Are classical features still relevant in the era of deep learning?

JESSE SNOIJER, University of Twente, The Netherlands

A deep-learning AI model for point clouds often requires a vast amount of training data, which is not always available or requires a time-expensive process to create. To reduce the amount of data needed, the complexity of the model can be decreased. To reduce the loss of accuracy by doing this, it is investigated if appending classical features to the data increases the accuracy for a model with low complexity. Multiple tests will be performed where data with and without classical features appended is compared. Also, models with different complexity are compared. Here we show that appending these classical features does increase the test accuracy for a deep-learning AI model with low complexity by 20%. It is also shown that decreasing the complexity of the model and appending classical features prevents the f1-score from dropping 22.4%. These findings could possibly translate to other kinds of data as well and solve the problem of not having enough data available. Because a less complex model is used, this also lowers the computational demand which speeds up training and has a lower power usage.

Additional Key Words and Phrases: AI, features, computer science, deep learning, point cloud

1 INTRODUCTION

The Netherlands is covered with about 7 thousand km of train tracks. These tracks are some of the busiest tracks in the world. This means maintaining the railway environment and corresponding structure is extremely important to prevent accidents and delays. Strukton Rail is a subcontractor of ProRail that inspects the rail infrastructure and detects where maintenance is needed.

Strukton Rail would like to work with a digital twin of the rail environment. A digital twin will ease the maintenance of the track. While for new tracks, this digital twin exists and is ready to use, old tracks do not have this. Strukton wants to create a digital twin by making use of point cloud data. These are sets of spatial data points captured by 3D scanning techniques such as lidar. These point clouds contain millions of data points, resulting in 3D representations of the railway environment.

Strukton Rail asked AMI Saxion if it was possible to localise objects in the rail structure. This has been an ongoing project of Saxion since 2020 and multiple students and researchers have been working on this problem [4, 14, 15]. These researchers have proven that semantically segmenting point clouds is possible and they build an AI model which could do it. However, this AI model requires a lot of epochs before it reaches adequate accuracy. This means the model needs to train on a lot of data. Manually labelling the data is a time-expensive process and Saxion would rather have a model that requires less data for the same performance.

The current AI is a deep-learning PointNet++ model [8]. This model learns the features of point clouds and based on that, is able to create a semantic segmentation. Features could of course also be crafted by hand instead of the AI model discovering them. Feeding a lighter AI model the by hand chosen features might reduce the number of epochs needed and thus make the AI model faster and require less training data. This means that features have to be selected carefully so they add value to the model and do not confuse the model with non-relevant data.

2 RESEARCH QUESTION

This paper tries to answer the problem presented in the introduction with the following research question: *What is the impact on the f1-score when classical features are appended to the input data of a deep-learning AI model with a low complexity for classifying point clouds?* It will attempt to answer the question by using several sub-research questions. These are listed below:

Q1: What features are available for a point cloud?
Q2: What feature achieves the highest f1-score?
Q3: When is adding features the most useful?

For the first sub-question, point clouds with only x, y, and z coordinates are considered. This makes the available features limited since some features require more parameters, such as colour or light intensity. Extracting the correct features is essential for the results of this research.

For the second sub-question, an investigation has to be done into what feature found in the previous sub-question achieves the highest accuracy. This makes sure that the right features are selected for this research. The best two features will be selected and a combination of these two features is considered as well.

Finally, for the third sub-questions, different models with a different amount of layers and thus complexity are considered. The features found in the previous sub-questions are fed into these models and the resulting data has to be analyzed. From the data can be read when appending classical features has the most impact on the test results in comparison to the baseline.

3 RELATED WORK

For this research, the focus lies on features for point clouds available in Python libraries. The reason for choosing Python is the PyTorch-geometric library [3]. PyTorch-geometric makes it possible to write and train Graph Neural Networks. The library can not only be used for learning on arbitrary graphs but also on 3D meshes or point clouds, which is what is needed for this research.

