



Dawid Machalica ¹, Marek Matyjewski ²

CAD models clustering with machine learning

Similarity assessment between 3D models is an important problem in many fields including medicine, biology and industry. As there is no direct method to compare 3D geometries, different model representations (shape signatures) are developed to enable shape description, indexing and clustering. Even though some of those descriptors proved to achieve high classification precision, their application is often limited. In this work, a different approach to similarity assessment of 3D CAD models was presented. Instead of focusing on one specific shape signature, 45 easy-to-extract shape signatures were considered simultaneously. The vector of those features constituted an input for 3 machine learning algorithms: the random forest classifier, the support vector classifier and the fully connected neural network. The usefulness of the proposed approach was evaluated with a dataset consisting of over 1600 CAD models belonging to 9 separate classes. Different values of hyperparameters, as well as neural network configurations, were considered. Retrieval accuracy exceeding 99% was achieved on the test dataset.

1. Introduction

1.1. 3D model retrieval

Over the last few decades, 3D models have been used in various applications including medicine, biology, chemistry, archaeology, games and industry [1, 2]. At the same time, an increasing number of 3D models stored in databases and product lifecycle management systems (PLM) enforces development of methods for 3D geometry comparison and retrieval. In the industrial applications, the opportunity to detect existing, similar components accelerates the design process of the new

✉ Dawid Machalica, e-mail: dawid.machalica1@ge.com

¹Warsaw Institute of Aviation, Warsaw, Poland.

²Warsaw University of Technology, Institute of Aeronautics and Applied Mechanics, Warsaw, Poland. E-mail: mmatyjew@meil.pw.edu.pl



© 2019. The Author(s). This is an open-access article distributed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives License (CC BY-NC-ND 4.0, <https://creativecommons.org/licenses/by-nc-nd/4.0/>), which permits use, distribution, and reproduction in any medium, provided that the Article is properly cited, the use is non-commercial, and no modifications or adaptations are made.

parts [3]. It also eliminates the risk of producing several identical components and thus higher unit cost. Finally, it reduces the number of manufacturing issues and the number of files that need to be maintained. All of those features result in a lower price of the component created based on the existing design.

The significance of the problem of the 3D geometry comparison and retrieval made it an area of an extensive study since 1990 [4]. However, the initial, text-based algorithms, utilizing the name of the file, its extension and assigned tags turned out to be insufficient. They had poor efficiency and unsatisfactory retrieval accuracy [5]. Simultaneously, it has been proven that better results can be obtained by incorporating content-based searches in which 3D model or its representation is used to formulate the query [6].

As far as we know, there is no method for a direct comparison of 3D models. Therefore, researchers have developed a series of descriptors called shape signatures, representing 3D model and its attributes. Available shape signatures will be discussed in section 2 of the article. Nonetheless, it is important to underline that most of the works focus on a development of a single “perfect” descriptor which will allow one to compare designs and precisely define their similarity factor. In reality, usage of just one shape signature usually results in the part descriptor that is suitable only in some specific area of application.

In this paper, an alternative method based on utilizing a combination of different shape signatures was proposed. The vector of easy-to-extract attributes of the geometry including total area, volume and number of edges was used as an input for supervised machine learning algorithms supposed to learn how to cluster components based on the aforementioned properties.

The paper is organized as follows: in section 2 existing methods of 3D shape clustering is discussed. The overview of the proposed technique together with shape signatures and machine learning algorithms to be used are discussed in section 3. Section 4 presents the implementation of the proposed method, evaluation of hyperparameters and discussion on obtained results. Finally, section 5 covers the discussion, whereas conclusions and future work are presented in section 6.

1.2. Machine learning

Machine learning (ML) is a field of computer science utilizing statistics to make computers to learn without being explicitly programmed [7]. Machine learning techniques can be divided into 2 main categories – supervised learning and unsupervised learning algorithms. Supervised learning algorithms, including, i.e., regression, need to be feed with both input data and desired outputs. Based on those 2 datasets, the algorithm learns how to obtain the desired output based on the corresponding inputs. On the other hand, in unsupervised learning the desired outputs are not presented to the model and the algorithm is supposed to find the optimal method for categorizing data without any guidance. Both supervised and unsupervised machine learning techniques proved to be useful in many real-world

applications. They can be used for classification, regression and dimensionality reduction type problems. Thanks to their capability to generalize patterns, they are not sensitive to noise (small variations) in the input vector. Additionally, as the operation of many ML models, such as neural network, is based on simple calculations, after being trained they operate faster than methods utilizing filtering and set of logical rules [8].

Machine learning algorithms, after being trained, are supposed to correctly predict desired output based on the previously unseen input vector. Because of that, accuracy of ML algorithms should be determined based on previously unseen samples. In practical applications, the dataset is usually divided into 3 groups – training, validation and test datasets. The parameters of the ML algorithm, called the hyperparameters, are tuned using validation dataset, the model is trained with the training dataset whereas the final model performance (e.g., accuracy) is estimated based on the test data. This approach gives more accurate predictions on how the model will behave in the future. As the extraction of the validation dataset leads to the reduction in the amount of data which can be used for training, the alternative approach, called the cross-validation, is commonly used. Cross-validation is a technique in which a dataset is divided into n portions. Then, all combinations of $n-1$ portions are used for training purposes when the last fold is used to test the performance of the model. This approach allows determining not only the average performance of the classifier using specific hyperparameters, but also standard deviation describing how much the performance of the model depends on the data used during the training.

