Use of network features to improve 3D object classification

Gal Petkovšek
gp19194@student.uni-lj.si
Faculty of computer science,
University of Ljubljana Večna pot 113
SI-1000 Ljubljana, Slovenia

Janez Božič
jb1236@student.uni-lj.si
Faculty of computer science,
University of Ljubljana Večna pot 113
SI-1000 Ljubljana, Slovenia

Marija Marolt
mm7522@student.uni-lj.si
Faculty of computer science,
University of Ljubljana Večna pot 113
SI-1000 Ljubljana, Slovenia

ABSTRACT

In our work we tackle a problem of 3D object classification, which is a task traditionally closer to computer graphics rather than network analysis. There are several different approaches for solving this problem including deep learning, topological data analysis, graph theory *etc.* We use the Mapper algorithm that transforms the point cloud into a graph, which simplifies the data while obtaining the key properties of the structure. This algorithm is already used to solve such a task however, the features extracted from the graph were very limited. The novelty that we introduce is the calculation of some network properties on such graphs which are used for classification. The results show that the models have better classification accuracy scores when using network analysis attributes in addition to attributes from topological analysis however, it is challenging to determine exactly which features will perform well for a classification of objects.

KEYWORDS

network analysis, 3D object classification, Mapper algorithm, feature selection

ACM Reference Format:
Gal Petkovšek, Janez Božič, and Marija Marolt. 2022. Use of network features to improve 3D object classification. In *Proceedings of Student Computing Research Symposium (SCORES'22).* ACM, New York, NY, USA, 4 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

3D object recognition is one of the most challenging fields in object recognition community. The goal of the task is for a model to differentiate between different 3D objects that are presented to it in a certain format.

In 3D computer graphics a polygon mesh is a collection of vertices, edges and faces that defines a polyhedral object [1] (however in this work we consider only vertices positioned in a 3D space - from now on this will be referred to as point cloud). Because the process of scanning objects and saving them in such format has become very common, a need has developed to be able to automatically classify and visualise such data. An example can be seen on Figure 1. Given points together can human easily discern silhouette as a chair. It takes a computer quite some time to process all the points and connect them intuitively, so we need to extract the necessary data from the point cloud, train a model, and then make a prediction.

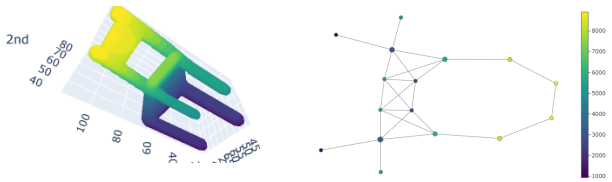

Figure 1: Chair point cloud and its network made with Mapper algorithm

The first and one of the most fundamental challenges of this task is choosing the best representation of such data to be inputted into the models for classification. One approach is to transform the data into a set of 2D images and feed them into the model [11] and another would be to just use the whole set of points [8]. In this work we used a Mapper algorithm [10] to transform the set of points into a compact network. An example of such transformation can be seen on Figure 1, where we can on the left side observe the original point cloud and the extracted network on the right. The idea is to represent the object in the simplest way possible, while still keeping the essential information required for the classification. From this relatively simple representation of an object we can compute some topological properties of the structure or some properties from network analysis that would be useful for classification. The options for attributes are quite extensive and in this work we limit ourselves to three topology features and 14 network analysis features. We then test different subsets of features to determine which combination is most effective for classification.

There are a lot of different approaches of either describing the data or doing the actual classification of 3D objects. A paper from 2016 [8] describes an approach that takes all the data points and feeds them to a neural network. Another approach [11] consider the data as a set of 2D images. Another research [4] uses a technique called topology matching on graphs. Some authors [3, 6, 11, 13] also describe the use of Convolutional Neural Networks (CNNs) for solving such problems. Instead of CNN we will be using Support vector machine (SVM), as in the article [5] Anupama *et. al.* The Mapper algorithm that we are using to transform the data was first described in the article by Singh *et. al* [10] however, for classification they limited themselves to the outputs of the Mapper algorithm while we explore the network further.