PointNet++ is a "deep hierarchical feature learning AI for point sets in metric space" [8]. But for the purpose of this paper, some of the features will be manually crafted.

For extracting the features, some related research has been done. The Point Cloud Library (PCL) [12] has a variety of features available. Most of these features come from research papers and thus it can more easily be identified if the feature is useful for the data. This library is written for C++, but with the help of the Python library pclpy [1], it can be used in Python as well.

TScIT 39, July 7, 2023, Enschede, The Netherlands

Table 1. Class distribution of the ModelNet-10 dataset.

Class	Train count	Test count
bathtub	106	50
bed	505	101
chair	889	101
desk	200	86
dresser	200	86
monitor	465	100
night_stand	200	86
sofa	680	100
table	392	100
toilet	344	101

The data used for training and testing the model comes from the publicly available dataset ModelNet-10 [16]. This dataset consists of point clouds which can be used for classification. This dataset is created for another research and made available for academic research. The set consists of point clouds of 10 different classes: a bathtub, a bed, a chair, a desk, a dresser, a monitor, a nightstand, a sofa, a table and a toilet. The set contains a total of 4899 files split into a test and train set. This set is also easily accessible via the before mentioned library PyTorch-geometric.

4 METHODOLOGIES

4.1 The model input

The input of the model are point clouds with 10 different classes from ModelNet-10 as described in the section "Related Work". The amount of files used for the model to train and test on per class is shown in table 1 These point clouds consist of points with 3 position dimensions. The files are down-sampled to 128 points and normals are computed for these points. This gives the points a fourth dimension. The test set will be rotated along all three axes before downsampling. Then finally for all down-sampled files, features are extracted by using the before mentioned PCL Library. The data is fed into the model in batches of 10. The train set will be used for training and the test set will be used for validating the results. These validation results are stored and used for answering the research question.

4.2 The model

The PointNet++ model is created by following the steps presented by M. Fey [2]. For this paper, we will use the model that accounts for a rotation of the data. The seeds presented in the steps by M. Fey are kept the same. Inserting features in this model requires only a minor change to the code presented in the tutorial. The model needs to accept the features as an argument and this argument is forwarded into the layers accordingly.

A small optimisation has been made to the model as well. The learning rate has been set to 0.005 in the optimizer. This value provided the best test accuracy for the ModelNet-10 dataset after running a series of small tests.

The model is shown in figure 1. A point has three dimensions, on top of this a normal is calculated which adds a fourth dimension.

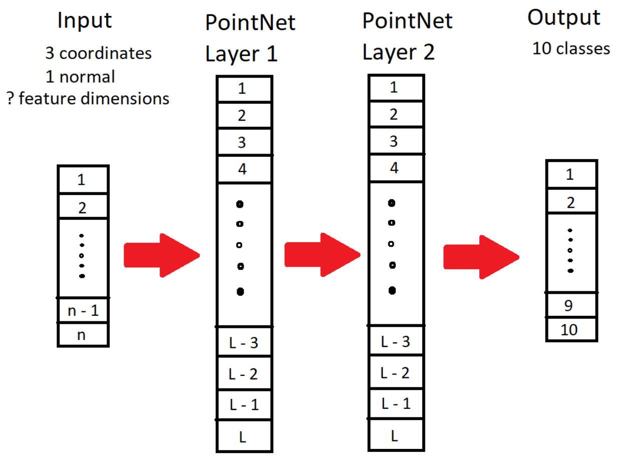

Fig. 1. A representation of the model used in this research paper.

Then the dimensions added by the features will be appended. These dimensions will go to the first PointNet layer. This layer will for most tests have a size of 32. This number was the default value. The "empty" dimensions are filled by the model itself since PointNet++ is a "deep hierarchical feature learning AI" [8]. The created model consists for most tests out of 2 PointNet layers of the same size. Finally, the data is transformed into a 10-dimensional linear layer. These 10 dimensions correspond with the 10 classes of data.

