From CAD Data to semantic Graphs: A Framework for Classification

Robin Taba, Anna Liza Schonlau, Joshua Falkenhain, Patrick Maas, Prof. Dr. Frank Köster
Deutsches Zentrum für Luft- und Raumfahrt (DLR)
Institute for AI Safety and Security, Sankt Augustin, Germany
{robin.taba, anna.schonlau, joshua.falkenhain, patrick.maas, frank.koester}@dlr.de

Abstract—**The growing complexity of digital assemblies presents significant challenges, particularly in understanding their characteristics and interdependencies. Traditional approaches, which focus largely on geometric properties, fall short in capturing the functional dependencies between components. This research introduces a novel approach using semantic enrichment of CAD components and Graph Neural Networks (GNNs) to classify mechanical parts and analyze their interconnections. By leveraging open-source gearbox designs which are based on native CAD data in addition to STEP data, the methodology showcases the ability of graph-based structures to account for both geometric and functional relationships, providing a more comprehensive digital understanding of assemblies.**

I. Introduction

The increasing complexity of digital assemblies poses a considerable challenge for engineers and designers, particularly with regard to analyzing the propagation of errors in assemblies. In order to support engineers and designers in their work with digital tools, a digital understanding of the components and their connection to each other is necessary [1]. A precise and efficient classification of the components is crucial in order to understand the interactions between them and identify potential weak points. It is particularly important not only to capture the geometric properties of a component but also to understand its semantic meaning and function.

A natural candidate for a data structure which can model objects and their interdependencies is a graph. Many objects within a system might appear similar in isolation but reveal distinct characteristics when their relationships with neighboring objects are considered. This interconnectedness reflects the way these objects function as part of a larger, complex system. Graphs enable us to capture and process these relationships, making them highly suitable for representing and analyzing CAD constructions. Through graph structures, the inherent relationships and dependencies within these complex assemblies can be effectively preserved and further utilized for enhanced analysis.

Therefore, we propose a novel approach to CAD part classification which is based on node classification of assembly graphs using Graph Neural Networks. We use open-source available gearbox designs to develop and test our methodologies. Gearbox assemblies are particularly suitable because many components have to fulfill technical functions (e.g. power transmission or sealing) but also have high classification requirements as some parts are similar in their structure and only differ in their properties (e.g. sealing ring & washer).

In our approach we operate on native CAD files as they encompass a larger variety of information than the Standard for Exchange of Product Data (STEP). Although native CAD data is more complex to handle, as it is not standardised across different CAD applications, it offers significantly more information about components and their properties.

Our main contributions are summarized as follows:

- We developed a software tool to extract information from native CAD data and enhance it by geometric information from STEP data.
- We introduced a novel framework for classification of mechanical parts, namely each part is embedded in an assembly graph and classified by nested GNNs.

II. Related work

First, we provide an overview of the broad field of CAD classification and 3D object retrieval, in particular we highlight methods which operate on assembly structures. We give a short introduction to deep learning methods, specifically Graph Neural Networks (GNNs), and finally turn to the field of native data extraction.

A. 3D Object Retrieval

A key factor for efficient product design is design and knowledge reuse. Most companies possess large data bases of CAD models, however to find the appropriate CAD model for a particular application, effective retrieval algorithms are required. Rather than keywords they should take as input a 3D object and retrieve (partially) similar 3D objects [2].

Any of the proposed algorithms operate in two steps: first, the CAD model is represented by a simpler data structure, usually a feature vector or attributed graph, then two objects are compared by assessing a similarity metric on their respective representations.

For one there exist *feature-extraction techniques* which were investigated by the CAD community starting from 1980 with the objective to automatically recognize the presence of features, i.e. some shape pattern with significance, in

CAD models. These techniques take as input the boundary representation[1] (B-Rep) of a CAD product. One famous graph-based approach is to represent the B-Rep with the help of an attributed adjacency graph (AAG) [4] where nodes represent faces which are connected by an arc if the faces are adjacent. The arc has the label 0 or resp. 1 denoting concavity or resp. convexity of the angle between adjacent faces. Features are then extracted as subgraphs given by some predefined template. For an extensive survey see [5].

