Robotics and Computer–Integrated Manufacturing 86 (2024) 102661

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcim

Full length article

AAGNet: A graph neural network towards multi-task machining feature recognition

Hongjin Wu, Ruoshan Lei, Yibing Peng *, Liang Gao

School of Mechanical Science & Engineering, Huazhong University of Science and Technology, Wuhan, PR China

ARTICLE INFO

Keywords:
Machining feature recognition
Attributed adjacency graph representation
Instance segmentation
Graph neural network

ABSTRACT

Machining feature recognition (MFR) is an essential step in computer-aided process planning (CAPP) that infers manufacturing semantics from the geometric entities in CAD models. Traditional rule-based MFR methods struggle to handle intersecting features due to the complexity of representing their variable topological structures. This motivates the development of deep-learning-based methods, which can learn from data and overcome the limitations of rule-based methods. However, some existing deep learning methods compromise geometric and topological information when using certain representations such as voxel or point cloud. To address these challenges, we propose a novel graph neural network, named AAGNet, for automatic feature recognition using a geometric Attributed Adjacency Graph (gAAG) representation that preserves topological, geometric, and extended attributes from neutral boundary representation (B-Rep) models. Furthermore, some existing methods (such as UV-Net, Hierarchical CADNet) lack the capability of machining feature instance segmentation, which is a sub-task of feature recognition that requires the network to identify different machining features and the B-Rep face entities that constitute them, and it is a crucial task for subsequent process planning. AAGNet is designed as a multi-task network that can perform semantic segmentation, instance segmentation, and bottom face segmentation simultaneously for recognizing machining features, the faces associated with those features, and their bottom faces. The AAGNet is evaluated on various open-source datasets, including MFCAD, MFCAD++, and the newly introduced MFInstSeg dataset with over 60,000 STEP files and machining feature instance labels. The experimental results demonstrate that AAGNet outperforms other state-of-the-art methods in terms of accuracy and complexity, showing its potential as a flexible solution for MFR in CAPP.

1. Introduction

Computer-aided process planning (CAPP) plays a pivotal role in modern manufacturing by facilitating the integration of computer-aided design (CAD) and computer-aided manufacturing (CAM) systems. CAPP is designed to streamline the manufacturing process by automating various aspects of production planning, such as tool selection, machining parameters, and sequencing. However, one of the primary challenges in CAPP is automatic feature recognition (AFR). AFR is a key step in the CAPP as it involves interpreting the CAD models and translating them into recognizable machining features that can be used by CAM. The accurate recognition is essential for generating effective machining plans and optimizing the manufacturing process. Therefore, AFR has become a significant area of research in the field of manufacturing automation.

AFR has been a topic of research for over four decades, with various proposed approaches aimed at identifying machining features in CAD models. Current AFR can be broadly categorized as rule-based and learning-based approaches [1]. Rule-based AFR [2] relies on a set of expert-defined rules to identify machining features based on their geometric and topological properties. Such approaches are commonly adopted in commercial software such as Autodesk Shape Manager (ASM) [3]. Rule-based approaches are roughly including logic rule and expert system [4], graph-based [5], volume decomposition [6], hint-based [7], and hybrid-based [8]. Despite the advantages of rule-based recognition in terms of programmable implementation, there are still many issues to be addressed. Firstly, it is impossible for researchers to define complete and general rules for all possible features that may appear [9]. Secondly, it is challenging to design precise rules for recognizing intersecting and incomplete features whose topology has changed [10].

Learning-based AFR is another promising approach that leverages machine learning (ML) and deep learning (DL) to learn the mappings between machining features and low-level representations from a large

* Corresponding author.
E-mail addresses: wuhongjin@hust.edu.cn (H. Wu), ybpeng@hust.edu.cn (Y. Peng), gaoliang@hust.edu.cn (L. Gao).

https://doi.org/10.1016/j.rcim.2023.102661
Received 13 May 2023; Received in revised form 18 August 2023; Accepted 17 September 2023
Available online 29 September 2023

amount of training samples, rather than relying on explicit rule definitions [11]. This approach has shown great potential in achieving higher accuracy and robustness in recognizing complex and varied machining features [1,12]. However, current learning-based AFR approaches face several challenges. For instance, some methods cannot directly learn feature recognition from boundary representations (B-Rep) and require conversion to a well-studied intermediate representation, such as point cloud, multi-view image, mesh, or voxel. This conversion may result in the loss of topology, geometry, and attributes, leading to less accurate feature recognition [13–15]. Moreover, some algorithms either separate feature instance segmentation and classification into two stages or can only perform one task, resulting in a more complicated and time-consuming combinatorial system [10].

Hence, addressing the aforementioned challenges is essential to further advance the application of learning-based AFR approaches. This paper proposes potential solutions to overcome these limitations:

- Inspired by recent advancements in graph representations [2,13–15], this paper proposes a novel graph structure called the Geometric Attributed Adjacency Graph (gAAG), designed specifically for neutral B-Rep data. The gAAG contains not only UV grids of surfaces and curves presenting geometric information, but also derived Face Adjacency Graph (FAG) encoding topological information. Additionally, it includes extended attributes from both face and edge entities.
- Based on the creation tool of MFCAD++ [13], we releases a open-source Machining Feature Instance Segmentation (MFInstSeg) dataset, which includes over 60k synthetic CAD models labeled with semantic segmentation, instance segmentation, and bottom face segmentation information. By training on this dataset, researchers can develop and evaluate multi-task segmentation networks capable of accurately identifying and distinguishing between different machining features within a given CAD model.
- Referring to recent studies on graph neural networks (GNNs) [16], a new network for machining feature recognition called AAGNet is proposed. AAGNet is designed to perform multi-task machining feature recognition using topological, geometric information, and extended attributes from neutral B-Rep data.