The model will run for 200 epochs. This number is chosen because the model seems to peek at around 150-175 epochs and 25 epochs are chosen to be added as a margin. This allows to gather data over the whole process of training the model.

4.3 Experiments

4.3.1 First sub-question. For this research, the features will be restricted to the ones provided by the Point Cloud Library. From the features available in this library, a list is created of all the features which require just point clouds with x, y and z coordinates. This list will be further filtered on the number of dimensions it adds to the data. A test will be performed to see what amount of dimensions benefit the model and those features will be used in the following questions.

4.3.2 Second sub-question. For this test, data will be plotted per epoch for all features found in the previous sub-question. These lines will be put into a single graph where the two best-performing features will be taken. From these two features, a combination will be made.

4.3.3 Third sub-question. To answer this research question, three different tests are done. The features discovered in the previous research question and the combination of those two are appended to the data in three different models. A model as shown in figure 1 with 1 PointNet layer, a model with 2 PointNet layers and a model with 3 PointNet layers. The results from these models are compared to investigate if the complexity of the model has an influence.

4.4 Visualizing the results

For every feature combination, the average f1-score of the runs is taken per epoch. The f1-score is chosen instead of the accuracy because of the class imbalance in the ModelNet-10 dataset (table 1). The amount of runs per test is rather limited, this still results in a graph that jumps up and down a lot. To get to a more realistic graph, a sliding-window-average is used with a window size of 5 epochs. This results in a precise enough graph of which the results can be read. Next to the graph, a table showing three different metrics is produced. The first metric is the mean f1-accuracy during the entire training process. The second metric is the maximum f1-score and the third metric is the mean f1-score of the last 10 epochs.

5 RESULTS

This section of the paper will present the results of the research described in the methodologies section. Experiments are conducted as described in section 4.3. All experiments ran three times and present averaged results.

5.1 Features available for a point cloud

Filtering the Point Cloud Library on features available for point clouds with just x, y and z coordinates results in a list of 34 features. This list consists of features which add a single dimension till features which add 1980 dimensions to the data. To prevent testing all available features a test is done for 2 features which add a single dimension to the data and a feature which adds 33 dimensions to the data. The feature with 33 dimensions is chosen because the amount of dimensions lies close to the chosen amount of dimensions of the PointNet layers. The features used for this test are BoundaryEstimation, DifferenceOfNormals and FPFHEstimation. These features add 1 dimension, 1 dimension and 33 dimensions respectively. For the test with FPFHEstimation, a model with PointNet layers consisting of 40 dimensions is used. The reason is that the model has more dimensions in the PointNet layer than as input since this proved to give better results. In this case, There are three "free" dimensions for the model to fill since the points have 3 dimensions and a normal dimension.

BoundaryEstimation results in 1 for points which are believed to lay on a boundary of an object and 0 otherwise. Removing all the points which resulted in 0, would leave the outline of an object. This feature requires a point cloud of the data and a point cloud of normals as input data. For the angle, the default value ($\pi/2.0$) is used. This results in a point cloud with for every point a single dimension which is either 1 or 0.

DifferenceOfNormals [5] compares the normal of a small area with the normal of a larger area. The difference of these normals provides a scale-based feature, somewhat like the Difference of Gaussians in image processing, but instead on surfaces. With these 2 normals, the curvature of an object can be captured. This difference between these normals is presented as a single dimension.

FPFHEstimation [10, 13] estimates the Fast Point Feature histogram (FPFH) descriptor for a given point cloud dataset containing points and normals. This FPFH descriptor is a fast version of the PFH descriptor. This is a histogram with 33 dimensions with various features for points presented in the research [9] and [11]. The features are "informative enough to differentiate between the underlying primitive geometric surfaces they represent" [10].