Detailed description of the various machine learning algorithms and their applications can be found in [9]. In this paper, 3 different supervised learning algorithms were compared. They include the random forest classifier, the support vector classifier (SVC) and the fully connected neural network. The first one – *random forest classifier* is actually a set of simpler predictive models called decision trees. Each decision tree is a tree-like graph using conditional instructions on input variables to obtain the desired output. As random forest utilizes a set of random trees which simultaneously “vote” for the final output, this method is less sensitive to the selection of samples used for training and thus less vulnerable to overfitting – obtaining perfect predictions for training dataset and low performance on the test set.

Another technique called *the support vector machine* (SVM), or its variety called *the support vector classifier* (SVC), creates a set of hyperplanes in the high dimensional space which is supposed to split the dataset into desired clusters while maintaining the maximum distance between hyperplanes and adjacent samples. Those hyperplanes can then be used for regression (in SVM application called the Support Vector Regression or SVR) or classification purposes. The SVC algorithm is considered to achieve high classification accuracy on datasets composed of a relatively small number of samples compared to the number of features describing the samples.

The last machine learning technique considered in this work was *the fully connected neural network* (NN). NN is made up of neurons grouped into layers and links connecting neurons belonging to adjacent layers. Every neural network contains input and output layers, separated by one or more “hidden” layers. NN including more than 1 hidden layer is sometimes called the “deep neural network”. Each value of an input vector provided to the input layer is multiplied by weights assigned to links connecting neurons. The resultant values are then summed and processed by an activation function in the neurons of the next layer. Following this scheme, some values will be obtained on the output layer. Those values can be treated as the output of the neural network. Training of a NN is thus optimizing the weights assigned to specific links, so they produce desired values on the output layer.

2. Related works

As far as we know, 3D models cannot be compared directly. Instead, body descriptors called shape representations or shape signatures are extracted from the geometry. Then, the similarity between components is obtained by comparing their shape descriptors. Set of requirements for the perfect shape representation was discussed in [10]. The most important ones include the effectiveness of extracting shape descriptor from the model and the number of values it is composed of. Nevertheless, the aforementioned requirements are only the guideline for the researchers looking for the new shape signatures, and to the best of our knowledge, none of the existing shape signatures fully meets all of the requirements.

3D shape clustering techniques are often classified based on the features utilized to describe geometry and its topology. Those general groups of algorithms rely on global and manufacturing features, graphs, histograms and 2D views of the model. Detailed comparison of the algorithms can be found in [2, 3, 10].

In the area of descriptors utilizing global features, those relying on geometric properties are usually the easiest to extract. In [11], search engine filtering 3D models based on their geometric properties was proposed. The utilized features included volume, total surface area as well as geometric ratios – crinkliness and compactness. In [12], the parameters used in filtering were expanded with those related to the hull covering the model. It reduced the variation in the shape descriptors due to minor features existing in the models. This work also suggested to incorporate additional features including bounding-box aspect ratio, number of faces and holes. Finally, in [13] similarity factor based on the average ratio of the properties such as the number of edges, faces, hollows etc., was utilized. What is worth mentioning is the fact that the algorithm was very robust and successfully worked with industrial parts. Other types of global feature-based algorithms utilize moments [14, 15] and spherical harmonics [16]. Obtaining those descriptors might be more complex or computationally expensive but their usage is not limited to solid modeling and thus can be utilized to compare 3D scans or surface meshes. Because of that, their applications extend far beyond industrial applications.

Another popular group of descriptors uses graphs. The method of transforming the 3D model into an attributed graph was presented in [17]. In this approach, the faces of a CAD model are represented by the nodes of a graph, whereas the edges of a model refer to its links. Attributes such as the face or edge type, supposed to facilitate graph comparison, are assigned to specific features of the resultant attributes graph. The process of indexing, retrieval and matching graphs representing CAD models was discussed in [18]. However, because of the insufficient computational resources (2003), inexact graph comparison method was recommended in this work. In [19], a more advanced approach to the comparison of attributed graphs was presented. It utilized simulated annealing optimization to obtain maximum common subgraph. The method proved the effectiveness of detecting both global and local similarities between CAD models. However, it was also computationally expensive and thus some of the queries took more than 4 hours. It is also worth mentioning that the application of the graph-based methods in 3D CAD model retrieval is limited by the existence of minor features like blends or chamfers. It is because these kinds of features can significantly affect the size and structure of the resultant graph.

In recent years, image analysis is increasingly used to compare 3D modes. First work in this area was probably presented in [20]. The authors utilized the fully connected neural network to classify shafts, based on their cross sections represented by 19×13 pixels images. In [21], “elevation descriptor” obtained based on 6 basic grey-scaled views (top, bottom, front etc.) was defined and utilized to 3D model retrieval. In [22], algorithm for selection of most representative views presenting 3D geometry was presented. It increased retrieval accuracy simultaneously reducing computational cost. Finally, in [23], convolutional neural network was utilized to detect similarities between sketches and 2D views of 3D geometry. This approach gives an opportunity to find an existing model based on freehand sketch rather than 3D geometry that might be unknown or unavailable.

3. Methodology overview

In the Fig. 1 general overview of the applied methodology was presented.



Fig. 1. Methodology overview

In the first step, 1653 CAD models were downloaded from the Product Lifecycle Management system (PLM), courtesy of General Electric. Those models were created with the use of Siemens NX CAD software and saved in the .prt file format. Those models belong to 9 different categories and are a part of a welded assembly commonly used in gas turbines. Some examples of the geometries, together with the number of components belonging to specific classes, were presented in Fig. 2.

