Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 109 (2022) 275–280

www.elsevier.com/locate/procedia

32 nd CIRP Design Conference

AI based geometric similarity search supporting component reuse in engineering design

Carmen Krahe[a,*], Milan Marinov[b], Theresa Schmutz[a], Yannik Hermann[a], Mike Bonny[b], Marvin May[a], Gisela Lanza[a]

[a]*Karlsruhe Institute of Technology (KIT), wbk Institute of Production Science, Kaiserstrasse 12, 76131 Karlsruhe, Germany*
[b]*USU Software AG, Rüppurrerstrasse 1, 76137 Karlsruhe, Germany*

* Corresponding author. Tel.: +49-721-608-44011 ; fax: +49-721-608-45005. *E-mail address:* carmen.krahe@kit.edu

Abstract

Today, companies are faced with the challenge to develop and produce individualized products in the shortest possible time at very low cost in order to remain attractive under strong competitive pressure. For reasons of efficiency, products are therefore often developed in generations. Proven components are adopted in a new product generation and only some of the components are newly developed to meet new customer requirements. Many companies, therefore, have a large database of 3D CAD product models containing years of engineering experience. Nevertheless, it is often difficult to execute database queries to find which products or components already exist and could be reused or adapted for a new product generation or variant. As a result, many duplicates are created, which are associated with high effort and costs, and the risk of introducing design errors increases.
Therefore, the aim of this paper is to develop an automated approach for geometric similarity search that also takes company-specific features of components into account. Machine learning methods are capable of automatically extracting relevant geometric features by learning a suitable representation of the corresponding 3D object. For this purpose, an autoencoder is developed which is trained to extract class-specific feature vectors. To improve the representativeness of those vectors for the similarity search, the architecture and hyperparameters of the autoencoder are optimized based on several experiments. Considering a real use case with a data set from the field of mechanical engineering, it is shown that geometrically similar CAD models can be found very quickly using the learned representation, and that better results are obtained than with conventional methods based on meta information, e.g. volume and bounding box. On the one hand, the fast finding of similar models encourages the reuse of existing solutions. On the other hand, standardization and, thus, economy of scale is promoted.

Keywords: Artificial Intelligence; Pattern Recognition; Design; Similarity Search; Product Development

1. Introduction

Nowadays, due to increasing individualization and globalization as well as shortening of product lifecycles [1], companies are faced with the challenge of bringing innovative products to market at the lowest possible price in an even shorter time. In order to meet these requirements, product development in particular faces the great challenge of creating innovative products under enormous cost and time pressure. The majority of companies develop their products in generations [2]. According to [2], to a large extent proven solutions (e.g. components) are taken over into new generations and are slightly adapted if necessary. Only a small part of a new product is actually newly developed. The advantage of this is that proven solutions and, thus, experience can be built upon. However, in many companies, due to the lack of transparency, this wealth of experience is not yet used systematically [3]. Instead of reusing existing product components, new variants are developed, leading to unnecessary variant creation and, hence, higher costs.

2212-8271
10.1016/j.procir.2022.05.249

A variety of commercial software for similarity search on 3D models exists, but they are usually based on hand-crafted attributes, e.g. [4,5] use so-called geometrical footprints.

In the state of the art, also a variety of approaches are based on predefined features. For example, in [6,7] several geometric features (e.g. volume, surface) are used to describe the geometry of the CAD models and, thus, to determine the similarity. The approach of [8] considers the CAD model tree to find geometrically similar models or model components via similarities of subtrees. The similarity is determined based on the CAD design features used. An overview is given e.g. in [9].