The rest of this paper is structured as follows: In Section 2, recent related works that motivated the proposed approach are reviewed. Section 3 provides the proposed methodology in detail, including the structure of the gAAG, the description of the MFInstSeg dataset, and the architecture of the AAGNet. Section 4 presents the experimental results, where the effectiveness of the proposed approach is demonstrated through various experiments and a comparison is made between the proposed approach and other approaches. Finally, in Section 5, the proposed method is summarized, and potential future research directions are discussed.

2. Related work

2.1. Rule-based approach

Logic rule and expert system [4,17–19] is an early attempt to automate the process of identifying machining features in CAD models by matching their geometric structure to a set of pre-defined rules and patterns. The programming implementation and application of the rule-based approach are well-developed. However, this approach has limitations in that it is unrealistic to encode all the relevant knowledge about machining features in the rules and patterns, which can result in an overly complex and difficult system [9].

Graph-based approach [2,5,8,20] has gained popularity due to their ability to leverage the high structural similarity between the graph and the B-Rep. This approach usually involves constructing an attribute adjacency graph (AAG) that captures relationships and attributes between faces and edges in a 3D CAD model. Then, heuristic algorithms are used to search for matching sub-graphs of features in the AAG. The graph-based approach has some limitations: (1) it is arduous to recognize intersecting and incomplete features whose sub-graph structure has changed [21]. (2) sub-graph matching is a classic NP-hard problem, which makes this method computationally complex [22].

Volume decomposition [23–27] is another well-known rule-based approach for AFR. It decomposes the removal volume of a CAD model into intermediate volumes and reconstructs machining features from them based on pre-defined rules. According to the decomposition process, this approach can be classified into two categories: the convex-hull and the cell-based [28]. The convex-hull represents a solid model as a series of convex volumes and delta volumes of polyhedra, and combining them into potential features according to their Boolean operation relationships. The cell-based decomposes the removable volume into minimal cells and then recombines them into the maximum cells that can be machined. This method is useful for numeric control machining process, but it loses geometric and topological information during decomposition [2]. Additionally, there are many boolean operations in decomposition, leading to high computation costs [29].

Hint-based approach [7,24,30,31] is a promising rule-based method for recognizing intersecting features. It involves extracting geometric, topological patterns, and heuristic information from the remains of intersecting features as hints, followed by applying reasoning to recover the incomplete feature information from the obtained hints. While this approach can partially address the problem of recognizing intersecting features and can be beneficial for subsequent process planning, like other rule-based approaches, it may be challenging for experts to define complete hints and reasoning rules for intersecting features due to topology destruction.

Hybrid approach [32–35] attempts to overcome the limitations of rule-based algorithms by combining different approaches. Several researchers have proposed hybrid systems for feature recognition, leveraging the strengths of multiple techniques. For instance, Rahmani and Arezoo [33] developed a graph-hint hybrid recognition system to identify intersecting milling features. Similarly, Verma and Kumar [34] presented a graph-hint feature recognizer for B-Rep solids of 2.5D parts. Subrahmanyam [35] proposed a hybrid system that combined a hint-based method with cell-based volume decomposition. While hybrid systems aim to improve the accuracy and robustness of feature recognition, like traditional pattern recognition algorithms, they cannot cover all possible patterns of features and include complete rules for matching infinitely varied structures.

2.2. Learning-based approach

To overcome the limitations of the rule-based system, recent research has turned attention to the learning-based approach, particularly using artificial neural networks (ANNs). Unlike rule-based methods that rely on pre-defined rules, ANNs can learn the task of feature recognition directly from a large dataset of CAD models with labeled machining features. The development of modern ANNs depends on several factors, including the scale of labeled datasets, neural architecture design, the hardware performance of computer, and develop tool chains.

Early networks. [36–41] The first application of a multi-layer perceptron (MLP) to recognize machining features from a Face Adjacency Graph (FAG) was done by Prabhakar and Henderson [36]. Over the years, researchers have made many efforts to improve the performance and robustness of early networks [37–39,41]. Due to limited computer performance at the time, early networks had a small receptive field on a B-Rep model [15]. Additionally, early studies lacked an understanding of input representation, resulting in the loss of topological and geometric information in B-Rep and the degradation of network performance [9].

Voxel-based networks. [11,42] Voxel is a data structure in 3D computer graphics that represents a 3D object as a set of regular

volumetric occupancy grids. Due to its success in 3D vision [43,44], Zhang et al. [11] proposed a voxel-based 3D convolutional neural network (CNN) called FeatureNet to achieve AFR. This method first divides the voxelized CAD models into several isolated features using an unsupervised segmentation algorithm and then uses FeatureNet to classify the category of the extracted single features. Zhang et al. also contributed a dataset that includes 24,000 single feature models with 24 classes for training FeatureNet. Building upon this work, Peddireddy et al. [42,45] presented a similar voxel-based method that not only identifies the class of features but also recognizes the machining process of the target model (milling or turning). Ning et al. [22] developed a hybrid AFR system that combined an AAG-based minimum sub-graph searching method for segmenting multiple features with a voxel-based 3D CNN for classifying single features. Voxel-based networks have some drawbacks: (1) the conversion from CAD model to voxel representation results in the loss of topological and geometric information, leading to a reduction in classification performance. (2) The recognition performance of the two-stage method heavily relies on the precision of the segmentation algorithm used in the first stage [29].