All the models were created with regard to the same axis of rotation, however, their axial and circumferential position might vary. It is also worth mentioning that some classes (1–3, 4–6 and 7–9) are visually similar to each other. Classes 1–3 differs mainly because of the orientation in 3D space. Classes 4–6 differs because of the thickness and existence of small holes. Finally, classes 7–9 have different diameters and wall thicknesses.

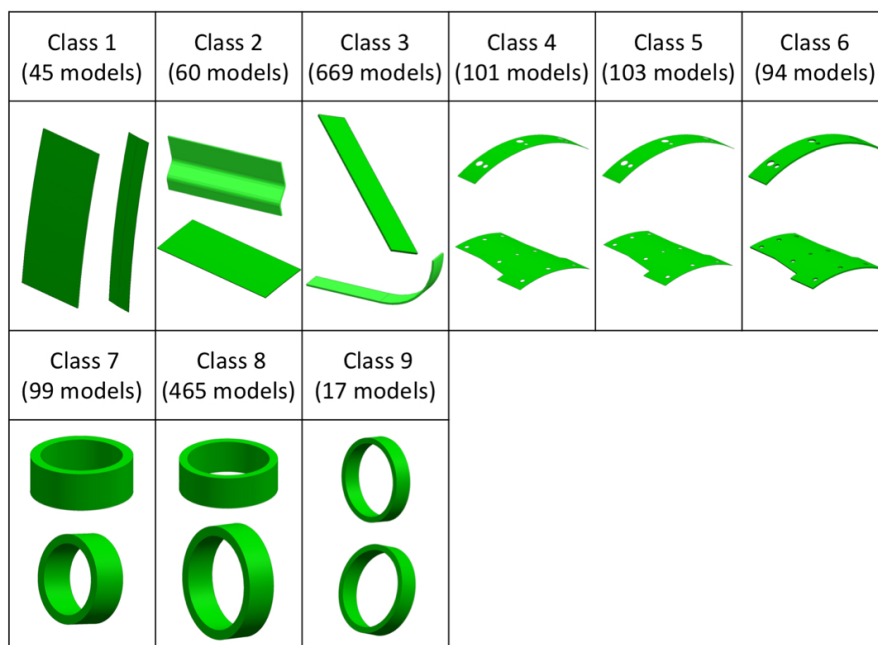


Fig. 2. Examples of CAD models belonging to specific classes

In the second step, the collected models were processed with an application developed in C# programming language, utilizing NXOpen API. With the use of the aforementioned software, 45 variables characterizing each CAD model were extracted. They are: number of faces, number of edges, total surface area, volume, volume/area ratio, area/volume ratio, compactness (volume squared divided by the cube of the total surface area), crinkliness (ratio of the total surface area of a model and surface of the sphere having the same volume), area of the largest face, relative area of the largest face (with regard to the total area of the model), type of the largest face (e.g., planar, cylindrical), sizes of the bounding box, moments of inertia (around 3 axes that constitute absolute coordinate system), number of faces of each type (e.g., planar, cylindrical), number of edges of each type (e.g., line, arc), percentage of faces having relative size within predefined ranges (0–10%, 10–20%, 20–30%, 30–40%, 40–50%). The collected variables characterizing CAD models were then saved into the .csv file. The program was able to extract all variables

for every single model. Thanks to that, there were no missing values in the dataset what could constitute a problem in the case of some machine learning algorithms.

Raw data to be used by machine learning algorithms need to be preprocessed. The numerical data, e.g., volume, surface area was normalized. It was to avoid algorithms being biased by attributes with higher absolute values. Additionally, categorical data (LargestFaceType) were converted into “dummy variables” (IsPlanar, IsCylindrical, IsConical). As a result, the number of features increased to 47. They are presented in Table 1. All of the data processing was executed with Python libraries: Pandas and NumPy.

Table 1.

Features used by machine learning algorithms

Feature		Data Type
Number of faces		Numerical
Number of edges		Numerical
Total surface area		Numerical
Volume		Numerical
Volume / Total surface area		Numerical
Total surface area / Volume		Numerical
Compactness		Numerical
Crinkliness		Numerical
Area of the largest face		Numerical
Relative area of the largest face		Numerical
Size of the bounding box	<i>x</i> direction	Numerical
	<i>y</i> direction	Numerical
	<i>z</i> direction	Numerical
	diagonal	Numerical
Moment of inertia	<i>x</i> axis	Numerical
	<i>y</i> axis	Numerical
	<i>z</i> axis	Numerical
Number of faces of each type (e.g. planar, conical)	(11 types)	Numerical
Number of edges of each type (e.g. line, arc)	(11 types)	Numerical
Percentage of faces having relative size within predefined ranges	0–10%	Numerical
	10–20%	Numerical
	20–30%	Numerical
	30–40%	Numerical
	40–50%	Numerical
Type of the largest face	IsPlanar	Categorical; Converted to 3 dummy variables
	IsCylindrical	
	IsConical	

In the last step, 3 different machine learning algorithms: the random forest classifier, the support vector classifier and the fully connected neural network; were fed with the preprocessed data. As the accuracy of those algorithms strongly depends on the values of the hyperparameters, their values were optimized with the use of grid search and 5-folds cross-validation.

Finally, the machine learning algorithms were trained with the optimal values of hyperparameters and train-test split defined as 0,25. As a result, overall classification accuracy, F1 score, the precision-recall curve and confusion matrix were generated for each of the considered algorithms. To implement machine learning algorithms, the scikit-learn and the Keras (running on a top of the TensorFlow) libraries were used.