Machine learning methods, on the other hand, offer the possibility of automatically extracting suitable and representative features from the data [10]. In recent years, the increasing availability of 3D models in public databases has promoted the further development of existing 2D Deep Learning methods to 3D. In the 2D domain, Deep Learning methods are already widely used in practice [11]. In the 3D domain, however, special challenges have to be met, in particular a suitable form of a 3D object representation [11]. Common approaches are multi-view [12], voxel [13], point clouds [14,15], graphs [16] or meshes [17]. A comprehensive overview of current 3D data representations used in deep learning is given in [11]. Based on these representations, supervised learning tasks such as classification or segmentation are performed. However, there are also some approaches, such as [18–20], in which unsupervised methods are first used to learn suitable representations based on point clouds. For this purpose, autoencoder (AE) architectures are used and the learned representations are tested on tasks like classification. In general, AEs are capable of learning a compressed representation of a dataset by encoding the relevant features in a self-supervised way. In [19] a point cloud AE based on a graph kernel operation is used. The graph kernel functions, in analogy to convolution kernels in CNNs, extract semantically meaningful local features from the local neighborhood of a point cloud. This improves the quality of the latent vectors and at the same time, the robustness of the model. In [20] a combination of an AE and Capsule networks is presented, which first learns an alignment of the objects in a canonical frame allowing the processing of non-aligned objects. Based on the representations learned via AE, a clustering of 3D objects is performed in [21] or [22]. However, those promising deep learning approaches have largely been performed only on simple datasets, such as ShapeNet [23]. These datasets contain simple objects that are normalized and aligned to the unit cube.

Recently, there has been a growing interest in using deep learning methods, especially in the field of CAD [24]. In CAD, however, it is often dealt with data on component-level, some of which may be very similar within a component class, but some of which may be very different. Often, companies are also unaware of corresponding class labels, but further information, such as maximum dimensions or material, can be extracted from the CAD models. Existing methods do not yet take such information into account.

Therefore, the goal is to adapt methods of deep learning for 3D objects for the application in the CAD domain and to use them in the form of a geometric similarity search. To this end, a design assistance system according to [25] is being developed that is intended to show designers similar components at part level that already exist in the early stages of product development in order to make better use of already existing knowledge and, thus, to avoid unnecessary creation of variants.

The specific contributions of this paper are:

- Development of an AE architecture based on state of the art point cloud AEs specifically for use in the product development process.
- Approaches to improve the representativeness of learned coding specifically for product development applications.
- Development of a geometric similarity search methodology based on the learned representations.
- Evaluation of the approach based on a real data set from the engineering domain to find similar CAD models.

2. Methodology

The aim of this approach is to find the geometrically most similar existing CAD models from a database for a given CAD model. The corresponding features, which are used to determine the similarity between the geometries, are learned automatically from the corresponding data set using deep learning methods. In contrast to conventional methods based on predefined attributes, company-specific component features can be learned. On this basis components can be distinguished. A point cloud AE based on [19] is used to extract these features. For this, the CAD models firstly are transformed into point clouds. Based on the point clouds, the latent representations are learned which finally represent the geometric properties of the 3D model. Consequently, in the learned latent space, latent vectors whose corresponding 3D models are geometrically similar are supposed to be close to each other. This property is used for similarity search. For a given model, the latent representation of an input CAD model can first be created using the trained encoder. For this latent vector, the most similar CAD models are then found by searching for the closest vectors in the latent space. For example, the Euclidean distance can be used as a distance measure for this. The approach is represented in Fig. 1. The core prerequisite for the approach is a sufficient representativeness of the learned latent space. For this purpose, the basic architecture of the AE, which is based on [19] was first tested on a sample data set. Based on these results, iterative optimizations were made, on the one hand by preprocessing the input data, and on the other hand by adapting the AE architecture itself.

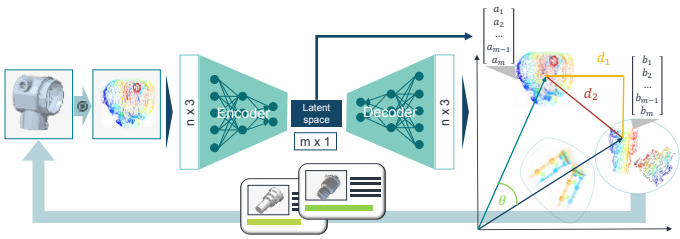

Fig. 1. Overview of the AE based similarity search approach.