Multi-view-based networks. [10,46,47] This approach involves representing a 3D object as a series of multi-view 2D images, which are can be processed by mature 2D visual neural networks. Shi et al. [46] proposed the Multiple Sectional View (MSV) representation to represent 3D CAD models and introduced the MsvNet for feature recognition. MsvNet follows a similar procedure as FeatureNet, where the 3D CAD model is first segmented into individual features. These single features are then converted to multi-view 2D images, which are subsequently classified by MsvNet. Hence, MsvNet is an inefficient and complex two-stage algorithm. SsdNet was introduced as an improved method to overcome the drawbacks of MsvNet. It is a multi-view-based single-stage machining feature recognition method that can be regarded as an 3D extended version of the classic 2D object detection algorithm called single shot multibox detector (SSD) [48]. In rapid sequence, RDetNet was proposed, which is an upgraded version of SsdNet that addresses the challenge of recognizing interacting machining features with small training samples. It incorporates a more advanced neural architecture and several data augmentation strategies to improve performance in identifying highly interacting features. Although these methods show great potential to handle the interacting feature problem, the conversion from 3D CAD model to MSV representation may suffer from loss of fidelity and critical mapping back to the original B-Rep entities [14]. In addition, These approaches could limit their usefulness in subsequent process planning because they produce cuboid bounding boxes that roughly describes the locations of the machining features [9].

Point-cloud-based networks. [9,49,50] The use of point cloud [51,52], which is a set of 3D coordinates and normal vectors that represent the surface or shape of an object, has recently inspired researchers to explore their potential for machining feature recognition. Ma et al. [53] proposed an approach using CNN and PointNet architecture achieves automatic recognition of machining features from 3D point cloud data. Colligan et al. [49] proposed a method for creating and labeling point clouds from B-Rep CAD models for machine learning, while maintaining a link between the two representations. Worner et al. [54] introduced a novel concept for the automated recognition of form features using PointNet, and for assigning connections between these features as joints to construct the assembly. Yao et al. [50] devised a hierarchical neural network-based methodology that is capable of accurately recognizing complex overlapping machining features in point clouds. The proposed approach incorporates an unsupervised segmentation algorithm that splits multi-feature regions, and uses PointNet++ [51] for recognizing individual features. Zhang et al. [9] presented a new multi-task AFR method called Associatively Segmenting and Identifying Network (ASIN) for machining feature recognition based on point cloud data. The proposed method is capable of simultaneously performing semantic segmentation, instance segmentation, and bottom face identification, which enables the efficient recognition of intersecting machining features. To the best of our knowledge, ASIN is the first single-stage machining feature instance segmentation approach on point cloud data, inspired by recent advances in point cloud instance segmentation [55,56]. Although point clouds offer significant advantages in various applications, they have several limitations that must be considered. One issue is that small feature faces may be under-sampled during point cloud generation [15]. Furthermore, point cloud models do not capture topological relations between B-Rep entities, such as vertices, edges, and faces.

Mesh-based networks. [57,58] A mesh is a geometric representation of a 3D object that is composed of edges, vertices, and faces. Polygon meshes are made up of polygon-shaped faces, while regular meshes consist of a single type of polygon. Triangular meshes are a common type of mesh used in various applications [59]. Recently, some approaches [57,58] proposed specialized convolution kernel schemes, attempting to perform different 3D vision tasks on meshes. Jia et al. [60] proposed a novel machining feature recognition method based on Mesh-Faster RCNN, which combines the original MeshCNN with the Faster RCNN. Despite the simplicity of triangular mesh structure, generating high-quality manifold meshes from B-Reps demands special meshing procedures, as direct handling of B-Rep can avoid the conversion between two representations [14].

Graph-based networks. [13,14,61,62] The use of graph representations has gained popularity in both rule-based and learning-based approaches due to the high structural similarity between graphs and B-Rep [1]. Cao et al. [61] contributed a large synthetic datasets of 3D CAD models contains multiple machining features with face labels and its generation tools. Besides, a graph neural network (GNN) was proposed to classify the category of machining faces. One of the Autodesk research teams [14] proposed a novel GNN called UV-Net that encodes the geometry of u and v parameters of the surfaces and curves as well as the topology of the FAG. Another team form Autodesk [14] proposed a specialized graph convolution network called BRepNet designed to operate directly on B-Rep data structures. BRepNet consists of multiple special convolutional kernels which can perform message passing in winged edge structures and aggregate the features from the faces, edges and coedges in the neighborhood of each central coedge. Inspired by graph-based and mesh-based approaches, Colligan et al. [13] presented the Hierarchical CADNet that can learn machining feature recognition from a novel hierarchical graph representation. This graph has two levels: the first level describes the topological structure using a FAG, and the second level presents the geometry of the surface using mesh facets. Wang et al. [62] developed a hybrid learning framework called DeepFeature relying on AAG-based feature extraction algorithm and GNN. The DeepFeature firstly utilizes a human-designed subgraph extraction algorithm to separate single features from CAD models, and then six GNNs are employed to judge the type of the isolated features. These GNN-based AFR can only perform face classification or graph classification task and require auxiliary segmentation algorithms to separate multi-feature instances, making them a complex two-stage process. In contrast, point-cloud-based ASIN [41] simultaneously performs semantic segmentation, instance segmentation, and bottom face identification, making it more efficient. Nonetheless, the above-mentioned graph-based approaches have shown promising results in solving highly intersecting feature recognition problems and efficient representation learning from B-Rep structure.

Other networks. Apart from the existing methods discussed earlier, there have been recent developments in the use of alternative representations of B-Rep models for machining feature recognition. Yeo et al. [63] introduced a novel feature descriptor with attributes of B-Rep face entities as ANN input representation, and a multi-layer perceptron (MLP) to classify the category of machining faces. Fu et al. [64] proposed a new 3D descriptor, Improved Dexel Representation (IDR), which captures precise distance information and high-resolution details for small-scale local features. They also presented a modified dataset