SCORES'22, October 6, 2022, Ljubljana, Slovenia

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

2 METHODS

2.1 Problem definition

We have tested our approach on a dataset that consists of four classes (table, chairs, octopuses and spiders). We tackle this problem as a binary classification (deducing whether a sample is *e.g.* a table or not). More precisely we compare four different classification tasks (one binary classifier for every one of the four classes) in order to determine how the use of network analysis features affect the performance of models on different target classes.

For every one of the four models we test every possible subset of features that are described in Section 2.4.

New models (with network theory features) are compared to models with one homology feature.

2.2 Data

The data is obtained from McGill 3D Shape Benchmark. [14] It is a repository of point clouds for testing 3D shape retrieval algorithms. We use the mesh format where each object's surface is provided in triangulated form. To provide versatility but also enough models in every class, one class is containing goal object models and second class containing three other objects models. There are point clouds of 21 tables, 23 chairs, 25 spiders and 31 octopuses taken from McGill 3D Shape Benchmark repository (together 100 samples). Examples of the referred samples can be seen on Figure 2.

Figure 2: Examples of four types of objects in the database

2.3 Network construction

To create a network from data point cloud we use the Mapper algorithm. It is composed of three main steps: the filtering function, the cover function and the clustering function [12]. After some initial testing we chose the best working components. Example of the transformation is shown on figure 1.

The main purpose of the filtering function is to reduce the dimensionality of the data. In our case this function projects data points defined in a 3D space onto the xy plane.

The next step is to calculate groups of overlapping points with the help of the covering function. This function takes as input the filtered space, calculates overlapping groups with dividing and transform points from each group back to the original space. Overlapped groups are computed with division of each input dimension by 3 equal-length intervals with fractional overlap of 0.15.

Finally, we apply a clustering algorithm on each group and calculate an undirected graph. This is done with a clustering algorithm that is able to determine the number of clusters in a dataset automatically or by manually defining it. For this purpose we use the DBSCAN [7] algorithm where the maximum distance between two points for one to be considered as in the neighborhood of the other is 10 using Chebyshev distance. Each cluster generates a node and the edges between them are created if the two clusters share some points.

The resulting network contains a lot less nodes than there are points in the point cloud and is therefore more easily handled than the original data representation. An example of such network can be seen in Figure 1.

2.4 Attribute calculation

A novel practice in point cloud object recognition is applying different homology based attributes to the output of the mapper. A "continuous" shape is built on the top of the data in order to highlight the underlying topology or geometry. By using homology, we observe numbers of components, holes, and voids while adding connections between the points. We measure when connected components appear and when they merge, representing their lifetime. In our case we use persistence entropy (entropy of the points in a persistence diagram later referred to as h1), and amplitudes of persistent diagrams, which plot lifetimes of connected components, computed with Wasserstein or Bottleneck distance as a classification parameters (later referred to as h2 and h3). From network theory we included the following features:

(0) **Number of articulation points** (NA) - nodes that separate the notwork into multiple sub-networks.
(1) **Average degree of node in a network** ($\langle k \rangle$)
(2) **Network density** (ρ)
(3) **Average clustering coefficient** ($\langle C \rangle$)
(4) **Normalised number of nodes in top 10% closeness centrality** ***(number of nodes having 90% or more of maximum node closeness centrality divided by number of nodes)*** ($top10\_l^{-1}$)
(5) **Normalised number of nodes in top 10% betweenness centrality** ***(number of nodes having 90% or more of maximum node betweeness centrality divided by number of nodes)*** ($top10\_\sigma$)
(6) **Normalised number of cliques with 4 nodes** ***(number of cliques containing 4 nodes divided by number of nodes)*** (NClique)
(7) **Degree assortativity coefficient** (DAC)
(8) **Normalised number of leaves** ***(number of nodes with degree 1 divided by number of nodes)*** (Nl)
(9) **Maximum node closeness centrality** ($Max l^{-1}$)
(10) **Maximum node betweenness centrality** ($Max\sigma$)
(11) **Average node closeness centrality** ($Avg l^{-1}$)
(12) **Average node betweenness centrality** ($Avg\sigma$)
(13) **Number of Louvain Communities** (NLC)