2.1. Data set

The basis for the experiments is an industrial database containing CAD models of real-world, practice-relevant components. In total, the data set comprises 1000 objects of

five different classes: 524 adapters, 137 covers, 94 flanges, 102 housings and 143 O-Rings. For Processing, the objects are converted to point clouds, centralized and normalized. For normalization, each coordinate is scaled by an individual scaling factor in the form of the maximum dimension of its bounding box. 95% of the data, both per class and consequently in total, is used for training, with the remaining 5% constituting the test set. The split is chosen because in a real use case a large amount of data is already available and only a small amount of new components will be added.

2.2. Architecture of the autoencoder

In contrast to 2D images or 3D voxels, the individual points of a point cloud, consisting of 3 coordinates, are not sorted in any specific order. One of the consequences is that convolutions such as those used by CNNs, can not be applied to point clouds. The AE introduced by [19] for 3D point cloud processing consists of a graph based encoder and a folding based decoder, as shown in Fig. 2. The encoder architecture implements a graph kernel based approach, which captures local features of point clouds in a manner analogous to CNNs capturing local semantic features of images. The decoder uses two consecutive multilayer perceptrons to bend a fixed 2D grid into the original 3D point cloud shape based on the information stored in the feature vector. The encoder component of the AE learns to produce a latent representation of the original point cloud *X*. The input of the encoder is a list of points, respectively an unordered list of 3D-coordinates. The output of the encoder is a latent vector of size *M* which is an *Mx1*-dimensional list of real numbers. The size of the vector corresponds to the size of the AE's bottleneck, the so-called bottleneck size (BNS). The learned latent vector represents the most significant features of the point cloud, which are necessary for the decoder to learn a reconstruction *X'* of the original input point cloud *X*.

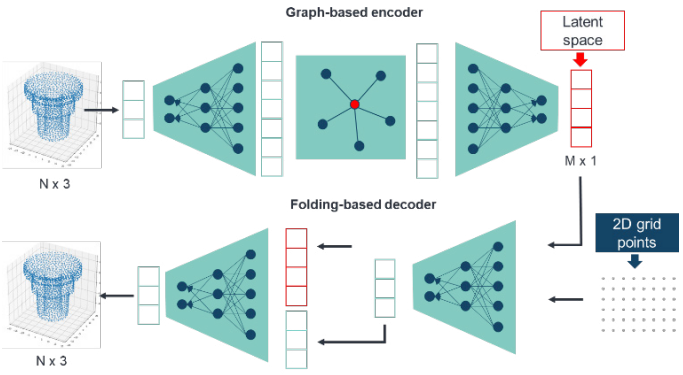

Fig. 2. Architecture of the AE based on [19].

2.3. Results with basic autoencoder

In the first experiment, the basic AE was trained with a BNS of 512 and a learning rate of 0,0005. Subsequently, used to generate the latent vectors. To investigate the representativeness of those vectors, a k-Means clustering is performed that leads to a Homogeneity Score (HS) [26] of 35.74 %. A maximum HS is achieved when there are only objects of one class in a cluster. Fig. 3 shows the visualization of the latent space of the training data set in two dimensions using tSNE, a method introduced by [27] to visualize high dimensional data. In general, the object vectors in the latent space do not form distinct clusters and are highly intermixed across the different classes. Only O-rings (purple), with the exception of a few outliers, arrange themselves in an elongated cluster. Considering the flanges (green) several small clusters can be identified. One cluster is concentrated at the top and the bottom respectively, several small clusters are located in the middle of the diagram. In general, round and angular flanges are each grouped in separate clusters. In addition, differences in the orientation of the components are noticeable for the individual clusters. The upper cluster contains flanges aligned vertically in space. The same applies to the lower cluster, however those objects are rotated by 90-°. The components of the middle clusters are oriented horizontally.