On the other hand, there exist *shape-based techniques* which became popular around the 2000's triggered by new developments in computer vision and computer graphics. These techniques usually work on polygonal mesh models. For a comprehensive overview of shape-based methods we refer to [2].

B. CAD Assembly Model Retrieval

In recent years attention has shifted from part to assembly retrieval. Assemblies are much richer in their structure which gives need for more sophisticated algorithms which can extract both local and global features.

Lupinetti et al. [6] consider the context of a component to improve classification accuracy. Their approach is based on the Enriched Assembly Model (EAM) introduced in [7] which consists of four layers. The statistics layer captures basic attributes like the number of subassemblies and fasteners. Structure represents the hierarchical arrangement of the assembly while the interface layer details part relationships, including contacts and joints. Shape categorizes the form of the assembly and its parts. These layers are used according to specific retrieval objectives [8].
In [9] Chen et al. extend assembly descriptions by incorporating degrees of freedom and geometric connections but manual intervention is required for complex kinematic pairs. Park and Oh [10] propose an automatic method to identify revolute and prismatic joints though it is limited to simple geometries. Swain et al. [11] suggest an extended structure to integrate product and process information for joint types, but this method is highly algorithmically intensive. Kim et al. [12] use ontology-based methods to represent assembly joints which require additional manual input [8].

C. Deep Learning in CAD Classification

1) Previous Methods: There are various deep learning approaches for classifying parts in CAD models which differ in terms of the methods and data structures used [13]. Convolution-based approaches, in particular convolutional neural networks, are often used to process 3D data such as meshes or voxel representations [14], [15]. These networks use convolutional operations to extract local geometric features of the parts such as point coordinates, normals and curvature. Graph Neural Networks (GNNs), on the other hand, are designed to process irregular structures such as the B-Rep of CAD models [16]. The features extracted in GNNs often include geometric properties like face area, edge length, curvature and adjacency relations. These networks aggregate information from neighboring nodes allowing them to capture both local geometry and the global topology of the parts.

2) Graph Attention Networks: Graph Attention Networks (GATs) [17] extend Graph Convolutional Networks (GCNs) by introducing an attention mechanism that adaptively weighs neighboring nodes during graph convolutions. Unlike GCNs, which assign uniform weights, GATs compute attention scores to focus on the most relevant neighbors, improving the quality of learned representations. These scores are learned during training.

GATs are widely used for node classification and can also handle graph-level classification by employing global pooling methods [18]. Additionally, GATs can process edge features, enabling richer structural representations, and their local computations make them scalable and robust to varying graph structures [19].

D. Extraction of Information from Native CAD Data

Native CAD data is high level data which, in addition to the B-Rep, contains machining and processing information like tolerances, materials and product constraints.

Most classification methods are based on neutral CAD data, e.g. STEP and STL. To the best of our knowledge native CAD data has thus far only been employed in the classification of sketches and 2D-drawing files as in [20].

Native data extraction is more prominent in research dealing with the exchange of data between different CAD software. Here it is important that no change or loss of information occurs during translation. One approach similar to ours is to record macros during the design process with the objective that the design intent matches the interpretation results of the log by implemented translators for several CAD systems [21].

III. Our Approach

In the following we explain how we model the assembly graph and classify its nodes. As previously mentioned we solely focus on gear assemblies. Figure 1 provides an overview of all intermediate steps needed to pass from a native product model to an assembly graph with classified nodes.

A. Assembly Graph

The assembly graph we construct is node and edge attributed. Node attributes encode individual part information like geometry and part attributes while edge attributes declare the types of relations this edge represents. Edges here display the connection of two components A and B whose measured minimum distance is below a threshold value of $0.01mm$.

[1]A solid can be represented by segmenting its boundary into a finite collection of objects, called faces. This idea is iterated; faces are represented by a collection of edges and edges by their vertices. The B-Rep comprises these 'boundary' objects including their adjacency relations. See e.g. [3].