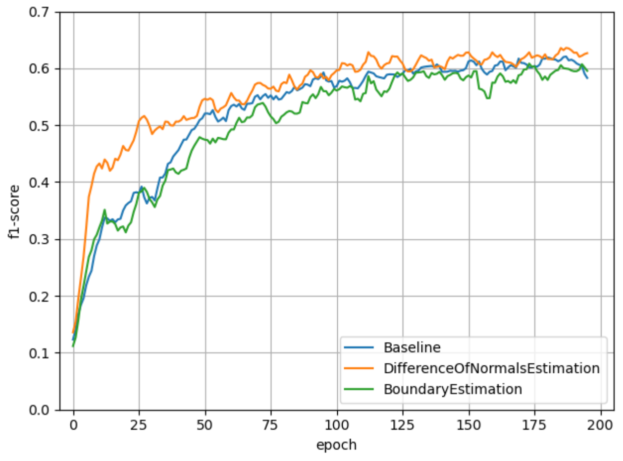

Fig. 2. This graph shows the f1-score over the number of epochs. The model consists of 2 layers and 32 features. BoundaryEstimation and DifferenceOfNormals are the baseline + 1 dimension gained from the feature.

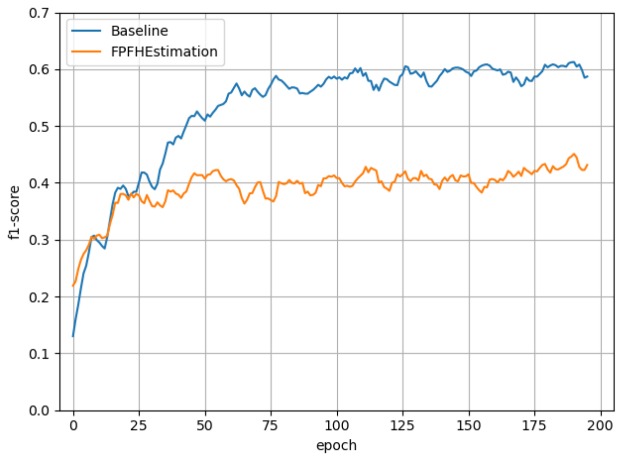

Fig. 3. This graph shows the f1-score over the number of epochs. The model consists of 2 layers and 40 features. FPFHEstimation is the baseline + 33 dimensions gained from that feature.

The test performed resulted in the graphs in figure 2 and figure 3. It can clearly be seen that FPFHEstimation in figure 3 follows the baseline, until epoch 20. Then the baseline continues rising in f1-score while the feature line stays fluctuating around 40%.

The graph in figure 2 seems to provide more promising results. In this graph, the data with the feature BoundaryEstimation appended produces slightly worse results in comparison with the baseline. The feature DifferenceOfNormals stays on top of this baseline for the whole 200 epochs this test was run.

From this test, it can be concluded that features which fill most dimensions have a negative influence on the f1-score of the model.

Table 2. Features to be tested

Feature	Amount of dimensions
BoundaryEstimation	1
DifferenceOfNormalsEstimation	1
RSDEstimation	2
MomentInvariantsEstimation	3
LinearLeastSquaresNormalEstimation	3
NormalEstimation	1
IntegralImageNormalEstimation	1
PPFEstimation	5
NormalBasedSignatureEstimation	12
GRSDEstimation	21

Features that only add 1 or 2 dimensions to the data and leave enough dimensions open for the model to learn, seem to either follow the baseline or have a positive effect on the f1-score of the model. This makes further research into features which add 33 or more dimensions not interesting for this research. This leaves the features listed in table 2.

Further investigation into these features makes IntegralImageNormalEstimation not applicable for this research because it requires organized point clouds and the point clouds provided by ModelNet-10 are not organized. PPFEsimation returns features for each point pair in the given point cloud. This is a long list of data which can not be traced back to a point. This makes the feature not compatible with the data structure of PyTorch-geometric. NormalBasedSignatureEstimation produces for an unknown reason no results and thus can not be used for this research as well. GRSDEstimation returns the error "Leaf size is too small for the input dataset. Integer indices would overflow". This error can not be traced back because of the Python bindings. Also, the documentation of the Point Cloud Library does not provide a solution. This means the feature is not usable as well.