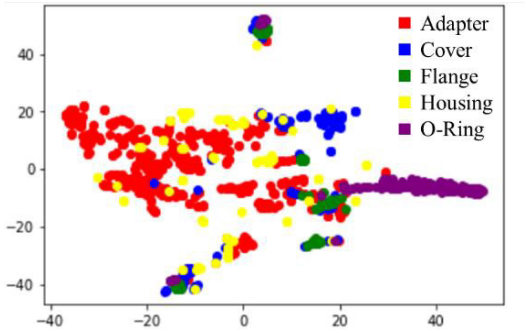

Fig. 3. tSNE visualization of the learned latent vectors for the basic AE.

2.4. Experiments for optimization

As shown in Section 2.3, different clusters have emerged due to inconsistent orientation of the objects. To eliminate such variation within the data set, in a first experiment the components were aligned in a uniform standard orientation. To better discriminate between different classes, additional class labels are considered in another experiment for a part of the dataset. A general drawback is that size information relevant to the engineering domain is lost due to the necessary normalization of the objects. Therefore, these are transferred into the latent space in a last experiment with a two-stage AE architecture.

2.4.1. Pre-alignment of input data

In the first optimization step the input point clouds are brought into a standard orientation according to the standard orientation algorithm introduced in [25]. With these standard aligned point clouds, the AE is trained with the same parameters as in Section 2.3. K-Means Clustering resulted in a HS of 46 %, which is an improvement of about 10 % compared to the basic approach. The tSNE visualization is shown in Fig. 4. It is quickly noticeable that the clusters are better separated from each other compared to the clusters of the basic approach, in particular for the O-Ring class. Only one adapter and one housing are located in this cluster. However, a closer look at these objects reveals that they are obviously outliers that are very close to the geometric shape of an O-ring. Compared to Section 2.3, the flanges now form two unique clusters consisting of angular or round flanges respectively. Furthermore, the thickness of the flanges is responsible for an

additional division for the round flanges. Considering the adapters, two larger clusters can now be identified with shorter ones in the lower area and much longer ones in the upper area. The geometries within the cover and housing classes are highly inhomogeneous, which makes clustering fundamentally difficult. Nevertheless, for covers, a grouping with similar diameter and wall thickness can be found.

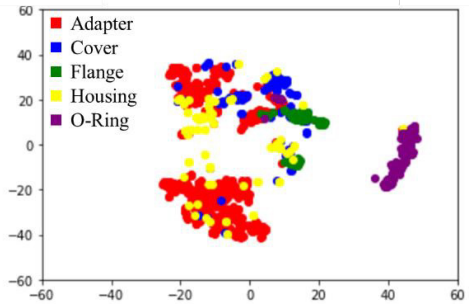

Fig. 4. tSNE visualization of the learned latent vectors for the AE with standard oriented input.

2.4.2. Transfer learning with partially labeled data

To better distinguish between latent vectors of different classes despite similar geometries, the labels are additionally used for training from a small part of the dataset. Therefore, the AE architecture is extended by a classifier, which can be optionally activated (see Fig. 5). In this way, the AE can be trained both with and without labels. The classifier has a multilayer perceptron (MLP) architecture with dropout and batch normalization layers, as well as a softmax activation function applied on the last layer. It is attached to the final layer of the encoder in parallel with the decoder.

In the first training phase on the labeled data, the loss functions of the decoder (Chamfer distance) and the classifier (cross entropy loss) are combined via weighted sum and used for updating the encoder network via back propagation. The goal is that the encoder now also takes into account the class information in the latent vectors in addition to the pure geometry of the point clouds. In the second training phase for the unlabeled part of the data set, the classifier is deactivated again and only reconstruction loss is used for network updates. The aim is to transfer the initially learned representation improved for class-specific clustering of the smaller, labeled dataset, to the training of the unlabeled dataset. For the experiment, the original training data set is subdivided: First, 10 % of the training data is passed to the AE with labels. The second phase of the training without labels is carried out on the remaining 90 %. This division is chosen because in a real use case usually only a few data with labels exist. Given this input data, the AE is iteratively trained with a BNS of 512, a learning rate of 0,001 with labels and of 0,0001 without labels. Clustering on the latent vectors using k-Means leads to a HS of 53,72 %. Fig. 6 shows the resulting tSNE visualization of the latent space. Compared to Section 2.4.1, a relative improvement in HS of about 16 % can be achieved. Nevertheless, housings as well as partly covers remain highly dispersed and intermingle with other clusters. This observation can be attributed to the wide range and high variation of housings and covers in terms of their geometry. However, a closer look at the covers can reveal geometric peculiarities of the latent space. For example, the covers at the bottom left of the tSNE-diagram, i.e. those that intersect the flange cluster, are flat, round covers that strongly resemble a flange.