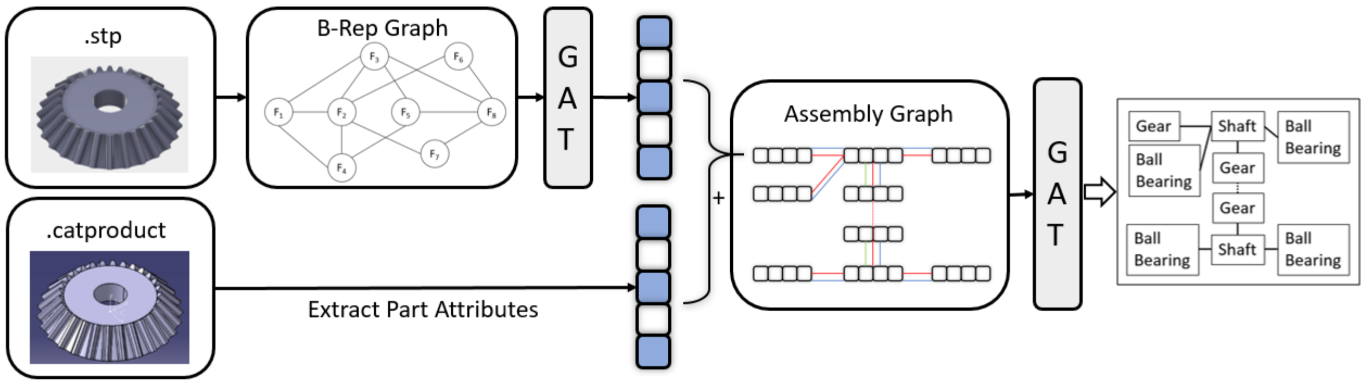

Fig. 1. The proposed Method. a) For each mechanical part we work with the .step and .catproduct file b) We extract the B-Rep from the .step file and represent it as a graph. With the help of a GAT the graph is transformed into a feature vector. c) Extract part attributes from .catproduct file. d) The part attributes and B-Rep are concatenated and serve as node attributes in the assembly graph. e) The nodes of the assembly graph are classified with another GAT which is connected to the first GAT. f) After classification we obtain a graph representation of the assembly depicting its (now known) components and relations among components

1) Edge Attributes: Edge attributes are represented by a 4-dimensional feature vector with binary feature coding. It contains four relations that originate from product constraints - *coincidence*, *offset*, *contact* and *angle* - which can take the value 1 for true and the value 0 for false. There exists one more constraint, *fix*, but since this references only one part this is added as a node attribute. Product constraints are meaningful because they encode the (indirect) degrees of freedom of each part.

Figure 2 illustrates the extraction of minimum distances and product constraints from a native CAD model. Note that product constraints have to be extracted from native product files, STEP files do not contain this type of information.

2) Node Attributes: The node attributes provide information about the individual parts. In general, there are three types of data provided for mechanical parts in a CAD model: product structure, geometrical information and part attributes (see [22]). We ignore the product structure in that we always access the lowest level of every structure i.e. a mechanical part which has no further subcomponents. For each part we extract a list of attributes from the native product model containing its weight, volume, area, center of gravity, principal moments, principal axes, material and bounding box. On the other hand, we use geometric data provided by the B-Rep which we extract from the STEP file. To be precise, we extract for each solid the faces and edges it is composed of and save them in a graph. Details will be given in the next section. To use the B-Rep graphs as node attributes they have to be transformed into vectors. This will be done by another 'inner' GNN which is combined with the outer GNN.

B. B-Rep Graph

To capture the geometry of mechanical parts we use a graph structure similar to that of the attributed adjacency graph (AAG). Namely, faces are represented by nodes and edges by arcs. Two nodes are connected by an arc if the corresponding faces share an edge. But in contrast to the AAG we define attributes on both edges and nodes.

