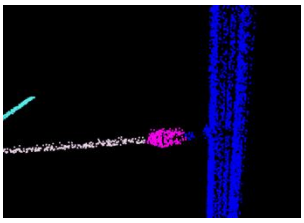

Figure 20 Small Movement Predicted Labels

As you can see the model labeled the insulator correctly. This is to be expected since the insulator was only moved a small amount.

In Figure 21 and Figure 22 we can see how the model segmented the point cloud when the insulator was moved a little bit further.

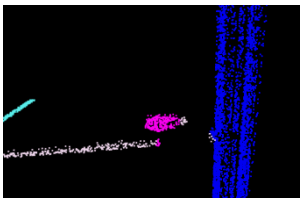

Figure 21 Medium Movement True

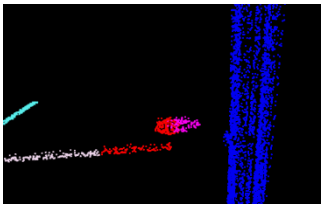

Figure 22 Medium Movement Predicted

The model struggled a lot more when segmenting this insulator. The red points are points that are unlabeled. It still managed to label some of the insulator correctly, but it failed in labeling the entire connecting rod between the insulator and the contact wire.

Now we will have a look how the model performs when the insulator is moved even further. The point cloud with the true labels can be found in Figure 23 and the point cloud with the predicted labels can be found in Figure 24.

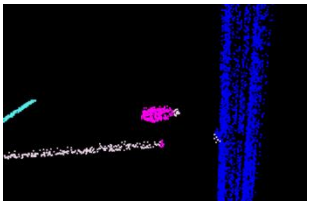

Figure 23 Large Movement True

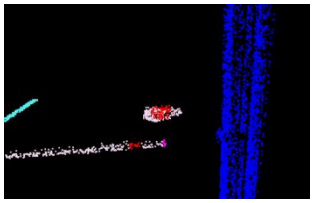

Figure 24 Large Movement Predicted

As you can see, the model gave barely any of the points a correct label. The red points in Figure 24 were unlabeled, meaning that the model was unable to label them. The white points were labeled as the connection rod that connects the insulator with the contact wire.

The model performed worse as the insulator was moved a larger distance from its original place. When the insulator was only move slightly it still performed well. The direction in which the insulator was moved did not matter, as you can see in Figure 25, where two point clouds with the predicted labels are displayed as one.

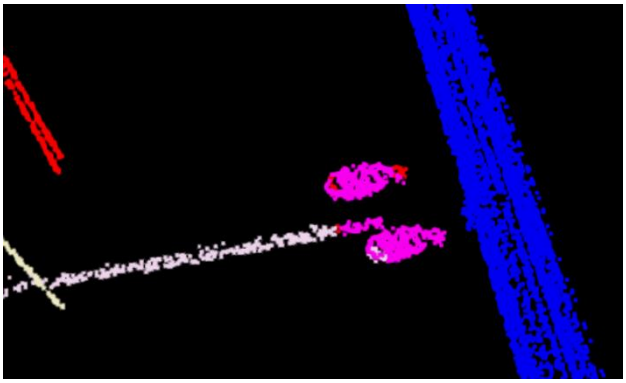

Figure 25 Two Moved insulators

6.2.2 Confusion Matrices

When looking at the visualizations of the segmented point clouds we can see that the model struggles when the insulator is moved a large distance. This can also be seen in the corresponding confusion matrices.

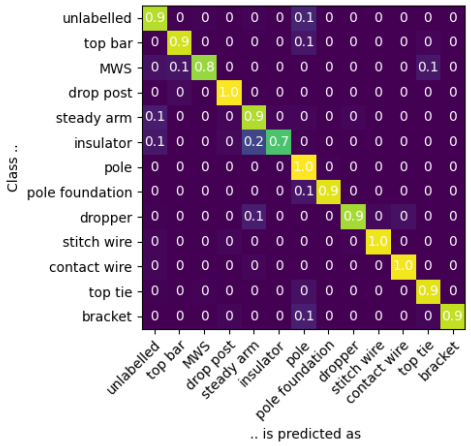

Figure 26 Confusion Matrix Large Movement

The confusion matrix corresponding to the point cloud visualized in Figure 24 can be seen in Figure 26. As you can see, where in the original point cloud 90% of the insulators were labeled correctly, now only 70% of the insulators were labeled correctly. Since only one insulator was moved, the others were still labeled correctly. 10% was segmented as “unlabeled” and 20% was segmented as “steady arm”. The moving of the insulator did not have impact on the segmenting of the other objects in the point cloud. This can be confirmed when looking at the mIoU, which was 0.84. This is only 0.02 lower than the mIoU of the original point cloud.