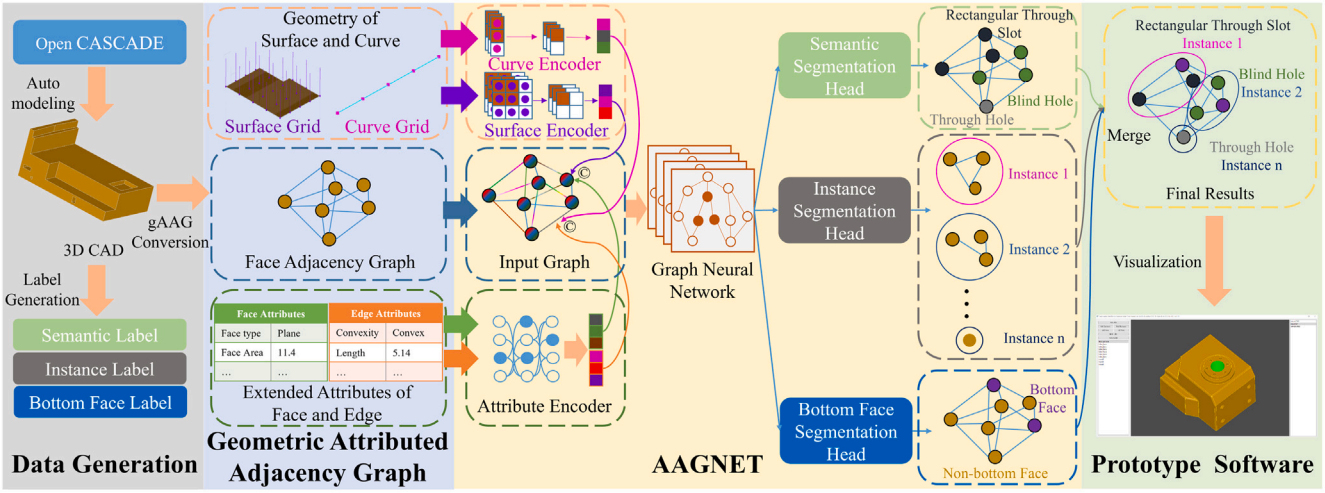

Fig. 1. The pipeline of the proposed framework.

for Manufacturing Process Identification (MPI) to evaluate the effectiveness of the proposed approach. Shi et al. [21,65] introduced a novel shape signature based on heat persistence map (HPM) that can describe both the topology and geometry characteristics of 3D shapes. Then, the HPM is simplified into attributed graph by clustering and fed into a 2D CNN to recognize interacting features. Researchers have explored direct analysis of B-Rep structure using ANN for machining feature recognition tasks, enabling adaptation to a wider range of models. Jian et al. [66] proposed quaternion semantic cell convolution (QSCC) graph neural network for MBD product model classification incorporates a quaternion product semantic cell (QPSC) and a quaternion simple graph convolution (QSGC) to process multi-dimensional PMI information on of the MBD model. Miles et al. [67] developed an automatic STEP analyzer based recursive neural network (RNN) that can be finetuned for specific CAD model datasets. A STEP parser is firstly used to process STEP file into a hierarchical tree structure of vectors which can be used as network inputs. Subsequently, a recursive encoder network receives the vectors and extract key features for downstream single feature classification task. Notwithstanding, the former approach encounters limitations in processing intersecting features, prompting the researchers to devise a novel, resilient, and size-invariant solution for machining feature recognition, as expounded in the later study by Miles et al. [68]. The proposed methodology employs a recurrent neural network to generate a sequence of instances that identify the machining features' classes, overcoming the shortcomings of the previous approach. Despite the potential of using STEP files as inputs for machining feature recognition, they often contain redundant information that can hinder learning performance. Thus, a rule-based parser is utilized in the aforementioned approach to filter out irrelevant information and facilitate efficient learning.

3. Methodology

3.1. Overview of the proposed framework

Drawing on recent advancements in the learning-based machining feature recognition approaches, this paper presents a GNN-based approach for multi-task machining feature recognition. As traditional graph representation methods lack complete information, a novel representation called Geometric Attributed Adjacency Graph (gAAG) was introduced to encode the topological and geometric information as well as extended attributes derived from the B-Rep structure. Furthermore, we also published the Machining Feature Instance Segmentation (MFInstSeg) dataset, containing annotated 60k synthetic CAD models with semantic segmentation, instance segmentation, and bottom face identification labels. It facilitates training and evaluation of multi-task networks for precisely recognizing machining features in CAD models. The proposed framework leverages the information encoded in the gAAG to develop a novel graph neural network, AAGNet, which can simultaneously perform machining feature semantic segmentation, instance segmentation, and bottom face identification, thereby enabling accurate recognition of machining features within a CAD model. The pipeline of the proposed framework is illustrated in Fig. 1.

3.2. Geometric Attributed Adjacency Graph (gAAG) representation

Boundary representation (B-Rep) is a widely used neutral format in CAD to represent the geometric and topological information of 3D objects. The B-Rep primarily entails the depiction of entities through both geometry and topology. Geometric information refers to the dimensions, positions, and shape parameters of the entity, such as point coordinates, edge parameters, and face parameters. On the other hand, topological information involves the internal connectivity relationships among the entity. The B-Rep facilitates the complete recording of both the geometric and topological information of solid models, thereby enabling the precise representation of 3D objects. However, using B-Rep models as input for neural networks can be challenging due to their complexity and non-uniform structure. To overcome this challenge, as mentioned in related works, various techniques have been developed to convert B-Rep Current representations such point cloud, mesh, and voxel are unable to capture the complete geometric and topological information of the B-Rep, which may lead to degraded recognition performance [14]. Besides, Some methods do not have the flexibility to handle the variable number of entities in B-Rep [13]. Hence, a novel B-Rep descriptor called Geometric Attributed Adjacency Graph (gAAG) is proposed, which provides the flexibility to handle varying number of entities in B-Rep models and preserves their geometric, topological information and extended attributes. The gAAG representation is shown in Fig. 2.