1) Intrinsic Boundary Description: One should be aware that the B-Rep and thus the inferred graph is not unique and depends on the modeling process. To obtain a unique representation, called *Intrinsic Boundary Description*, one has to use the concept of maximal faces and edges as introduced in [23] and [24]. Maximal faces were first studied in [3] as so-called c-faces. We apply to each part the method *ShapeUpgrade_UnifySameDomain* provided by Opencascade to obtain maximal faces and edges. For larger amount of data this method is not suitable because it is not very robust.

2) Edge Attributes: The edge attributes contain three types of information: the normalized length of the curve, if the curve is closed or not and its curvature at N uniformly sampled points. Curvature is translation and rotation invariant but not scaling invariant. Hence, we scale each curve such that the maximal side length l_{max} of its bounding box is equal to 1.

3) Node Attributes: The node attributes are structured similarly. Let $(u,v) \mapsto S(u,v)$ be the geometric surface of face F. Again, we compute its normalized area and check if S is closed in the u or v direction. Next, we cover the u,v coordinate plane by a grid of $N \times N$ points and compute the *shape index* $s_p \in [-1,1]$ [25] for each point $p = (u_i, v_i)$.

The shape index is closely related to curvature, it is given by the formula:

$$s = \frac{2}{\pi} \arctan\left(\frac{\kappa_{min} + \kappa_{max}}{\kappa_{min} - \kappa_{max}}\right)$$

where κ_{min} and κ_{max} are the maximal and minimal curvature exhibited at a point p on the surface. Here we set $\frac{0}{0} = 0$ and $\arctan\frac{a}{0} = \lim_{\varepsilon\to 0}\arctan\frac{a}{\varepsilon} = \mathrm{sign}(a)\frac{\pi}{2}$. The shape index is obviously scaling invariant.

Finally, we define $G(p) = s_p$ if p also lies inside the domain of F - this might not be the case if some part has been cut out or trimmed - else we set $G(p) = 100$. Then, G is flattened and concatenated with the other node attributes.

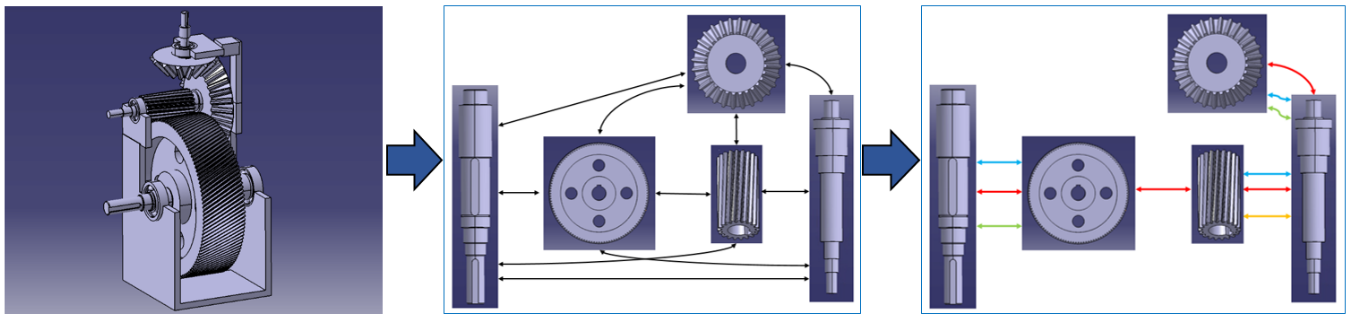

Fig. 2. Extraction of Edge Attributes. a) We work with the .catproduct file of the assembly. b) Compute all minimal distances and extract all product constraints. c) Connect parts where the minimal distance is below a certain threshold. Add edge attributes representing the product constraints. *red:* minimal distance $\leq 0.01mm$, *blue:* coincidence, *green:* contact, *orange:* offset.

C. Classification Methods

The classification process is divided into three approaches to better assess the framework we have developed. Each approach utilizes a single-layer Graph Attention Network following the principles of supervised learning.