Attributes such as number of articulation points, number of leafs or number of louvain communities tell us a lot about the articulation parts of the structure which could be a very valuable information for the model. Average degree of a node and network density provide information on how many edges formed in the network which in other words is how many of clusters obtained from the mapper algorithm described in Section 2.3 share some points. Features including node centralities contain information about the importance of specific nodes (either on average, maximum or just what proportion of nodes achieves the higher centralities). The idea

behind number of cliques and average clustering coefficient is that they could indicate if an object would contain some larger even surface that would be more fully connected.

Our goal is to determine the best combination of features, however since we only have a very limited number of samples, we do not consider too large subsets of features (on which the model would only overfit and produce meaningless results). Therefore we restrict the number of features to either 4 (if we also include some homology) or 6 otherwise.

We extract most of the measures from Python library NetworkX [9], and calculate the remaining with custom code using data from the library. Specifically average network clustering, number of articulation points, assortativity coefficient, number of louvain communities, betweenes and closeness centralities were calculated with NetworkX functions.

2.5 Learning

For modeling we use Support Vector Machines (SVM) implementation from SciKit Learn [2] library for Python with a linear kernel setting. Other hyperparameters of the model are set to their default values as specified in the SciKit Learn [2] documentation.

We evaluate our models with 10-fold cross validation to improve stability of the results.

For a metric we use AUC (area under the ROC curve) which is a metric ranging from 0 to 1 (in practice from 0.5 to 1 as any classifier scoring lower than 0.5 can be inverted, transitioning to AUC>0.5) with 1 being the most and 0 (in reality 0.5) being the best score. This metric was chosen since it shows how well the model can separate the samples into correct classes. It is calculated by first constructing the ROC curve, which shows the relation between true positive rate (TPR) and false positive rate (FPR). It is obtained by moving the threshold and for each value calculating the TPR and FPR. After all points are gathered, the AUC can be calculated.

2.6 Experimetal setup

Experiments are conducted on the data set of labeled point clouds, described in Section 2.2. As we have 4 different labels we conduct 4 independent experimets, each time choosing one label as positive (object belongs to that group) and the other labels as negative (object does not belong to the chosen group).

For each of those 4 experimets we first transform all the cloud points to smaller networks using the mapper algorithm. For each one of those we then extract the features (both topological and features form network analysis). Once we have the labeled feature vectors we perform a grid search over all feature subsets (that suffices the size limitations presented in Section 2.4). For each of these subsets we performe a 10-fold cross validation with the SVM model (each fold using 90 samples for training and 10 for testing). The results are then saved and analysed.

3 RESULTS

The goal of our research is to determine if the additional network features contribute to the scores and if so which features are the most beneficial to include.

To determine whether adding additional features is even beneficial for the predictions, we first take a look at the results of the best models (models built on subsets of features that performed best according to the AUC score). Combinations of attributes that achieve the highest score in our test cases are displayed below. In addition to the best performing subset of features and the corresponding scores we included the results for best performing homology for each use case (same results as can be seen in Table 1).