The two features for calculating normals do not produce results as well. At first, the features seem promising but the model gets for both features stuck on an accuracy of 0.0551 and the f1-score stays at 0.0057. The reason for this is unknown. Changing the learning rate to a lower value does not seem to help. So the features LinearLeastSquaresNormalEstimation and NormalEstimation are not usable for this research

This leaves the top 4 features of the table 2: BoundaryEstimation, DifferenceOfNormalsEstimation, RSDEstimation, and MomentInvariantsEstimation.

RSDEstimation [6, 7] estimates the Radius-based Surface Descriptor (minimal and maximal radius of the local surface's curves) for a given point cloud dataset containing points and normals. This feature results in 2 dimensions, one for the minimal and one for the maximal radius.

MomentInvariantsEstimation estimates the 3 moment invariants (j1, j2, j3) at each 3D point. This feature appends 3 dimensions to the data.

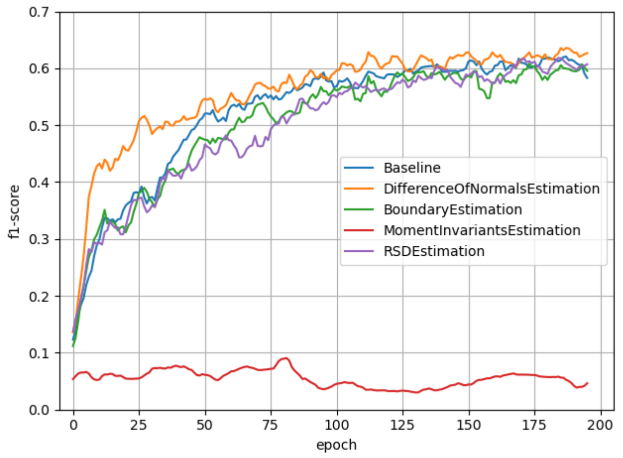

Fig. 4. This graph shows the f1-score over the number of epochs. The model consists of 2 layers and 32 features. The items in the graph are different features which are appended to the input data of the model and the baseline.

Table 3. Metrics for the f1-scores gathered in sub-questions 2

Feature	mean	max	mean last
Baseline	0.521	0.656	0.597
DifferenceOfNormalsEstimation	0.554	0.696	0.626
BoundaryEstimation	0.502	0.657	0.598
RSDEstimation	0.497	0.667	0.596
MomentInvariantsEstimation	0.054	0.131	0.044

5.2 The combination of features with the highest f1-score

The results are presented in figure 4. This figure shows that MomentInvariantsEstimation achieves poor results. The f1-score seems to fluctuate around 5%. The features BoundaryEstimation and RSDEstimation achieve until epoch 38 the same results as the Baseline. However, from epoch 38 onwards, the Baseline keeps achieving a higher f1-score. The feature DifferenceOfNormals achieves a 13% higher f1-score at epoch 12. But from epoch 50 onwards, they achieve about 2% more f1-score than the baseline.

The metrics are presented in table 3. Clearly, the DifferenceOfNormalsEstimation is the best-performing feature. For the second best, it would be either BoundaryEstimatoin or RSDEstimation. BoundaryEstimation has a higher overall mean and mean-last-10 f1-score, but RSDEstimation has a higher maximum f1-score. Since the maximum could be an outlier and the mean-last-10 is deemed most important, BoundaryEstimation is chosen as the second best feature.