4. Results

To objectively evaluate results obtained with machine learning algorithms, the reference “benchmark” classification accuracy ($CA_{\text{benchmark}}$) needs to be calculated. It can be defined as:

$$CA_{\text{benchmark}} = \frac{\text{Number of CAD models in the most common cluster}}{\text{Total number of CAD models in the dataset}}. \quad (1)$$

For the considered dataset, the benchmark is equal to 40.5% (669/1653). It means that machine learning algorithms need to obtain higher accuracy to prove they perform better than “betting” that specific sample belongs to the most common model type.

4.1. Random Forest

The random forest classifier was implemented with the use of the scikit-learn library. The number of trees voting for the final classification was defined as 100 and was a compromise between the ability of the resultant model to generalize and time required to train the model. Additionally, according to the suggestions on the specific algorithm implementation [24], the minimum size of the “leaf” was determined as 5. It is to force the algorithm to generalize data and, as a consequence, to avoid overfitting.

The other important hyperparameter of a random forest classifier is the *max depth of a tree*, defining the maximum number of steps in a division process of a dataset. Too small value of this hyperparameter results in low accuracy of the model, as only a few features are subsequently used to divide the dataset. On the other hand, the too large number will lead to overfitting – the model will learn how to perfectly split the models from the training data set when classification accuracy for the test dataset will be considerably lower. The mean classification accuracy obtained for different *max depth* hyperparameter values were presented in Fig. 3. Based on this chart, we can state that max depth equal to 8 is the

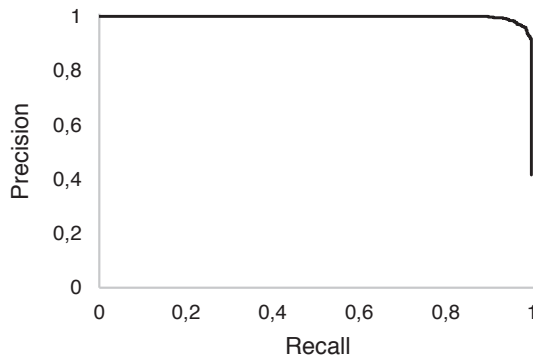


Fig. 4. Classification accuracy as a function of “max depth” of the forest

The obtained values prove that even relatively simple machine learning model can provide classification accuracy much higher than the benchmark. According to the precision-recall curve (Fig. 4), the model achieves almost ideal classification precision (99% or more) for the recall less than 0.9. In other words, if the goal is just to detect some examples of the models belonging to a specific class, the random forest classifier performs very well. On the other hand, if the goal is to detect all instances of the specific class (recall equal to 1), the number of incorrectly labeled models increases and thus the classification precision drops significantly.

Features most likely used in the random forest classifier were presented in Fig. 5. 10 features presented in the chart have over 63% of impact on the final accuracy of the model. According to those statistics, features representing relationship between area and volume are the most critical ones. Nevertheless, the total number of faces and edges included in the model is also important.

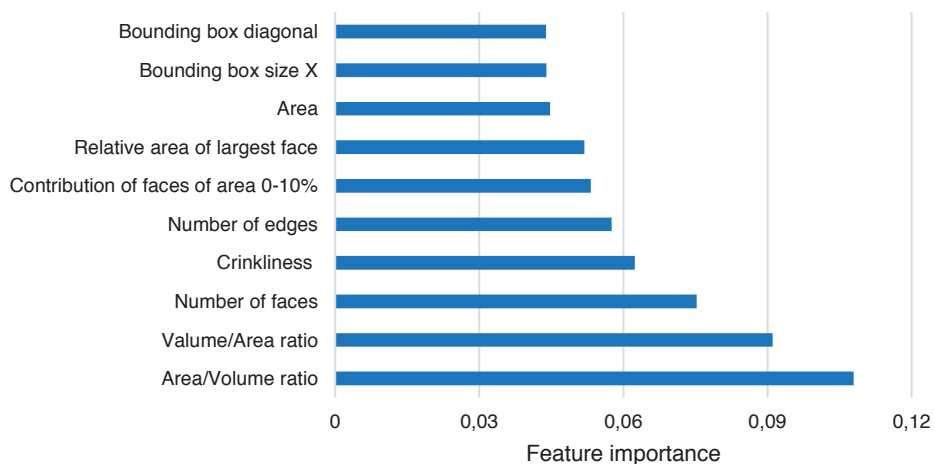


Fig. 5. The most important features according to the random forest classifier

4.2. Support Vector Classifier (SVC)

Support vector classifier (SVC) is an algorithm that proved to be useful in the case of problems that consist of a large number of features inside input vector compared to the number of samples used for training. However, the best accuracy of the algorithm is achieved with datasets whose classes do not overlap and can be easily separated with the hyperplanes. It can potentially be a problem in the case of the considered dataset.

SVC is an algorithm working with numerical data. Because of this requirement, categorical features (*LargestFaceType*) were converted into the set of dummy variables (*IsCylindrical*, etc.) Additionally, as SVC is not scale invariant, all numerical features were normalized to avoid the algorithm being biased by the absolute value of the specific feature.

Support vector classifier is also vulnerable to the values of hyperparameters. Because of that, four most important hyperparameters were optimized with the use of grid search [9]. Those hyperparameters included: the kernel defining the type of hyperplane used to split the data; γ defining how well algorithm should match the training data and thus defining the compromise between generalization and overfitting; C defining soft margin around hyperplanes within which points are not assigned to any of the classes. Additionally, for the polynomial kernel, different values of degree were tested. Optimal values of hyperparameters obtained in the optimization using 5-folds cross-validation were presented in Table 3.

Table 3.