First, node classification is performed on a simplified version of the assembly graph, which we call *CATIA graph*, using only information extracted from CATIA[2]. This graph has the same edge attributes as the assembly graph, but its nodes contain only part attributes and no B-Rep graphs. To enhance the network's performance and ensure comparability across different gearbox designs, node features are normalized using standard scaling. Since the constructions are fixed graphs it is not feasible to balance the dataset as this would require arbitrary removal or addition of nodes.

Second, graph classification is applied to the B-Rep graphs. In this approach, the entire graphs are classified using a single-layer Graph Attention Network along with global pooling. Additionally, the data set was not balanced in order to facilitate a better comparison with other methods.

Third, node classification is applied to the assembly graph, employing a nested Graph Neural Network architecture (see figure 1). The graph is organized into inner and outer components with both layers utilizing a single-layer Graph Attention Network. The outer graph is represented using information from CATIA where each node includes not only its parameters but also a trainable vector derived from the inner GAT applied to the B-Rep graphs.

IV. Data Extraction

We work with two different file formats: STEP and .catproduct. The main software application built and used in our approach works on native .catproduct files of gear assemblies, solely the B-Rep Graph is extracted from STEP files. In general STEP files can be converted to .catproduct files, however they would not contain product constraints. On the other hand, it is a complicated endeavour to extract B-Rep graphs from a native CAD model while STEP files can be handled and manipulated easily through the library pythonocc (7.7.0) [26] which is well adapted to this type of problem.

Since the automated extraction of data from .catproduct files is not very common, we describe the developed software tool in more detail.

A. Requirements concerning products and parts

We shortly list the requirements that assemblies need to fulfill on product and part level so that the software application works.

- Every part within the product has to have an individual part name.
- Methodically using constraints concerning part classes is important as there is no associated industry standard.
- Subproducts within the rootproduct need to be defined as 'flexible' so that one can reference the underlying parts when using constraints.
- Every part has to have a standard material assigned to it on the level of the partdocument.
- Parts must only consist of one main body.

B. Extraction of Data from the native CAD file

1) General Method: CATIA distributes an in-built interface that allows to execute scripts written in CatScript and VBScript (macros). This leads to a range of predefined commands stated in the CATIA Documentary which in our case can be used to retrieve data from the native CAD model of a product. Furthermore, we do semantic enrichment on part and product level with functions that access or create data which is not per default stored but could be retrieved by e.g. manually doing measurements. We automated these measurements and the resulting functionalities shall be explained further individually. To handle these cases and access the macro-controlled interface externally we created a software tool which uses the .Net Framework and the CATIA COM API.

2) Different functionalities based on generated output: In the following we outline which type of data is extracted and categorized based on the output.

a) General part parameters: These conclude the part index, center of gravity, the inertia matrix, material, mass, surface

[2]Catia V5 R29

& volume which can all be retrieved as numerical values explicitly from the part using commands from the CATIA users documentation. *b) Product constraints:* Each constraint is retrieved as a string that contains the name of the constraint and its references which are not mechanical parts but e.g. the specific surface that was used during construction. To find the referenced part we need to check for the name within that string which is why every part should have an individual name. *c) Minimal distances between parts:* This functionality is one of two which is not based on a given command but an automatic execution of the measurement tool in CATIA. Every component is iteratively paired with every other component which leads to $n(n-1)/2$ distances with n the number of mechanical parts within the product. *d) Bounding Boxes:* The bounding boxes can be used to retrieve the general dimensions of each component. This second automated process consists in loading a PowerCopy into the active document which measures the maximal length of the component in each direction.

V. Experiments

A. The Dataset

The training and testing of the classification task involves seven gearbox assemblies consisting of simple spur gear mechanisms with components from the following classes: *gear*, *inner ring*, *outer ring*, *roller*, *shaft* and a manually added *housing* as a fixed base part which every part is related to. Since these assemblies are adapted to real-world applications the distribution of components within the gearboxes is unbalanced, with a high frequency of rollers. The objective is to train the model on four gearbox assemblies and subsequently classify three completely unknown gearbox assemblies. The dataset includes parameters derived from CATIA as well as the B-Rep extracted from STEP. Table I provides an overview of the edges and nodes for both datasets. It can be observed that, on average, the boundary representations have more nodes and edges compared to the CATIA graphs and also offer more node and edge features.