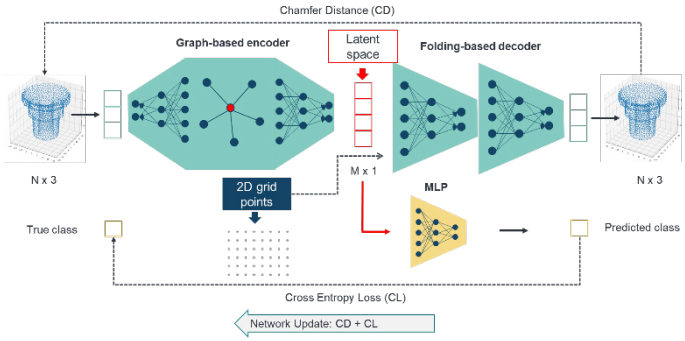

Fig. 5. AE with classifier.

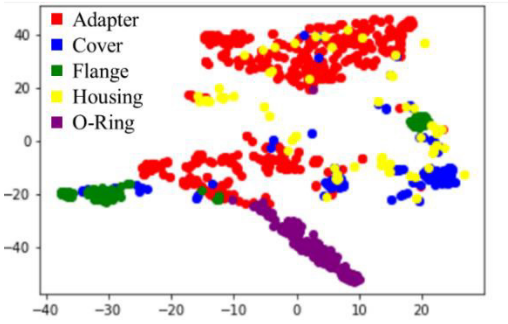

Fig. 6. tSNE visualization of the learned latent vectors for the AE with additional classifier.

2.4.3. Two-stage autoencoder with scaling factor input

For the second extension of the AE architecture, the object size, represented by the object's scaling factor (see Section 2.1), is used as an additional input for influencing the latent space. In contrast to Section 2.4.2, a second AE is implemented, which is independent of the point cloud auto-encoder (see Fig. 7). In the following, this AE is referred to as second order AE. Input for the second order AE are the learned latent vectors from the point cloud AE, in the following called first order AE. In this way, the learning of geometry (point cloud reconstruction) is clearly separated from the learning of other features, such as scaling factors.

The architecture of the second order AE is based on MLP layers, with one encoder and two decoders. Each of these components consists of only three MLP layers, so that the model architecture is simpler than the one of the first order AE. Correspondingly, the resource and time consumption of the second order AE is lower.

Input for the encoder are the first order latent vectors, which are extended by one dimension for the normalized scaling factor. First order latent vectors and corresponding scaling factors are transformed together into a second order latent space. Those second order latent vectors serve as input for both decoders. The first decoder learns to reconstruct the first order latent vector, while the second decoder learns to reconstruct the scaling factor. Mean Squared Error (MSE) is used as a loss function for both decoders. Consequently, the loss function for the AE is composed of the weighted sum of both decoder losses. The appropriate choice of the weighting factors is a

hyperparameter of training. For the experiment, the second order AE is trained with a BNS of 256 and a learning rate of 0,0005. The weighting factors are set to κ_1= 0,1 and κ_2= 0,9 based on a grid search. Input for the encoder are the first order latent vectors according to Section 2.4.1 .Clustering on the latent vectors using k-Means leads to a HS of 53 %. Fig. 8 shows the resulting tSNE visualization of the latent space. Compared to Section 2.4.1, a slightly better HS is achieved, but the clusters are not further separated. However, when looking more closely at the scaling factors, it can be seen that the mean values are very similar across the different classes with the exception of the O-Ring class. To illustrate the effect of the scaling factor, the scaling factors of some of the O-Rings are increased by a factor of 100 in a further experiment. The AE is trained with the same hyperparameters as before. As shown in Fig. 9, O-Rings with different scaling factors are clearly separated, which demonstrates the general viability of the approach.