Optimal hyperparameters values obtained for SVC

Kernel	γ	C	Degree
Polynomial	1	10	3

After optimal values of hyperparameters were evaluated, another SVC model was built to obtain the precision-recall curve and the confusion matrix. Those characteristics are presented in Fig. 6 and Table 4, respectively. According to the precision-recall curve, SVC performs worse than the random forest classifier in the

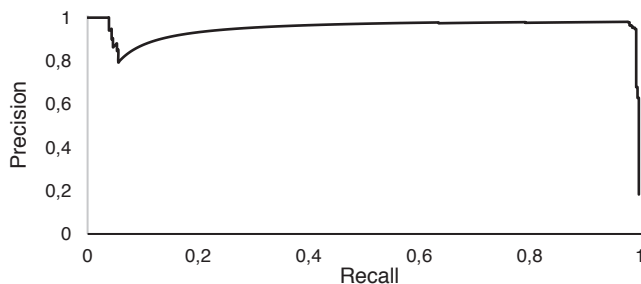


Fig. 6. Precision-recall curve obtained for the Support Vector Classifier

significant range of recall values. Nevertheless, SVC outperforms random forest classifier in the most important, top-right corner of the chart where both precision and recall are high. The fact that SVC performs better than the random forest classifier in this critical region was also captured in the confusion matrix (Table 3). It points out that SVC incorrectly classified only 9 out of 414 models achieving classification accuracy equal to 97.8%.

Table 4.

Confusion matrix obtained for support vector classifier

Predicted class	Actual class								
	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
Class 1	11	1	0	0	0	0	0	0	0
Class 2	0	14	0	0	0	0	0	0	0
Class 3	0	0	173	0	0	0	0	0	0
Class 4	0	0	0	15	6	0	0	0	0
Class 5	0	0	1	0	25	0	0	0	0
Class 6	0	0	0	1	0	17	0	0	0
Class 7	0	0	0	0	0	0	22	0	0
Class 8	0	0	0	0	0	0	0	123	0
Class 9	0	0	0	0	0	0	0	0	5

Unfortunately, because of the number of hyperparameters that need to be tuned, the creation of skillful SVC model is more time-consuming than the preparation of the random forest classifier model. Additionally, implementation of SVC available in the scikit-learn library might use *one-against-one* approach to multiclass classification [25]. It means the number of classifiers built in the background is proportional to the square of the number of classes. Because of that, utilization of SVC could be much more computationally expensive in the case of the real-world databases including many groups of models. In those cases, usage of *one-vs.-rest* approach could be more desirable.

4.3. Fully Connected Neural Network

There are two important hyperparameters controlling topology and architecture of the neural network. These are the number of hidden layers and the number of neurons in each layer. Neural networks including more than one hidden layer are commonly called deep neural networks [26]. It is said that deep neural networks usually outperform regular neural networks [27] and are better in recognizing complex patterns. In this work, both a “shallow” and a deep network (with 2 hidden layers) were considered.

Among the algorithms considered in this work, the neural network has the largest number of parameters to be tuned. However, in this paper, the optimization was limited to the network architecture. The remaining hyperparameters were adopted from the literature [28, 29] and are presented in Table 5. It was assumed that their suboptimal values will be partially compensated by the relatively high number of 5000 training epochs. Additionally, the initial tests indicated that the best classification accuracy was achieved with the use of the rectified linear unit (ReLU) activation function assigned to neurons in all hidden layers. The sigmoid function was used in the output layer to ensure that the output values are in a 0–1 range what can be regarded as a probability of the CAD model belonging to the specific cluster.

Table 5.

Hyperparameters used for training of the neural networks

Activation function (hidden layer(s))	ReLU
Activation function (output layer)	Sigmoid
Loss function	categorical cross entropy
Optimizer	Adam with default parameters
Number of epochs	5000
Batch size	32

As the considered dataset is relatively small, the exhaustive approach for neural network structure optimization was applied. In the exhaustive approach, all potential configurations of the neural network within predefined bounds are considered. Generally, the increase in the number of neurons in the hidden layer should increase the capability of the neural network to perceive patterns. On the other hand, too large number of neurons can result in overfitting and increase learning time. In this paper, the following heuristics were used to determine the maximum and minimum number of neurons in the hidden layers to be used in the optimization:

- The number of neurons in the hidden layer should be smaller than the size of the input layer and larger than the size of the output layer. In the considered case, it translates to the maximum of 46 neurons.
- The number of neurons in the hidden layer should not be larger than $2/3$ of the sum of neurons in the input and the output layers. In the considered case, it is the maximum of 37 neurons.
- The total number of neurons in all hidden layers should not be greater than twice the size of the input layer.

The influence of the number of neurons in the hidden layer on the classification accuracy (calculated with use of 5-fold cross validation) was presented in Fig. 7. It is observed that, starting from 8 neurons, the classification accuracy fluctuates

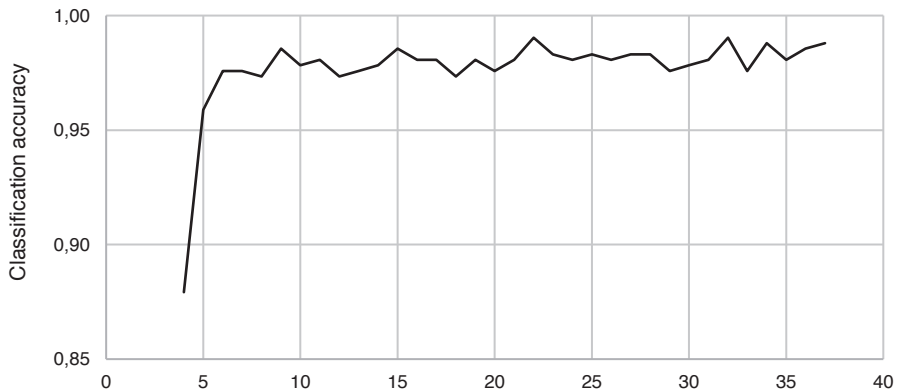


Fig. 7. Classification accuracy of a neural network as a function of the number of neurons in a hidden layer

around 98%. Nevertheless, the highest value was recorded for 22 hidden neurons and this value was used in further evaluation.