	tables	chairs	octopus	spiders	Average
h1	0.62	0.91	0.66	0.71	0.73
h2	0.48	0.80	0.48	0.77	0.63
h3	0.84	0.60	0.62	0.81	0.72
NA	0.50	0.48	0.50	0.48	0.49
$\langle k \rangle$	0.86	0.43	0.45	**0.79**	**0.63**
ρ	0.43	0.37	0.56	0.41	0.44
$\langle C \rangle$	0.24	0.36	0.27	0.33	0.30
$top10\_l^{-1}$	0.59	0.44	0.56	0.30	0.47
$top10\_\sigma$	0.74	**0.77**	**0.71**	0.37	**0.65**
NClique	0.35	0.42	0.46	**0.79**	0.51
DAC	**0.90**	0.44	0.46	0.39	0.55
Nl	0.62	0.35	0.48	0.49	0.49
$Maxl^{-1}$	**0.92**	0.43	0.37	**0.67**	0.60
$Max\sigma$	0.43	**0.58**	**0.57**	0.50	0.52
$Avgl^{-1}$	**0.92**	0.40	0.56	0.50	0.60
$Avg\sigma$	0.82	0.42	**0.69**	0.60	**0.63**
NLC	0.47	**0.49**	0.52	0.50	0.50

Table 1: The results of the model for each individual feature (only 1 feature used for classification).

- **Tables:** The best features are $Maxl^{-1}$, $Avgl^{-1}$ and h3. AUC score is 0.96 (AUC using only h3 was 0.84).
- **Chairs** The best features are NA, $\langle C \rangle$ and NLC in combination with either $top10\_l^{-1}$ and $Max\sigma$ or h2. AUC score is 0.96 (AUC using only h1 was 0.9).
- **Octopuses** The best features are NA, DAC and h3. AUC score is 0.85 (AUC using only h1 was 0.66).
- **Spiders** The best features are $top10\_\sigma$, Nl, $Maxl^{-1}$ and h2. AUC score is 0.89 (AUC using only h3 was 0.81).

The best performing features are greatly dependent on the domain (specific object we want to classify), which is also seen in Table 1. From our results of combination of best performing features we can see improvements over the baseline (using only homology for classification). Addition of network analysis attributes to the feature vector improves classification. However, based on the results above we cannot determine which features in general perform best.

To determine this we first viewed the scores of models using only single features to predict the class. Each column (classification problem) has the three best results bolded. As we can see, we could not point out a few features that would be significantly better than the others in all cases. The quality of each attribute is greatly dependant on the class that we are trying to predict. This can easily be explained as different objects tend to produce different networks and each has different properties that separate it from the rest. Some of the best performing features are $top10\_\sigma$, $Max\sigma$ and $Maxl^{-1}$ however, while most of them perform quite well on two classes they perform worse on the other two. More consistent

features that perform relatively well on most classes are for example $top10\_\sigma$ and $Avg\sigma$.

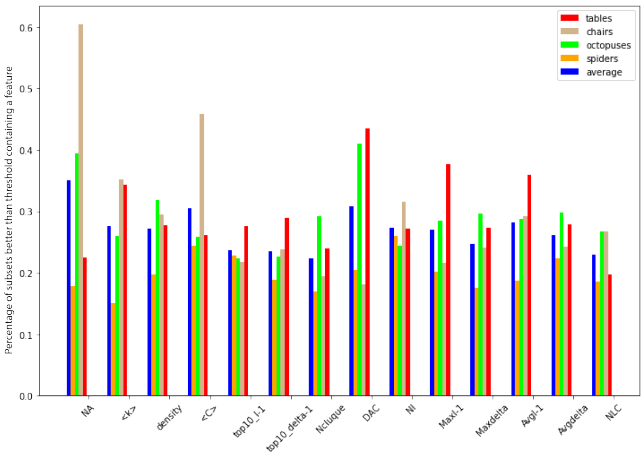

Figure 3: Chair point cloud and its network made with mapper algorithm

To further test the importance of features we set a cutoff point at the score of the model using the best performing homology feature (for each classification task) and only consider models with feature subsets that scored higher. For each feature we calculate in what percent of those models it appears. Those results are presented in Figure 3. We can observe again that the results mostly differ when it comes to the observed class. More preferable are features that perform well on all or most of the use cases. Such example would be $\langle C \rangle$, Nl, $Avgl^{-1}$ or $Avg\sigma$. The latter two are by themselves a relatively good features. $\langle C \rangle$ is not as successful on its own but is often present in the most successful subsets. It indicates how connected a network is therefore an objects with a more homogeneous body might be more connected than the ones with more articulation parts. The number of louvain communities could also be a good measure to determine out of how many parts the object consists since it also appears in many of the best subsets for all use cases.