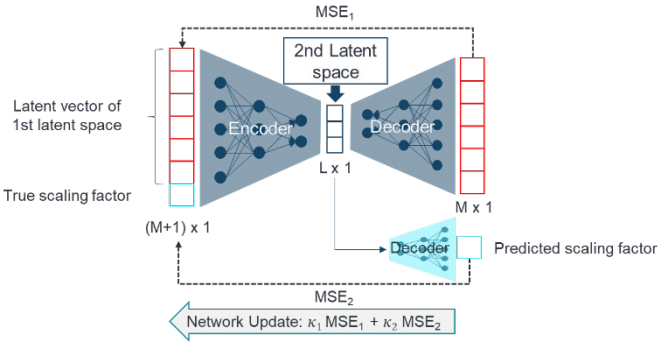

Fig. 7. Architecture of the second order AE.

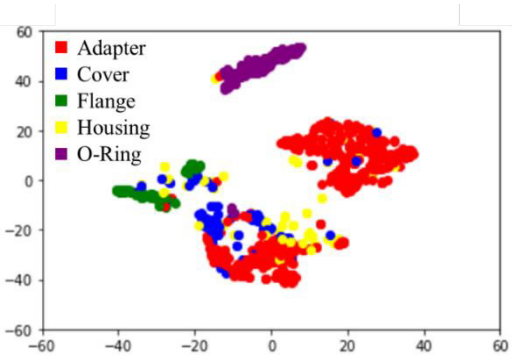

Fig. 8. tSNE visualization of the learned latent vectors for the two-stage AE with scaling factor.

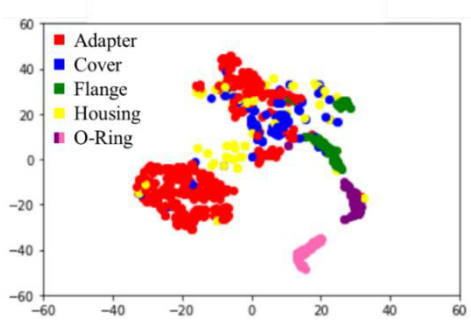

Fig. 9. tSNE visualization of the learned latent vectors for the two-stage AE with scaling factor. O-Rings with initial scaling factor are colored in purple and with manipulated scaling factors in pink.

3. Case Study

To validate the developed approach, the (geometric) similarity search is tested using the example data set of Section 2.1 from the field of mechanical engineering. For this purpose, the latent vectors of the AE with standard oriented input, with additional class labels as well as with scaling factors are used. To imitate a real industrial use case, all objects from the test data set are considered as new (unknown) CAD models, for which the most similar models from the existing database are to be identified. The existing database corresponds to the (known) training data. For a given input model from the (unseen) test data set, a latent representation is first created using the corresponding trained encoder. Finally, using the Euclidean distance, the most similar models are presented from the training dataset. A similarity search based purely on geometry descriptive metainformation (e.g. volume, bounding box) serves as a basis for comparison. For this purpose, 18 attributes in accordance with [7] are extracted from each model of the data set.