In the case of deep neural network, the third of the aforementioned heuristics imposes additional constraint on the total number of neurons. Optimization of the NN architecture indicated that the highest classification accuracy was obtained for the network with 20 neurons in the first hidden layer and 14 neurons in a second hidden layer (configuration: 47–20–14–9).

To obtain precision-recall curves and confusion matrices, additional models using optimal NN architecture were built. The model including 1 hidden layer achieved 98% classification accuracy, whereas the model with 2 hidden layers achieved accuracy as high as 99%. The obtained characteristics, i.e., precision-recall curves and confusion matrices are presented in Fig. 8 as well as Tables 6 and 7. Those characteristics prove that neural network can outperform other machine

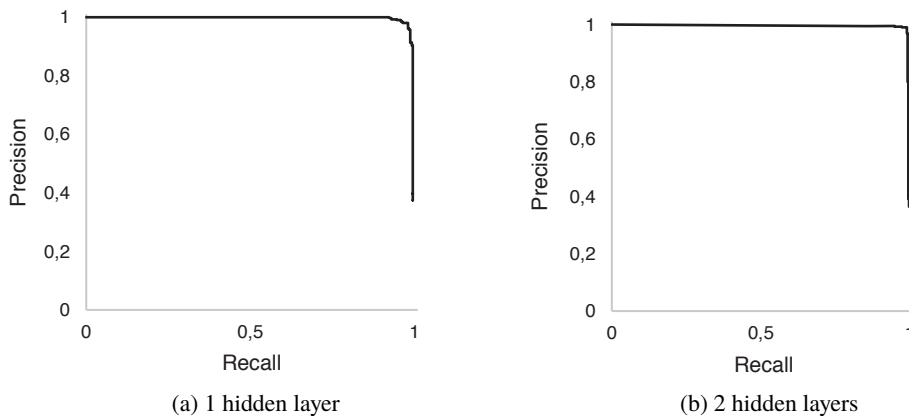


Fig. 8. Precision-recall curves obtained with the neural network

4.4. Comparison of machine learning models

All of the models considered in this paper achieved classification accuracy much higher than the benchmark. The comparison of the precision-recall curves presented in Fig. 9 shows that random forest classifier outperforms other models for the recall range 0–0.9. It means that this simple model can properly classify even 90% of the CAD models with the precision exceeding 99.5%. Nevertheless, the difference between random forest classifier and the neural network is very small (less than 0.5%). At the same time, the neural network performs much better in the top-right corner of the chart where both recall and precision are high. Actually, it is the most desirable point of operation for the machine learning model.

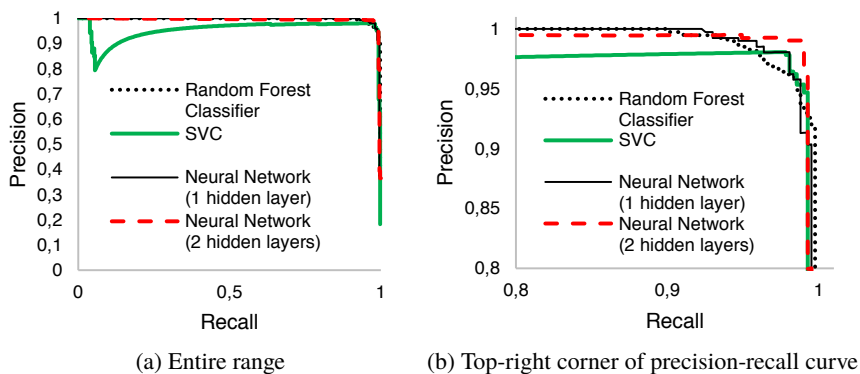


Fig. 9. Precision-recall curves obtained for different machine learning models

In Fig. 10 the F1 score, being the harmonic mean of the precision and recall was presented. It can be noticed that all the models achieve very high value of F1 for classes 3 and 6–9. However, for classes 1, 4 and 5, the random forest

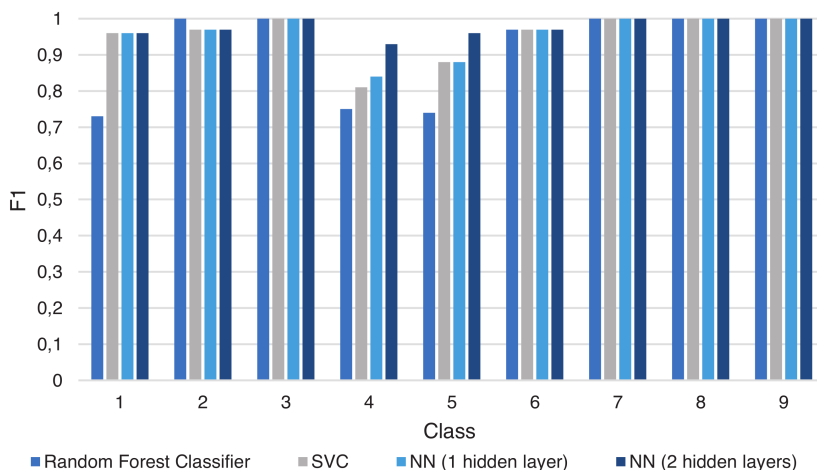


Fig. 10. F1 score obtained for different machine learning models

classifier performs worse than other models. Additionally, the difference in F1 values obtained for classes 4 and 5 proves that the neural network with 2 hidden layers is the best model to detect nuances distinguishing the part shapes.