TABLE I
Datasets

Name	Graphs	Average Nodes	Average Edges	Node Features	Edge Features
CATIA graph	7	49.71	162.29	31	4
B-Rep graph	348	57.80	399.28	103	12

B. Results

The classification was conducted in Python (3.11.5) using the PyTorch Geometric library (2.5.3) [18] which is specifically designed for Graph Neural Networks. The networks were trained on the basis of two gearbox assemblies over varying numbers of epochs with a gradually reduced learning rate. To evaluate the performance, classification accuracy, balanced classification accuracy, balanced accuracy, and the macro F1-score were used. Balanced accuracy and the macro F1-score are useful for imbalanced datasets where class distributions are uneven. Balanced accuracy calculates the accuracy for each class individually and then averages the results, ensuring that less frequent classes are equally considered [27]. The macro F1-score provides a balance between precision and recall across all classes, giving both common and rare classes fair weight in the evaluation [28]. These metrics help prevent overly optimistic results caused by dominant class distributions.

TABLE II
Classification Results

Name	Accuracy	balanced Accuracy	macro F1-Score
CATIA graph	0.829	0.672	0.644
B-Rep graph	0.899	0.792	0.718
Assembly graph	0.705	0.446	0.431

Table II summarizes the classification performance of the three evaluated approaches: CATIA, B-Rep and their combination; the assembly graph. The B-Rep method demonstrates the strongest results with an accuracy of 0.899 and a macro F1-score of 0.792. In comparison, the CATIA approach achieves an accuracy of 0.829 while the combined method shows even lower performance with an accuracy of 0.705 and a macro F1-score of 0.446. These findings indicate that the B-Rep technique is the most effective for this classification task.

VI. Conclusion

In conclusion, the results show that while the B-Rep method performs well, the other approaches struggle with the classification. This is mainly due to the strong class imbalance and the small size of the dataset, as using only seven gear assemblies is not enough. Because of that the model has difficulty identifying the subtle differences between classes that look similar. Many components in gear systems have similar shapes, especially ring-like or circular shaped forms, making it hard to distinguish between them without considering their context or neighboring parts.

To get better results it is important to expand the dataset, as having more examples helps the model to better differentiate between classes. Testing different deep learning methods can also help find the most effective classifiers. Additionally, using different sets of features or adding new ones, like degrees of freedom, could improve the accuracy of the classification.

The representation of assemblies as graphs not only facilitates classification but also opens up various applications, including rule-based simulation and error propagation analysis on the basis of class information, further underscoring the potential of this approach.

References

[1] S. O. Abioye, L. O. Oyedele, L. Akanbi, *et al.*, "Artificial intelligence in the construction industry: A review of present status, opportunities and future challenges," *Journal of Building Engineering*, vol. 44, p. 103 299, 2021.

[2] 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*, vol. 37, no. 5, pp. 509–530, Apr. 2005.

[3] C. Silva, "Alternative definitions of faces in boundary representatives of solid objects," University of Rochester. Production Automation Project, Technical Report, 1981.

[4] S. Joshi and T. Chang, "Graph-based heuristics for recognition of machined features from a 3d solid model," *Computer-Aided Design*, vol. 20, no. 2, pp. 58–66, Mar. 1988.

[5] Y. Shi, Y. Zhang, K. Xia, and R. Harik, "A critical review of feature recognition techniques," *Computer-Aided Design and Applications*, vol. 17, no. 5, pp. 861–899, Jan. 2020.

[6] K. Lupinetti, F. Giannini, M. Monti, M. Rucco, and J.-P. Pernot, "Identification of functional sets in mechanical assembly models," in *International Conference on Innovative Design and Manufacturing*, Milan, Italy, 2017, pp. 1–6.

[7] K. Lupinetti, F. Giannini, M. Monti, and J.-P. Pernot, "CAD assembly descriptors for knowledge capitalization and model retrieval," in *Tools and Methods for Competitive Engineering (Aix-en-Provence : 16 : 2016)*, vol. 1, Aix-en-Provence, France, 2016, pp. 587–598.