3.2.1. Topological information

Topological information in the B-Rep, such as a face adjacency graph (FAG), refers to the relationships between different face entities that make up the representation. The FAG provides a concise and efficient representation of the connectivity between adjacent faces in the B-Rep model. It encodes the relationships between entities, such as edges and faces, in a graph structure, enabling efficient traversal and analysis of the representation. For example, if two faces share an edge in the B-Rep model, there will be an edge connecting the corresponding nodes in the FAG. The construction of a FAG can be readily obtained

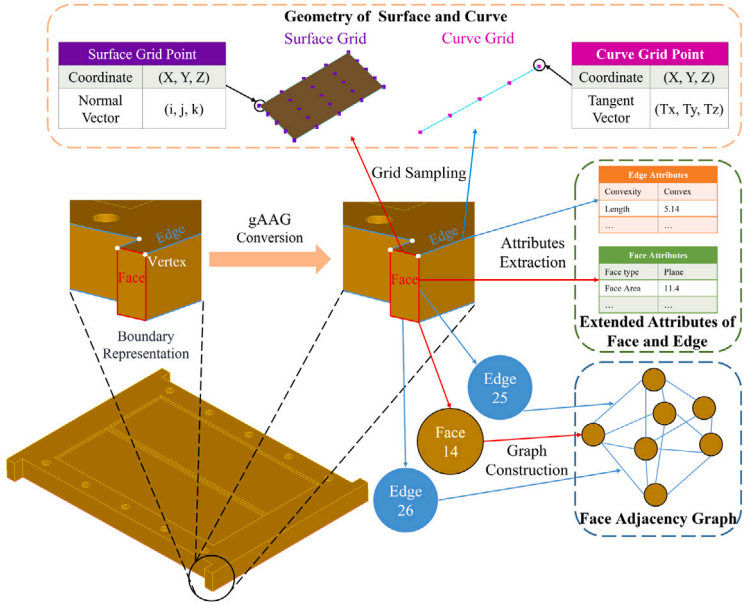

Fig. 2. The structure of Geometric Attributed Adjacency Graph (gAAG). The 'Face 14' denotes the index of this face is 14 in the B-Rep model and the 'Edge 25' denotes the index of this edge is 25.

from the B-Rep using the Open CASCADE API. This graph can then be conveniently stored as an undirected graph by the Deep Graph Library (DGL) [69], a widely used library for graph-based machine learning. The FAG is the fundamental structure of the proposed gAAG, which is shown in Fig. 2.

3.2.2. Geometric information

The representation of geometric information in the B-Rep is based on parametric surfaces and curves, which are defined using parametric equations. But neural networks often struggle to process such data, which has led to the proposal of various methods for discretizing curves and surfaces into discrete structures such as meshes, voxel grids, and point clouds. Nevertheless, these approaches have certain limitations when it comes to accurately describing complex B-Rep models. Recently, a novel approach called UV-Net [14] was introduced, which can effectively encode the most informative geometric and topological data from the B-Rep and convert it into a representation that can be easily and efficiently used with neural networks. The geometry of parametric surfaces and curves is discretized into a regular grid of points with a fixed step size. At each grid point, local features such as 3D absolute point coordinates, 3D absolute surface normal, and unit curve tangent are attached. The UV grids of surfaces and curves are presented in Figure Fig. 2. To some extent, UV grids can be regarded as a special type of point cloud, where the points are uniformly distributed over the grid rather than non-uniformly and disorderly. Compared to other discrete representations, such as voxel grids and meshes, which may falter to accurately describe complex surfaces and curves, UV grids can capture precise geometric information. Hence, the proposed gAAG leverages UV grids as the geometric descriptors of surfaces and curves, which are contained into the nodes and edges of the FAG.

3.2.3. Attributes of faces and edges

In order to enhance the representation of a B-Rep model for feature recognition, we propose to extend the FAG with attributes of faces and edges. This extension is inspired by the traditional Attributed Adjacency Graph (AAG) [5], which is a data structure that captures attributes of a solid model. The attributes of faces include their types (plane, cylinder, cone, sphere, etc.), areas, centroid coordinates, and number of loops. The attributes of edges include their types (line, circle, ellipse, etc.), lengths, convexity (concave, convex, or smooth), and dihedral angles between adjacent faces. The attributes of faces and edges can be easily derived from the B-Rep model and stored as node and edge features in the FAG. This way, we obtain a Geometric Attributed Adjacency Graph (gAAG) that represents both the topology and geometry of a B-Rep model. The gAAG can be used for various downstream tasks such as feature recognition, part classification, and 3D CAD retrieval. The detailed descriptions about attributes of faces and edges are shown in Table 1.

Table 1
Attributes of faces and edges.

Entity	Attribute	Description
Face	Type	The geometric type of the face (plane, cylinder, etc.)
Face	Area	The area of the face
Face	Centroid	The centroid coordinates of the face
Face	Loops	The number of loops in the face
Edge	Type	The geometric type of the edge (line, circle, etc.)
Edge	Length	The length of the edge
Edge	Convexity	The convexity of the edge (concave, convex, or smooth)
Edge	Angle	The dihedral angle between adjacent faces

3.2.4. Summary

In this subsection, we introduce the proposed gAAG representation that aims to capture the precious and complete information in B-Rep. Table 2 summarizes the main features and differences of different representations, including geometry, topology, attribute and the input for neural network. As can be seen from the table, gAAG representation method combines the advantages of UV grid and EAAG, which can effectively capture the global and local information of 3D shape, and has high fidelity and scalability. Compared with other graph representations, gAAG not only considers the topological and geometric information of B-Rep, but also contains the extended attribute of faces and edges in the B-Rep model.

3.3. Machining Feature Instance Segmentation (MFInstSeg) dataset

As expounded in the related work, existing learning-based methods necessitate copious amounts of annotated data for training and evaluation. However, there is few open-source datasets and unified labels for machining feature recognition. Such as Mechanical Components

Table 2
Main features of different representations.