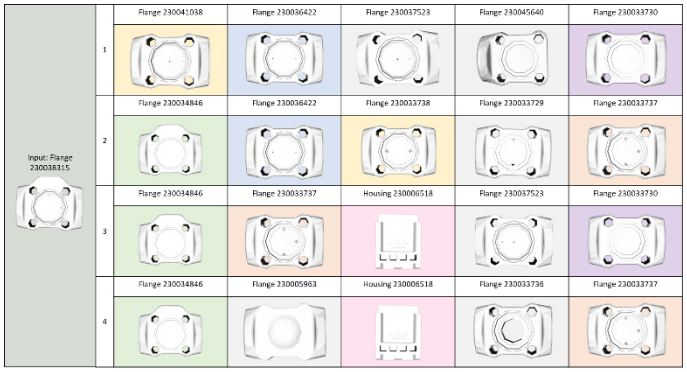

Fig. 10. Results of the similarity search based on line (1) metainformation, (2) latent vectors of the AE with standard oriented input, (3) latent vectors of the AE with Classifier and (4) latent vectors of the two-stage AE with scaling factors. Models found across different approaches are color coded.

Fig. 10 shows the results for an exemplary component. To determine the similarity, the calculated Euclidean distance is normalized with the maximum distance in the data set. Since the data are not based on labels as to how similar the components actually are, no quantitative statement is possible.

According to the results in Fig. 10, it is obvious that the similarity search via the attributes is not able to represent geometric details like the upper flap of the input model. The latent vector search based on the approaches of Section 2.3 and 2.4, on the other hand, returns flanges that also have such a flap. Moreover, the flange with the closest match is the same for all latent vector search approaches.

4. Discussion

As shown in the case study, with the developed approach data specific geometric details can be learned self-supervised. However, to show the effectiveness of the approach more clearly, testing on geometrically more complex data is necessary. In addition, only qualitative assessments of the similarities could be made in the case study. In various experiments, it was shown that the learned representations are significantly enhanced by a standard alignment. Using

additional class labels leads to a further improvement. For the application in practice, this means that at least a small part of the data would have to be labeled manually. The general functionality of the second order AE could be demonstrated using the manipulated scaling factors for O-Rings. Nevertheless, it should be investigated again on a more suitable data set with larger differences in the scaling factors. A general drawback of the approach is that no accurate representation can be learned for outliers. For practice this means that very rare components can be represented less well.

5. Conclusion and Outlook

In this paper, an approach for a deep learning based geometric similarity search is developed to enable the reuse of existing product models in the early phase of product development. For this purpose, an AE is used to extract relevant features of existing components in the form of latent representations. In several experiments it is shown how the learned representations can be further optimized for the similarity search. First, by aligning the data in a standard orientation, a significant improvement can be achieved. Second, the AE architecture is extended by an additional classifier which enables a better separation of different classes in the latent space. Third, the architecture is extended in a modular way, so that additional information like size or material can be encoded. Finally, a proof of concept is demonstrated in a case study with real industrial data from the field of engineering. In addition to its application in design, the approach can also be used in strategic procurement to achieve economies of scale or in manufacturing, e.g., to reuse routings of similar products. Further work should investigate how the approach works with semi-finished parts as input for the similarity search. In addition, it should be considered how the approach works when not only one, but several additional pieces of information are processed simultaneously, besides geometry. Furthermore, it should be investigated how the approach performs in recognizing certain components within an assembly and thus suggesting the complete assemblies in which the component is installed.

Acknowledgements

This paper was also funded by the German Federal Ministry of Education and Research (BMBF) project AIAx, Machine Learning-driven Engineering – CAx goes AIAx (01IS18048B).