5.3 When is adding features most useful

The results for this experiment are displayed in figures 5, 6 and 7 and in the tables 4, 5 and 6. In the figure 5, there is a large difference in the f1-score between the baseline and the features. With a model consisting of two layers, that difference becomes smaller,

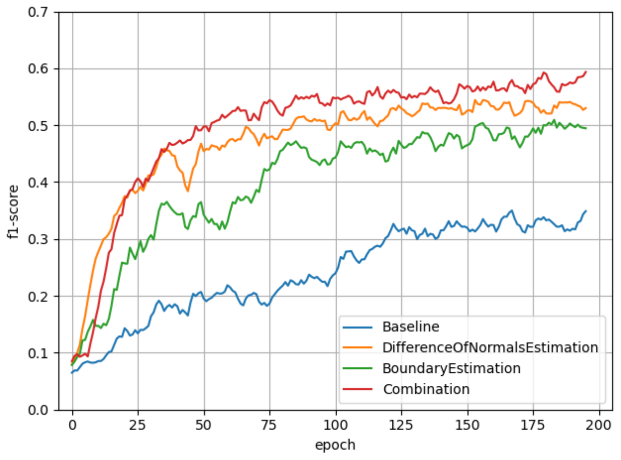

Fig. 5. This graph shows the f1-score over the number of epochs. The model consists of 1 layer and 32 features. DifferenceOfNormalsEstimation is the baseline + 1 dimension gained from that feature. BoundaryEstimation is the baseline + 1 dimension gained from that feature. And Combination is the baseline + 2 dimensions gained from the previous two features.

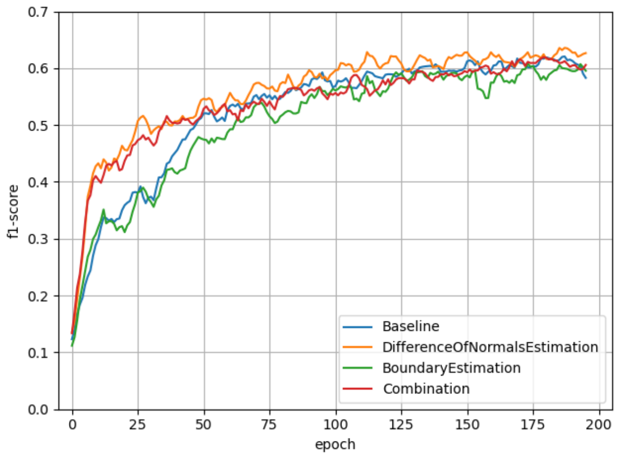

Fig. 6. This graph shows the f1-score over the number of epochs. The model consists of 2 layers and 32 features. DifferenceOfNormalsEstimation is the baseline + 1 dimension gained from that feature. BoundaryEstimation is the baseline + 1 dimension gained from that feature. And Combintation is the baseline + 2 dimensions gained from the previous 2 features.

Table 4. Metrics for the f1-scores gathered in sub-questions 3, 1 layer model

Feature	mean	max	mean last
Baseline	0.241	0.458	0.333
DifferenceOfNormalsEstimation	0.467	0.608	0.534
BoundaryEstimation	0.398	0.558	0.497
Combination	0.493	0.627	0.583

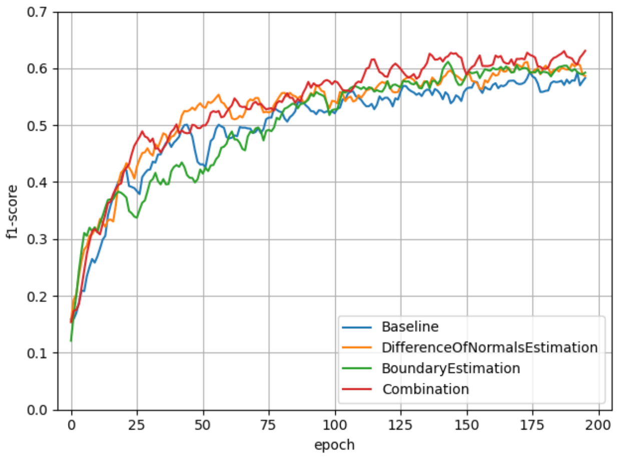

Fig. 7. This graph shows the f1-score over the number of epochs. The model consists of 3 layers and 32 features. DifferenceOfNormalsEstimation is the baseline + 1 dimension gained from that feature. BoundaryEstimation is the baseline + 1 dimension gained from that feature. And Combination is the baseline + 2 dimensions gained from the previous two features.