Representation	Geometry	Topology	Attribute	Input for neural network
Voxel	3D grid of voxels (cubic cells)	Implicit connectivity between adjacent voxels	–	3D CNN
Point cloud	3D data points	No explicit connectivity between points	–	Point cloud network (Such as PointNet)
Multi View	2D images from multi viewpoints	No explicit connectivity between images	–	2D CNN
Mesh	Polygon	Explicit connectivity between vertices, edges and faces	–	Mesh network (such as MeshCNN)
AAG	–	Undirected graph (Explicit connectivity between nodes)	Attributes of B-Rep faces and edges	GNN
EAAG	–	Undirected graph	Extended attributes of B-Rep faces and edges	GNN
UV graph [14]	Regular grid of points (UV grid)	Undirected graph	–	GNN
Hierarchical graph [13]	Triangular mesh	Hierarchical graph	Attributes of B-Rep faces and convexity of edges	Hierarchical GNN
gAAG (Ours)	UV grid	Undirected graph	Extended attributes of B-Rep faces and edges	GNN

Benchmark (MCB) [70] and Fusion 360 Gallery [15] do not include the required annotations for machining feature recognition task. Furthermore, some datasets such as MFCAD [61] and MFCAD++ [13] only provide machining feature semantic segmentation labels to recognize the type of machining faces, but this entails additional steps to differentiate the distinct feature instances. Moreover, a few datasets, like FeatureNet dataset [11], are limited to single or simple intersecting features, which cannot reflect the complexity and diversity of real-world parts. Hence, we propose an open-source synthetic dataset called Machining Feature Instance Segmentation (MFInstSeg) to address the aforementioned challenges. The generation of MFInstSeg is based on scripts from MFCAD++, but unlike MFCAD++, our dataset provides instance labels and bottom labels for each machining feature, which are indispensable for segmenting and identifying individual machining features from complex parts. Moreover, our dataset also employs the topological sanity check to ensure the validity of the generated 3D CAD models. The details of our dataset generation, topological sanity check and instance label structure are elucidated as follows.

3.3.1. Dataset generation

To automatically generate a large scale of training data, we employed parametric modeling technology to produce 3D CAD models using Open CASCADE and its python wrapper. The scripts create cuboid stocks of arbitrary size and randomly append machining features to the stock. We can obtain numerous models with intersecting features, ranging from 5 to 14 instances per model and the dataset includes 24 types of common machining features (see Fig. 8 in Appendix A that depicts some examples). Although the creation tool is modified from MFCAD++, the MFInstSeg diverges from it in providing instance labels for each feature, which facilitate segmenting feature instances from intersecting parts. In the subsequent Section 3.3.3, the details of label generation are presented.

3.3.2. Topological sanity check

To guarantee the correctness of the generated 3D CAD models, we performed a topological sanity check on each model using Open CASCADE's built-in tools and the Topology Checker. The topological sanity check verifies that the model is valid, closed, manifold, and has unique coedges. A valid model means that it passes the BRepCheck Analyzer test, which checks for various defects such as self-intersections, gaps, or degenerated shapes. A closed model means that it has no free edges or missing faces. A manifold model means that it has no non-manifold shells or faces shared by more than two shells. A unique coedge means that each coedge belongs to exactly one wire loop. These conditions are essential for a proper representation of machining features and their instances. If any defect is detected, the model will be discarded to improve the quality of the data and facilitate the training of models.

3.3.3. Instance label structure

Semantic segmentation label. We consider semantic segmentation as a multi-class node classification problem on the adjacency graph derived from a B-Rep model of a 3D shape. The objective is to assign each node to one of the predefined classes that correspond to different machining features. For instance, in Fig. 3, the face with id 12 in the CAD model is a through slot face and its corresponding node on the graph has the label of through slot class (the mapping of machining feature class indices is consistent with MFCAD++ [13]).

Instance segmentation label. In traditional graph-based machine feature recognition, the subgraphs of machining features are predefined and then subgraph search on the attributed adjacency graph is performed to identify the matched machining features. This method fails to handle intersecting features due to the alteration of their subgraph structures. Therefore, we perform graph generation rather than subgraph search on the attributed adjacency graph, which aims to directly generate the corresponding subgraph of the machining features. To further simplify this task, we consider it as a link prediction problem, which can estimate the probability of a connection between two nodes in a graph based on the observed connections, attributes, and the structure of the graph. As shown in Fig. 3, the example part contains two instances of rectangular pockets. The rectangular pocket 1 is composed of faces 24, 25, 26, 27, and 28. Hence, the corresponding instance label of this pocket has links connecting these nodes (faces). On the other hand, there are no edges between nodes from distinct instances, such as faces 29 that belongs to the rectangular pocket 2. Moreover, some nodes are isolated, indicating the stock faces that are not part of any instance. This linking relationship is easily described by an adjacency matrix $A \in \mathbb{R}^{n\times n}$ where n is the number of nodes in the graph. If there is a link between nodes n_i and n_j then

$$A_{ij} = \begin{cases} 1 & \text{if link exist between } n_i \text{ and } n_j, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Additionally, self-loops are also added where $A_{ii} = 1$.

Bottom face segmentation label. The bottom face identification is essential for the subsequent process planning, which can affect some machining parameters such as tool path direction. Hence, the bottom face labels are included in the proposed dataset. Unlike semantic segmentation, the bottom face segmentation is a simple task that can be viewed as a binary node classification problem on the graph. For