The mIoU of all the segmented point clouds where the insulator was moved ranged from 0.86, where the insulator was moved only slightly to 0.84, where the insulator was moved a larger distance.

6.3 Adding Holes in Pole

The results of the segmentation of the point cloud with 8 holes in one of the poles can be seen in Figure 27. The result of the point cloud in which 80 holes were added in one of the poles can be seen in Figure 28. The model had no problem segmenting the point clouds in which holes were added in the poles. The number of holes made no difference, the model correctly segmented all the adapted poles.

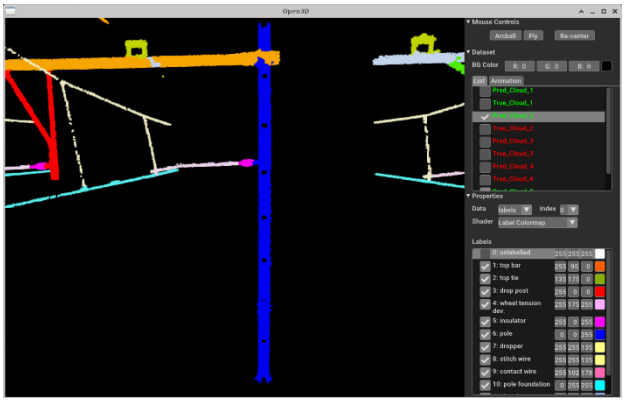

Figure 27 Pole with 8 Holes

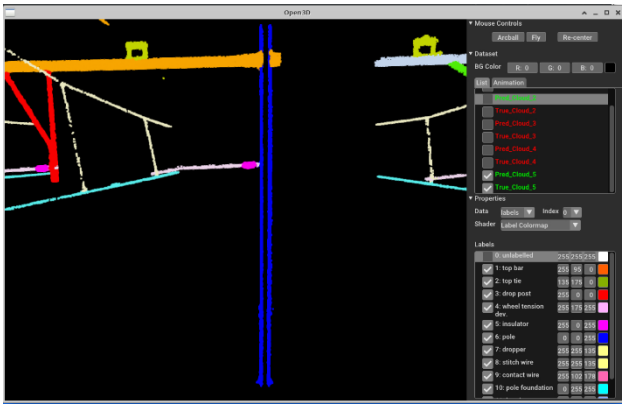

Figure 28 Pole with 80 Holes

Since the model segmented the objects correctly the confusion matrices did not change with respect to the original confusion matrix from Figure 12. This is confirmed by the mIoU of each of the adapted point clouds, which all had a value of 0.86, just like the original point cloud.

7. CONCLUSION

When comparing the visualizations of the adapted and original segmented point clouds we can see that the model relies on the (relative) location of the object in order to segment them. The model was not able to correctly label the insulators that had been moved a large distance from their original location, regardless of the direction. However, the model did perform well when the insulators were only moved a small distance. The mIoU of the point clouds in which the insulator was moved a large distance dropped from 0.86 to 0.84. This means that moving one insulator has no effect on the segmenting accuracy of the other objects in the point cloud.

Furthermore, we can see that changing the shape of the insulator into a cylinder has no effect on the performance of the model. This means that the model does not rely solely on the shape of the object in order to segment it.

Adding holes in one of the poles also has no effect on the performance of the model. The mIoU remained the same.

All in all, the model seems to be quite robust. A lot of changes did not have effect on the performance of the model. The model does perform worse when the objects are moved. But in a catenary arch the object will always be around the same area.

8. LIMITATIONS AND FUTURE WORK

The shape of the insulator was only changed into a perfect cylinder so it cannot be concluded that the model does not take the shape of the object into account when segmenting the point clouds. It would be interesting to see how the model performs when completely different shapes are used.