[8] K. Lupinetti, F. Giannini, M. Monti, and J.-P. Pernot, "Automatic extraction of assembly component relationships for assembly model retrieval," *Procedia CIRP*, vol. 50, pp. 472–477, 2016, 26th CIRP Design Conference.

[9] X. Chen, S. Gao, S. Guo, and J. Bai, "A flexible assembly retrieval approach for model reuse," *Computer-Aided Design*, vol. 44, no. 6, pp. 554–574, 2012.

[10] S. C. Park and J. W. Oh, "Kinetic model extraction from a geometric model," *Computer-Aided Design and Applications*, vol. 12, no. 3, pp. 338–343, 2015. eprint: https://doi.org/10.1080/16864360.2014.981464.

[11] A. K. Swain, D. Sen, and B. Gurumoorthy, "Extended liaison as an interface between product and process model in assembly," *Robotics and Computer-Integrated Manufacturing*, vol. 30, no. 5, pp. 527–545, 2014.

[12] K.-Y. Kim, H. Yang, and D.-W. Kim, "Mereotopological assembly joint information representation for collaborative product design," *Robotics and Computer-Integrated Manufacturing*, vol. 24, no. 6, pp. 744–754, 2008, FAIM 2007.

[13] N. Heidari and A. Iosifidis, *Geometric deep learning for computer-aided design: A survey*, 2024. arXiv: 2402.17695 [cs.CG].

[14] Z. Wu, S. Song, A. Khosla, *et al.*, *3d shapenets: A deep representation for volumetric shapes*, 2015. arXiv: 1406.5670 [cs.CV].

[15] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, *Pointnet: Deep learning on point sets for 3d classification and segmentation*, 2017. arXiv: 1612.00593 [cs.CV].

[16] L. Mandelli and S. Berretti, *Cad 3d model classification by graph neural networks: A new approach based on step format*, 2022. arXiv: 2210.16815 [cs.CV].

[17] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, *Graph attention networks*, 2018. arXiv: 1710.10903 [stat.ML].

[18] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[19] S. Xiao, S. Wang, Y. Dai, and W. Guo, "Graph neural networks in node classification: Survey and evaluation," *Machine Vision and Applications*, vol. 33, Jan. 2022.

[20] M. F. Jabal, M. S. Rahim, N. Z. Othman, and D. Daman, "Computer-aided design data extraction approach to identify product information," *Journal of Computer Science*, vol. 5, no. 9, p. 624, 2009.

[21] J. Li, S. Han, S. Shin, *et al.*, "Cad data exchange using the macro-parametrics approach: An error report," *International Journal of CAD/CAM*, vol. 10, no. 2, 2010.

[22] K. Lupinetti, J.-P. Pernot, M. Monti, and F. Giannini, "Content-based cad assembly model retrieval: Survey and future challenges," *Computer-Aided Design*, vol. 113, pp. 62–81, Aug. 2019.

[23] K. Li, G. Foucault, J.-C. Léon, and M. Trlin, "Fast global and partial reflective symmetry analyses using boundary surfaces of mechanical components," *Computer-Aided Design*, vol. 53, pp. 70–89, Aug. 2014.

[24] F. Boussuge, J.-C. Léon, S. Hahmann, and L. Fine, "Extraction of generative processes from b-rep shapes and application to idealization transformations," *Computer-Aided Design*, vol. 46, pp. 79–89, Jan. 2014.

[25] J. J. Koenderink and A. J. van Doorn, "Surface shape and curvature scales," *Image and Vision Computing*, vol. 10, no. 8, pp. 557–564, Oct. 1992.

[26] T. Paviot, *Pythonocc*, en, 2022.

[27] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, "The balanced accuracy and its posterior distribution," *2010 20th International Conference on Pattern Recognition*, pp. 3121–3124, 2010.

[28] J. Opitz and S. Burst, "Macro f1 and macro f1," *arXiv preprint arXiv:1911.03347*, 2019.