References

[1] Fleischer, B., 2019. Methodisches Konstruieren in Ausbildung und Beruf. Springer Fachmedien Wiesbaden, Wiesbaden.
[2] Albers, A., Bursac, N., Wintergerst, E., 2015. Product generation development-importance and challenges from a design research perspective, in: New developments in mechanics and mechanical engineering: proceedings of the International Conference on Mechanical Engineering (ME 2015).
[3] Ehrlenspiel, K., Kiewert, A., Lindemann, U., Mörtl, M., 2014. Kostengünstig Entwickeln und Konstruieren. Springer Berlin Heidelberg, Berlin, Heidelberg.
[4] simus systems GmbH, 2021. Geometrical Similarity Search for 3D Models. https://www.simus-systems.com/en/applications/geometrical-similarity-search-3d-models/.
[5] PDXVISION, 2021. ModelSearch. https://www.pdsvision.com/solutions/all-solutions/modelsearch/.
[6] Rea, H.J., Corney, J.R., Clark, D.E.R., Pritchard, J., Breaks, M.L., Macleod, R.A., 2002. Part-sourcing in a Global Market. Concurrent Engineering 10 (4), 325–333.
[7] Machalica, D., Matyjewski, M., 2019. CAD models clustering with machine learning. Archiv of Mechanical Engineering (vol. 66), 133–152.
[8] Bai, J., Gao, S., Tang, W., Liu, Y., Guo, S., 2010. Design reuse oriented partial retrieval of CAD models. Computer-Aided Design 42 (12), 1069–1084.
[9] Lupinetti, K., Pernot, J.-P., Monti, M., Giannini, F., 2019. Content-based CAD assembly model retrieval: Survey and future challenges. Computer-Aided Design 113 (5), 62–81.
[10] Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep learning. MIT Press, Cambridge, Massachusetts, London, England, 785 pp.
[11] Gezawa, A.S., Zhang, Y., Wang, Q., Yunqi, L., 2020. A Review on Deep Learning Approaches for 3D Data Representations in Retrieval and Classifications. IEEE Access 8, 57566–57593.
[12] Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E., 2015. Multi-View Convolutional Neural Networks for 3D Shape Recognition, in: Proceedings of the IEEE International Conference on Computer Vision, pp. 945–953.
[13] Sedaghat, N., Zolfaghari, M., Amiri, E., Brox, T. Orientation-boosted Voxel Nets for 3D Object Recognition, in: British Machine Vision Conference (BMVC).
[14] Qi, C.R., Su, H., Mo, K., Guibas, L.J., 2017. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, in: Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 77-85.
[15] Qi, C.R., Yi, L., Su, H., Guibas, L.J., 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space, in: Advances in Neural Information Processing Systems, pp. 5100–5109.
[16] Simonovsky, M., Komodakis, N. Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs, in: Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, pp. 29–38.
[17] Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S., Cohen-Or, D., 2019. MeshCNN: A Network with an Edge. ACM Trans. Graph. 38 (4), 1–12.
[18] Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L., 2018. Learning Representations and Generative Models for 3D Point Clouds. 35th International Conference on Machine Learning (ICML).
[19] Yang, Y., Feng, C., Shen, Y., Tian, D., 2018 through 2018. FoldingNet: Point Cloud Auto-encoder via Deep Grid Deformation, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 206–215.
[20] Sun, W., Tagliasacchi, A., Deng, B., Sabour, S., Yazdani, S., Hinton, G., Yi, K.M., 2020. Canonical Capsules: Unsupervised Capsules in Canonical Pose.
[21] Aljalbout, E., Golkov, V., Siddiqui, Y., Strobel, M., Cremers, D., 2018. Clustering with Deep Learning: Taxonomy and New Methods.
[22] Hassani, K., Haley, M., 2019. Unsupervised Multi-Task Feature Learning on Point Clouds, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 8160–8171.
[23] Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F., 2015. ShapeNet: An Information-Rich 3D Model Repository.
[24] Yoo, S., Lee, S., Kim, S., Hwang, K.H., Park, J.H., Kang, N., 2021. Integrating Deep Learning into CAD/CAE System: Generative Design and Evaluation of 3D Conceptual Wheel. Structural and Multidisciplinary Optimization (Volume 64), 2725–2747.
[25] Krahe, C., Iberl, M., Jacob, A., Lanza, G., 2019. AI-based Computer Aided Engineering for automated product design - A first approach with a Multi-View based classification. Procedia CIRP 86, 104–109.
[26] Rosenberg, A., Hirschberg, J., 2007. V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure, in: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 410–420.
[27] van der Maaten, L., Hinton, G., 2008. Viualizing data using t-SNE. Journal of Machine Learning Research 9 (2605), 2579–2605.