5. Discussion

Most industries still use text-based search engines to retrieve CAD models. In the background, they utilize file name and manually-entered attributes describing the geometry. It makes those methods vulnerable to human error and difficult to maintain.

The machine learning approach using geometry attributes for classification provides a convenient alternative. By using different training datasets, the machine learning model can easily meet the requirements of the specific industry. It can also be expanded with new classification “rules” without the necessity to manually define them – the algorithm just needs to be retrained with the expanded/modified dataset. As a result, the algorithm is easy to maintain. Finally, the resultant (trained) model is relatively small compared to the methods based on *if-then* statements. It makes it possible to use the model even on computers with limited computational power.

The advantages of the proposed methodology provide many potential industrial applications. First of all, it could be implemented in the Product Lifecycle Management systems (PLM) to help designers find similar components already in the production. As a result, they could utilize the knowledge (e.g., about the acceptable tolerances) already stored in the parts being produced. It would also prevent designing very similar components or duplicates. The proposed approach could also enable Computer-Aided Process Planning (CAPP) systems to faster define manufacturing processes or estimate the manufacturing cost based on the most similar components which already have been considered.

6. Conclusions and future work

This paper introduces a methodology for utilizing machine learning (ML) algorithms for clustering CAD models. 45 properties of CAD model geometry (volume, surface area, type of the largest face etc.) were used as an input vector for 3 different machine learning models: the random forest classifier, the support vector classifier and the fully connected neural network. Each of those algorithms achieved average classification accuracy exceeding 95% when being trained and tested on the dataset generated with use of 1653 CAD models coming from the industry. The best classification accuracy, exceeding 99% was achieved with the deep neural network supposed to split the dataset into 9 different clusters. Confusion matrices prepared for each of the ML models showed that the biggest challenge for the algorithms was to properly classify CAD models that differ only because of

the existence of some minor features (e.g., holes). The initial tests proved that even simple machine learning techniques (random forest classifier), using only a few hyperparameters, can achieve relatively high classification accuracy. Additionally, due to the flexibility of machine learning techniques, the resultant models can be easily adjusted to specific requirements of the company or industry.

Further work should verify the reliability of the proposed method on a dataset that consists of more classes as well as includes some complex geometries. Additionally, only the features that mostly contribute to the final performance of the algorithms should be extracted. The input vector provided to the algorithm can also be expanded with new properties such as average length of the edge. Finally, further tuning of architecture and hyperparameters of the neural network can be considered as a potential method for obtaining even better results.

Acknowledgements

The authors want to express gratitude to the General Electric for the possibility of presenting results published in this paper.

Manuscript received by Editorial Board, October 11, 2018;
final version, January 29, 2019.

References

- [1] T. Funkhouser, P. Min, M. Kazhdan, J. Chen, A. Halderman, D. Dobkin, and D. Jacobs. A search engine for 3D models. *ACM Transactions on Graphics (TOG)*, 22(1):83–105, 2003. doi: [10.1145/588272.588279](https://doi.org/10.1145/588272.588279).
- [2] Y. Yang, H. Lin, and Y. Zhang. Content-based 3-D model retrieval: A survey. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 37(6), 1081–1098, 2007. doi: [10.1109/TSMCC.2007.905756](https://doi.org/10.1109/TSMCC.2007.905756).
- [3] N. Iyer, S. Jayanti, K. Lou, Y. Kalyanaraman, and K. Ramani. Three-dimensional shape searching: State-of-the-art review and future trends. *Computer-Aided Design*, 37(5):509–530, 2005. doi: [10.1016/j.cad.2004.07.002](https://doi.org/10.1016/j.cad.2004.07.002).
- [4] Z. Zhang, Z. Jiang, and X. Wang. Biased support vector machine active learning for 3D model retrieval. In: *2010 International Conference on Mechanic Automation and Control Engineering*, pages 89–92, Wuhan, China, 26–28 June, 2010. doi: [10.1109/MACE.2010.5535431](https://doi.org/10.1109/MACE.2010.5535431).
- [5] H. Cheng, C. Chu, E. Wang, and Y. Kim. 3D part similarity comparison based on levels of detail in negative feature decomposition using artificial neural network. *Computer-Aided Design & Applications*, 4(5):619–628, 2007. doi: [10.1080/16864360.2007.10738496](https://doi.org/10.1080/16864360.2007.10738496).
- [6] B. Bustos, D.A. Keim, D. Saupe, T. Schreck, and D.V. Vranić. Feature-based similarity search in 3D object databases. *ACM Computing Surveys*, 37(4):345–387, 2005. doi: [10.1145/1118890.1118893](https://doi.org/10.1145/1118890.1118893).
- [7] J.R. Koza, F.H. Bennett, D. Andre, and M.A. Keane. Automated design of both the topology and sizing of analog electrical circuits using genetic programming. In: J.S. Gero, F. Sudweeks, editors, *Artificial Intelligence in Design '96*, pages 151–170, Springer, Dordrecht, 1996. doi: [10.1007/978-94-009-0279-4](https://doi.org/10.1007/978-94-009-0279-4).