Table 5. Metrics for the f1-scores gathered in sub-questions 3, 2 layer model

Feature	mean	max	mean last
Baseline	0.521	0.657	0.598
DifferenceOfNormalsEstimation	0.555	0.697	0.627
BoundaryEstimation	0.503	0.657	0.598
Combination	0.535	0.670	0.605

Table 6. Metrics for the f1-scores gathered in sub-questions 3, 3 layer model

Feature	mean	max	mean last
Baseline	0.496	0.662	0.581
DifferenceOfNormalsEstimation	0.522	0.675	0.597
BoundaryEstimation	0.501	0.670	0.594
Combination	0.534	0.676	0.624

but appending the classical features makes the f1-score of the best-performing feature in the end, 2,9% above the baseline. Then with three PointNet layers, the f1-score of the best-performing feature in the last epochs is 3.3% above the f1-score of the baseline. For the models with 1 and 3 layers, the best-performing feature is the Combination of DifferenceOfNormalsEstimation and BoundaryEstimation. For the model with 2 layers, the best-performing feature is DifferenceOfNormalsEstimation.

Looking at the number of epochs and the difference in f1-score between the baseline and feature-lines. It can be seen that for the 1-layer model and the 2-layer model, appending features to the data results at the beginning of the training stage for a larger advantage compared to the end of the training stage.

If we compare for each graph the differences between the three models for the best-performing feature and the baseline. The difference for the best-performing feature in the last 10 epochs between model 3 and 2 is an increase of 0.4% and from the 2-layer to the 1-layer model is a decrease of 4,4%. While for the baseline these differences are an increase of 1,7% and a decrease of 26,8%.

6 DISCUSSION

From the results in the third sub-questions, it can be seen that appending features increases the f1-score of the model in all test cases. This would mean that appending features could potentially always benefit the model. The most significant difference is between models of different complexity. When no features are appended to the data, decreasing the complexity of a model can decrease the f1-score significantly. However, appending features to the data and decreasing the complexity of the model only gives a small decrease in the f1-score. In the case of this research, the differences are 26,8% in comparison to 4,4%. This is a significant change in results.

7 CONCLUSION

On the first sub-question "*What features are available for a point cloud?*" is concluded that features which fill most dimensions do not have a positive effect on the f1-score of the model. Features that only add 1 or 2 dimensions to the data and leave enough dimension for the model to fill, seem to have a positive effect on the f1-score of the model. From the second sub-question, "*What features achieves the highest f1-score?*" is concluded that with this PointNet++ model, ModelNet-10 dataset and available features, BoundaryEstimation, DifferenceOfNormals and the combination of both seem to achieve the highest f1-score. On the final sub-question, "*When is adding features most useful?*" is concluded that the models with a low complexity benefit the most from appending features.

With the results from the sub-questions, the research question "*What is the impact on f1-score when classical features are appended to the input data of a deep-learning AI model with a low complexity for classifying point clouds?*" can be answered. The impact on the f1-score is that this increases when classical features are appended to the data. How much the f1-score increases depends on the complexity of the model. The test results also indicate that for a model with low complexity, appending features could increase the f1-score by 20%. And if the complexity of a model needs to be decreased, appending features would decrease the f1-score by just 4,4% instead of 26,8% when features are not used.

8 LIMITATIONS AND FUTURE WORK

The first limitation of this work is that not a lot of time is spent optimizing the model. The learning rate variable has been tweaked but not thoroughly tested. Optimizing the model in different ways may affect the f1-score of the results and thus might influence the results.