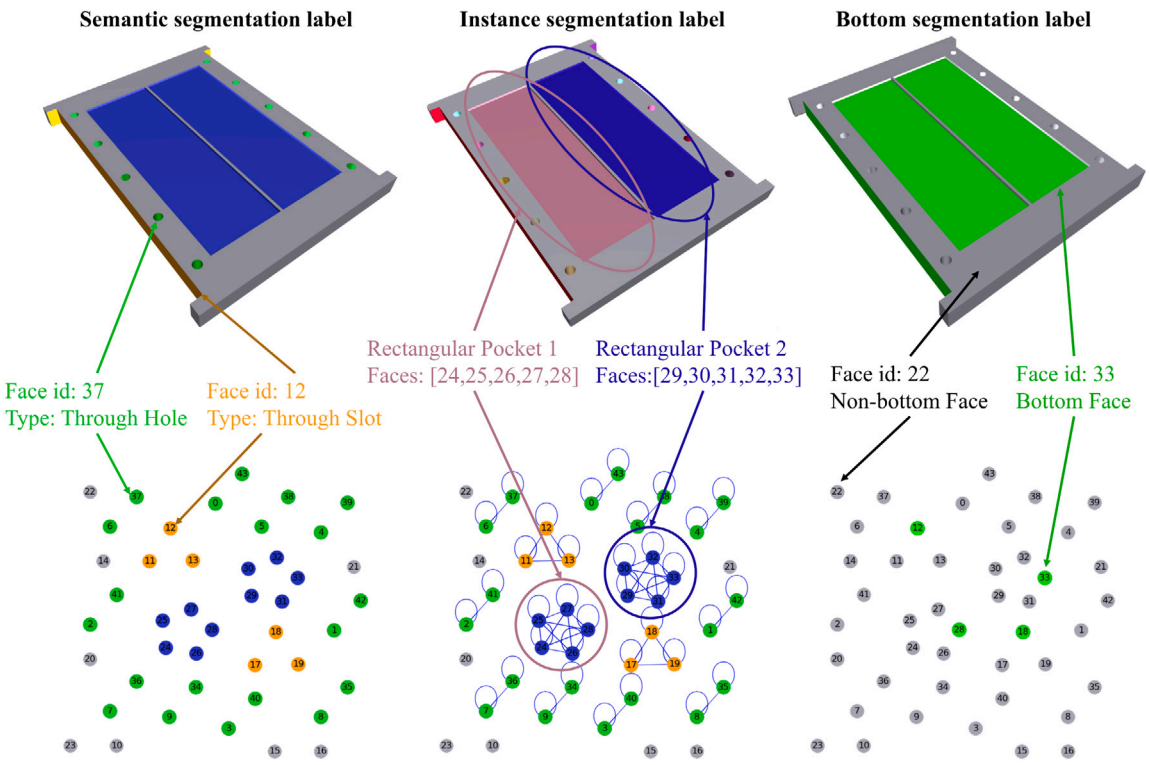

Fig. 3. The semantic, instance, and bottom face label of MFInstSeg.

example, in Fig. 3, the face with id 33 is the bottom face of the rectangular pocket 2, so its corresponding node in the graph has a label of 1. On the contrary, the face with id 22 is not a bottom face and its node label is 0.

3.3.4. Summary

This section delineates the challenges and limitations of existing open-source datasets for machining feature recognition and proposes a novel synthetic dataset called Machining Feature Instance Segmentation (MFInstSeg). The extant datasets lack the requisite annotations for the task, provide only semantic segmentation labels or are confined to single features or simple intersecting features. Conversely, MFInstSeg provides instance and bottom face labels for each of the 24 types of prevalent machining features in complex parts. The dataset is generated using parametric modeling technology and scripts from MFCAD++ with a topological sanity check to ensure model validity. The semantic segmentation is modeled as a multi-class node classification problem, and instance segmentation is performed by directly generating the corresponding subgraphs of the machining features using a link prediction approach. The bottom face segmentation is considered as a binary node classification problem. The proposed dataset is anticipated to surmount the limitations of current datasets and facilitate the development of learning-based methods for machining feature recognition. Table 3 summarizes some datasets related to machining feature recognition.

3.4. AAGNet: A graph neural network designed for multi-task machining feature instance segmentation

Graph neural networks (GNNs) are a class of deep learning models that can operate on irregular data structures such as graphs. They have been successfully applied to various domains such as recommendation system, knowledge graph reasoning, and computer vision [74,75]. Recently, GNNs have also been explored for machining feature recognition [13–15], as boundary representation can be conveniently represented as graphs where nodes are faces and edges are connections between adjacent faces.

This paper proposes a novel GNN model, named AAGNet, which is designed to perform multi-task machining feature recognition using topological, geometric, and extended attributes from neutral B-Rep models. AAGNet comprises three main components: an input encoder, a graph encoder, and a multi-task head, as depicted in Fig. 4.

The input encoder employs a plain 2D Convolutional Neural Network (2D CNN) to extract features from a grid of uniform surface points with 3D coordinates and normal vectors. Furthermore, a 1D CNN is utilized to encode the curve points with 3D coordinates and tangent vectors. Additionally, two Multi-layer Perceptrons (MLPs) generate the embeddings of face and edge attributes, respectively. Therefore, the gAAG is a graphical representation for B-Rep model, which can be described as $G = \{N, E\}$ where the nodes N represent the surfaces and the edges E denote the curves in the B-Rep. Then, the face attribute embeddings and surface feature vectors are concatenated into the nodes of the face adjacency graph, while edge attribute embeddings and curve feature vectors form the edges of the graph. The graph encoder utilizes multiple GNN blocks, each comprising a Message Passing Neural Network (MPNN) and an MLP layer, to extract high-level features from the input graph. The graph decoder generates the output graph with the same topology as the input graph, but with different node features encoding global and local information. The multi-task head includes semantic segmentation, instance segmentation, and bottom face segmentation heads, each providing outputs for the relevant tasks. The semantic segmentation head provides the probability of a sample belonging to machining feature classes for each node (face), while the instance segmentation head outputs a link score matrix, predicting the possibility of a link between each pair of nodes. The bottom face segmentation head provides the likelihood of whether each node (face) is a bottom face.

3.4.1. Encoder

Input encoder comprises a 2D CNN called surface encoder, a 1D CNN called curve encoder, and two MLPs as the face and edge attribute encoder. The UV grids of surfaces, which have a shape of $6D \times GridSize^2$ containing 3D coordinates and normal vector, are fed

Table 3
Some datasets related to machining feature recognition.