It would also be interesting to see how the model performs when the objects are rotated with respect to each other. Would the model still be able to segment the insulators correctly if they were rotated 90 degrees around the z-axis?

In this research only one object was adapted each time. The model could behave differently when for example all the insulators were moved.

9. ACKNOWLEDGEMENTS

I would like to thank Faizan Ahmed for all the guidance, help and supervision during this research. Furthermore, I would like to thank Bram Ton and Jeroen Linssen for their guidance, feedback and weekly meetings.

10. REFERENCES

[1] Centraal Bureau voor de Statistiek. 2021. *Hoeveel wordt er met het openbaar vervoer gereisd?* Centraal Bureau voor de Statistiek https://www.cbs.nl/nl-nl/visualisaties/verkeer-en-vervoer/personen/openbaar-vervoer. [Accessed 21 1 2022].

[2] Chen, X., Jiang, K., Zhu, Y., Wang, X., & Yun, T. 2021 *Individual tree crown segmentation directly from uav-borne lidar data using the pointnet of deep learning.* Forests.

[3] Feng, D., Haase-Schutz, C., Rosenbaum, L., Hertlein, H., Glaser, C., Timm, F., Wiesbeck, W., Dietmayer, K. 2021. Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. *IEEE Transactions on Intelligent Transportation Systems*, 22(3), 1341-1360.

[4] Gao, H., Cheng, B., Wang, J., Li, K., Zhao, J., & Li, D. 2018. Object classification using CNN-based fusion of vision and LIDAR in autonomous vehicle environment. *IEEE Transactions on Industrial Informatics*.

[5] Guo, W. 2020. Explainable artificial intelligence for 6G: Improving trust between human and machine. *IEEE Communications Magazine*, 58(6), 39-45.

[6] Haralick, R. M. & Shapiro, L. G. 1985. Image segmentation techniques. *Computer Vision, Graphics, & Image Processing*

[7] Jayakumari, R., Nidamanuri, R. R., & Ramiya, A. M. 2021. Object-level classification of vegetable crops in 3D LiDAR point cloud using deep learning convolutional neural networks. *Precision Agriculture*, 22, 1617-1633.

[8] Li, Y. & Ibanez-Guzman, J. 2020. Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems. *IEEE Signal Processing Magazine*, 37(4), 50-61.

[9] Lin, S., Xu, C., Chen, L., Li, S., & Tu, X. 2020. LiDAR point cloud recognition of overhead catenary system with deep learning. *Sensors (Switzerland)*, 20(8), 2212.

[10] Open3D. 2021. Open3D-ML. http://www.open3d.org/docs/release/open3d_ml.html. [Accessed 23 1 2022].

[11] Qi, C. R., Su, H. , Mo, K. & Guibas, L., 2016. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *CoRR*, 2, 19.

[12] Qi, C. , Yo, L. , Su, H. & Guibas, L. , 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *CoRR*, 1, 14.

[13] Shen, D., Wu, G., & Suk, H., 2017. Deep learning in medical image analysis. *Annu Rev Biomed Eng.* 19, 221-248.

[14] Shin, D. 2021. The effects of explainability and causability on perception, trust, and acceptance: Implications for explainable AI. *International Journal of Human-Computer Studies*, 146

[15] Stewart, M. 2020. Guide to Interpretable Machine Learning. *Towards Data Science*, https://towardsdatascience.com/guide-to-interpretable-machine-learning-d40e8a64b6cf [Accessed 12 1 2022]

[16] Ton, B. . Ahmed, F. & Linssen, J, 2022. Semantic segmentation of railway catenary arches. *ISPRS Journal of Photogrammetry and Remote Sensing (Preprint).*

[17] University of Twente. 2021. Jupyter wiki https://jupyter.wiki.utwente.nl/. [Accessed 23 1 2022].

[18] Xu, Y., Zhu, M., Li, S., Feng, H., Ma, S., & Che, J., 2018. End-to-end airport detection in remote sensing images combining cascade region proposal networks and multi-threshold detection networks. *Remote Sensing*, 10(10), 1516.