- [8] V.B. Sunil and S.S. Pande. Automatic recognition of machining features using artificial neural networks. *The International Journal of Advanced Manufacturing Technology*, 41(9–10):932–947, 2009. doi: [10.1007/s00170-008-1536-z](https://doi.org/10.1007/s00170-008-1536-z).
- [9] A.C. Müller and S. Guido. *Introduction to Machine Learning with Python: A Guide For Data Scientists*. O'Reilly Media Inc., 2016.
- [10] Z. Qin, J. Jia, and J. Qin. Content based 3D model retrieval: A survey. In: *2008 International Workshop on Content-Based Multimedia Indexing*, pages 249–256, London, UK, 18–20 June, 2008. doi: [10.1109/CBMI.2008.4564954](https://doi.org/10.1109/CBMI.2008.4564954).
- [11] H.J. Rea, J.R. Corney, D.E.R. Clark, J. Pritchard, M.L. Breaks, and R.A. MacLeod. Part-sourcing in a global market. *Concurrent Engineering*, 10(4):325–333, 2002. doi: [10.1177/a032004](https://doi.org/10.1177/a032004).
- [12] J. Corney, H. Rea, D. Clark, J. Pritchard, M. Breaks and R. MacLeod. Coarse filters for shape matching. *IEEE Computer Graphics and Applications*, 22(3):65–74, 2002. doi: [10.1109/MCG.2002.999789](https://doi.org/10.1109/MCG.2002.999789).
- [13] P. Cicconi, R. Raffaeli, and M. Germani. An approach to support model based definition by PMI annotations. *Computer-Aided Design and Applications*, 14(4):526–534, 2016. doi: [10.1080/16864360.2016.1257194](https://doi.org/10.1080/16864360.2016.1257194).
- [14] G. Cybenko, A. Bhasin, and K.D. Cohen. Pattern recognition of 3D CAD objects: towards an electronic yellow pages of mechanical parts. *International Journal of Smart Engineering Systems Design*, 1(1):1–13, 1997.
- [15] Z. Li, X. Zhou, and W. Liu. A geometric reasoning approach to hierarchical representation for B-rep model retrieval. *Computer-Aided Design*, 62:190–202, 2015. doi: [10.1016/j.cad.2014.05.008](https://doi.org/10.1016/j.cad.2014.05.008).
- [16] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz. Rotation invariant spherical harmonic representation of 3D shape descriptors. In: *Proceedings of Eurographics Symposium on Geometry Processing*, pages 156–164, 2003.
- [17] M. El-Mehalawi and R.A. Miller. A database system of mechanical components based on geometric and topological similarity. Part I: representation. *Computer-Aided Design*, 35(1):83–94, 2003. doi: [10.1016/S0010-4485\(01\)00177-4](https://doi.org/10.1016/S0010-4485(01)00177-4).
- [18] M. El-Mehalawi and R.A. Miller. A database system of mechanical components based on geometric and topological similarity. Part II: indexing, retrieval, matching, and similarity assessment. *Computer-Aided Design*, 35(1):95–105, 2003. doi: [10.1016/S0010-4485\(01\)00178-6](https://doi.org/10.1016/S0010-4485(01)00178-6).
- [19] C.F. You and Y.L. Tsai. 3D solid model retrieval for engineering reuse based on local feature correspondence. *The International Journal of Advanced Manufacturing Technology*, 46(5–8):649–661, 2010. doi: [10.1007/s00170-009-2113-9](https://doi.org/10.1007/s00170-009-2113-9).
- [20] H. Kaparthi and N.C. Suresh. A neural network system for shape-based classification and coding of rotational parts. *International Journal of Production Research*, 29(9):1771–1784, 1991. doi: [10.1080/00207549108948048](https://doi.org/10.1080/00207549108948048).
- [21] J. Shih, C. Lee, and J.T. Wang. A new 3D model retrieval approach based on the elevation descriptor. *Pattern Recognition*, 40(1):283–295, 2007. doi: [10.1016/j.patcog.2006.04.034](https://doi.org/10.1016/j.patcog.2006.04.034).
- [22] Y. Gao, M. Wang, Z.J. Zha, Q. Tian, Q. Dai, and N. Zhang. Less is more: efficient 3-D object retrieval with query view selection. *IEEE Transactions on Multimedia*, 13(5):1007–1018, 2011. doi: [10.1109/TMM.2011.2160619](https://doi.org/10.1109/TMM.2011.2160619).
- [23] Z. Zhu, C. Rao, S. Bai, and L.J. Latecki. Training convolutional neural network from multi-domain contour images for 3D shape retrieval. *Pattern Recognition Letters*, 119:41–48, 2019. doi: [10.1016/j.patrec.2017.08.028](https://doi.org/10.1016/j.patrec.2017.08.028).
- [24] Scikit-learn, documentation.
- [25] S. Knerr, L. Personnaz, and G. Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In: F.F. Soulie and Jeanny Hérault, editors, *Neuro-computing: Algorithms, Architectures and Applications*, pages 41–50, Springer-Verlag, 1990.

-
- [26] Y. Bengio. Learning Deep Architectures for AI. *Foundations and Trends® in Machine Learning*, 2(1):1–127, 2009. doi: [10.1561/2200000006](https://doi.org/10.1561/2200000006).
- [27] J. Patterson and A. Gibson. *Deep Learning. A Practitioner's Approach*. O'Reilly Media Inc., 2017.
- [28] M.A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.
- [29] D.P. Kingma and J.Ba. Adam: a method for stochastic optimization. In: *Proceedings of 3rd International Conference for Learning Representations*, San Diego, 7–9 May, 2015.