The second limitation would be that the number of dimensions for the PointNet layer is set to 32 except for 1 test. The effect of changing this to a higher or lower value is unknown and might influence the results of this research as well. The number of dimensions added by classical features is not thoroughly tested as well. The test performed is between adding 1, 2, 3 or 33 dimensions to the data. But an optimum of dimensions can not be found in the tests performed for this research.

A third limitation is that a sliding-window-average with a window size of 5 is used. This average makes the graphs more readable and thus makes it easier to deduct conclusions from the data. However, this sliding-window-average makes the data less precise. If the data points follow a more logical sequence, the necessity for a sliding-window average can be avoided. This might be possible by performing tests more often than was done, this is left as future work.

Some more future work would be changing the dataset ModelNet-10 to the data from Strukton. For achieving this, the model should be changed such it is made for segmentation and not classification. An optimisation would be to construct the features without the help of a library. In the current model, data is constantly converted from one library to the other. This costs unnecessary computation time.

REFERENCES

[1] David Caron. 2020. pclpy. https://anaconda.org/davidcaron/pclpy.

[2] Matthias Fey. 2022. *Point Cloud Classification.* https://colab.research.google.com/drive/1D45E5bUK3gQ40YpZo65ozs7hg5l-eo_U?usp=sharing

[3] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds.*

[4] J. Hentschel. 2023. Point cloud segmentation via active learning in the context of railway infrastructure. http://essay.utwente.nl/94542/

[5] Yani Ioannou. 2010. *Automatic urban modelling using mobile urban lidar data.* Ph. D. Dissertation.

[6] Zoltan Marton, Dejan Pangercic, Nico Blodow, and Michael Beetz. 2011. Combined 2D–3D categorization and classification for multimodal perception systems. *International Journal of Robotic Research - IJRR* 30 (Oct. 2011), 1378–1402. https://doi.org/10.1177/0278364911415897

[7] Zoltan-Csaba Marton, Dejan Pangercic, Nico Blodow, Jonathan Kleinehellefort, and Michael Beetz. 2010. General 3D modelling of novel objects from a single view. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems.* 3700–3705. https://doi.org/10.1109/IROS.2010.5650434 ISSN: 2153-0866.

[8] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. https://doi.org/10.48550/arXiv.1706.02413

[9] Radu Rusu, Zoltan Marton, Nico Blodow, and Michael Beetz. 2008. Learning Informative Point Classes for the Acquisition of Object Model Maps. 643–650. https://doi.org/10.1109/ICARCV.2008.4795593

[10] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. 2009. Fast Point Feature Histograms (FPFH) for 3D registration. In *2009 IEEE International Conference on Robotics and Automation.* 3212–3217. https://doi.org/10.1109/ROBOT.2009.5152473 ISSN: 1050-4729.

[11] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. 2008. Aligning point cloud views using persistent feature histograms. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems.* 3384–3391. https://doi.org/10.1109/IROS.2008.4650967 ISSN: 2153-0866.

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

[13] Radu Bogdan Rusu, Andreas Holzbach, Nico Blodow, and Michael Beetz. 2009. Fast geometric point labeling using conditional random fields. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems.* 7–12. https://doi.org/10.1109/IROS.2009.5354763 ISSN: 2153-0866.

[14] Nils Rutgers. 2022. Point cloud based semantic segmentation for catenary systems using deep learning : Compressibility of a PointNet++ network. http://essay.utwente.nl/92901/

[15] Z.J. Vieth. 2022. Point cloud classification and segmentation of catenary systems. http://essay.utwente.nl/89565/

[16] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 2015. 3DShapeNets: A Deep Representation for Volumetric Shapes. In *Proceedings of 28th IEEE Conference on Computer Vision and Pattern Recognition (CVPR2015).* Santiago, Chile.