NA is one of extreme examples that perform very well on two classes but quite worse on the others. Number of articulation points is greatly dependant on how many nodes there are in general and while in some cases it might be a very good indicator of how many articulation parts there are in others it might be deceiving as for example a long leg of a spider might have multiple articulation points. Such features (DAC is another such example) might also be taken into consideration when building a model however it is not guaranteed that they will perform well on all use cases.

4 CONCLUSION

Our experiments have shown that it is beneficial for a model for 3D object classification to use (in addition to topology features) network analysis features. However, it is difficult to determine which will perform best for a certain use case. We have tested our models on four different classification problems and the results have shown that while some attributes are indeed in general better than the others, for the majority of objects it is hard to determine which combination will perform best.

In future work we could expand our research by including a larger dataset (include more objects). We would also like to test more network features, and get an even better insight into what are the most beneficial attributes.

REFERENCES

[1] 2022. Polygon Mesh. https://en.wikipedia.org/wiki/Polygon_mesh
[2] David Cournapeau. 2022. SciKit Learn: Machine learning in Python.
[3] Kan Guo, Dongqing Zou, and Xiaowu Chen. 2016. 3D Mesh Labeling via Deep Convolutional Neural Networks. *ACM Trans. Graph.* 35, 1, Article 3 (dec 2016), 12 pages. https://doi.org/10.1145/2835487
[4] Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura, and Tosiyasu L. Kunii. 2001. Topology Matching for Fully Automatic Similarity Estimation of 3D Shapes. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 203–212. https://doi.org/10.1145/383259.383282
[5] Anupama Jawale and Ganesh Magar. 2019. Comparison of Image Classification Techniques : Binary and Multiclass using Convolutional Neural Network and Support Vector Machines. (12 2019).
[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger (Eds.), Vol. 25. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
[7] Jiirg Sander Martin Ester, Hans-Peter Kriegel and Xiaowei Xu. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.
[8] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2016. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *CoRR* abs/1612.00593 (2016). arXiv:1612.00593 http://arxiv.org/abs/1612.00593
[9] Aric Hagberg Pieter Swart Dan Schult. 2022. NetworkX: Network analysis library.
[10] Gurjeet Singh, Facundo Memoli, and Gunnar Carlsson. 2007. Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition. In *Eurographics Symposium on Point-Based Graphics*, M. Botsch, R. Pajarola, B. Chen, and M. Zwicker (Eds.). The Eurographics Association. https://doi.org/10.2312/SPBG/SPBG07/091-100
[11] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. 2015. Multi-view Convolutional Neural Networks for 3D Shape Recognition. *CoRR* abs/1505.00880 (2015). arXiv:1505.00880 http://arxiv.org/abs/1505.00880
[12] Guillaume Tauzin, Umberto Lupo, Lewis Tunstall, Julian Burella Pérez, Matteo Caorsi, Wojciech Reise, Anibal Medina-Mardones, Alberto Dassatti, and Kathryn Hess. 2021. giotto-tda: A Topological Data Analysis Toolkit for Machine Learning and Data Exploration. arXiv:2004.02551 [cs.LG]
[13] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. 2019. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)* 38, 5 (2019), 1–12.
[14] Juan Zhang, Kaleem Siddiqi, Diego Macrini, Ali Shokoufandeh, and Sven Dickinson. 2005. Retrieving Articulated 3-D Models Using Medial Surfaces and Their Graph Spectra. 285–300. https://doi.org/10.1007/11585978_19