Dataset	Source	Format	Open-source	Task	Scale
Mechanical Components Benchmark (MCB) [70]	Real	OBJ	Yes	Mechanical components classification and retrieval	58,696 mechanical 3D components with 68 classes
ABC [71]	Real	Multiple formats (such as STEP, OBJ, Parasolid)	Yes	Differential quantities, patch segmentation, geometric feature detection	Over 1 millions without label
Fusion 360 Gallery [15]	Real	STEP	Yes	Modeling operation segmentation	35,858 models with 8 modeling operation classes
FabWave [72]	Real	STEP	Yes	Mechanical part classification	5373 models with 52 modeling operation classes
FeatureNet [11]	Synthetic	STL	Yes	Single feature recognition	144,000 models with 24 machining feature classes
FeatureNet+ [64]	Synthetic	STL & STEP	No	Single feature recognition	216,000 models with 36 machining feature classes
MPI [42]	Synthetic	STL	Yes	Single feature recognition machining process recognition manufacturability analysis	18,000 models with multiple labels
Ning et al. [22]	Synthetic	STL & STEP	No	Single feature recognition	140,000 models with 14 machining feature classes
Shi et al. [10,46,47]	Synthetic	STL	Yes	Intersecting feature detection	1000 models with 24 machining feature classes
MFCAD [61]	Synthetic	STEP	Yes	Machining feature semantic segmentation	15,488 models with 15 machining feature classes
MFCAD++ [13]	Synthetic	STEP	Yes	Machining feature semantic segmentation	59,665 models with 24 machining feature classes
Zhang et al. [41]	Synthetic	Point cloud	Yes	Machining feature instance segmentation and bottom face recognition	55,000 models with 8 machining feature classes
MFDataset [73]	Synthetic	Point cloud & STL	Yes	Single feature recognition	66,000 models with 33 machining feature classes
MFInstSeg (ours)	Synthetic	STEP	Yes	Machining feature instance segmentation and bottom face recognition	62,495 models with 24 machining feature classes

Table 4
The details of surface encoder. $k = 3$ denotes that convolution kernel size is set to 3. The data format of input and output is $channel \times height \times width$.

Operator	Input $(C \times H \times W)$	Output $(C \times H \times W)$
Conv2D, k = 3	$6D \times GridSize^2$	$16D \times GridSize^2$
BatchNorm2D	$16D \times GridSize^2$	$16D \times GridSize^2$
Activation	$16D \times GridSize^2$	$16D \times GridSize^2$
Conv2D, k = 3	$16D \times GridSize^2$	$32D \times GridSize^2$
BatchNorm2D	$32D \times GridSize^2$	$32D \times GridSize^2$
Activation	$32D \times GridSize^2$	$32D \times GridSize^2$
Conv2D, k = 3	$32D \times GridSize^2$	$64D \times GridSize^2$
BatchNorm2D	$64D \times GridSize^2$	$64D \times GridSize^2$
Activation	$64D \times GridSize^2$	$64D \times GridSize^2$
GAP2D	$64D \times GridSize^2$	$64D \times 1^2$
Flatten	$64D \times 1^2$	$64D$

Table 5
The details of curve encoder. The data format of input and output is $channel \times length$.

Operator	Input $(C \times L)$	Output $(C \times L)$
Conv1D, k = 3	$6D \times GridSize$	$16D \times GridSize$
BatchNorm1D	$16D \times GridSize$	$16D \times GridSize$
Activation	$16D \times GridSize$	$16D \times GridSize$
Conv1D, k = 3	$16D \times GridSize$	$32D \times GridSize$
BatchNorm1D	$32D \times GridSize$	$32D \times GridSize$
Activation	$32D \times GridSize$	$32D \times GridSize$
Conv1D, k = 3	$32D \times GridSize$	$64D \times GridSize$
BatchNorm1D	$64D \times GridSize$	$64D \times GridSize$
Activation	$64D \times GridSize$	$64D \times GridSize$
GAP1D	$64D \times GridSize$	$64D \times 1$
Flatten	$64D \times 1$	$64D$

into the surface encoder, which outputs a $64D$ vector representing the extracted information of the surface. Analogously, the curve encoder processes the U grids of curves with a shape of $6D \times GridSize$ containing 3D coordinates and tangent vector and also produces a $64D$ embedding representing the curve. The configuration of the surface encoder is shown in Table 4 and curve encoder in Table 5. The input dimension of face attribute is $10D$ and edge is $12D$. Then both input vectors are transformed into two $64D$ embeddings by two MLPs that consist of linear, layer normalization, and activation layers. Notably, The weights of the surface and curve encoder as well as attributes encoders are shared among all edges and faces in a graph, respectively, ensuring permutation invariance [14]. Finally, the embeddings of surface grid and attributes are concatenated and assigned to the corresponding node in the Face adjacency graph. The same operation is performed on the embeddings of curve. Next, a graph neural network receives the graph as input.

Graph Encoder is a graph neural network that leverages message passing between the nodes of graphs to capture their structural dependencies. As depicted in Table 5, the graph encoder comprises n GNN blocks, each of which adopts a MetaFormer-style [76] architecture and can be formulated as:

$$\begin{aligned} h'_i &= h_i + \mathrm{MPNN}(\mathrm{Norm}_1(h_i)), \\ h_{i+1} &= h'_i + \sigma(\mathrm{Norm}_2(h'_i)W_1)W_2, \end{aligned} \tag{2}$$

where Norm denotes layer normalization, σ denotes nonlinear activation function, W_1 and W_2 are learnable parameters in the MLP, h_i denotes the hidden node embeddings in graph block $i \in 1 \dots N$, where N is the number of blocks. We augment the GNN model by appending MLPs after each GNN layer in accordance with recent studies [77,78]

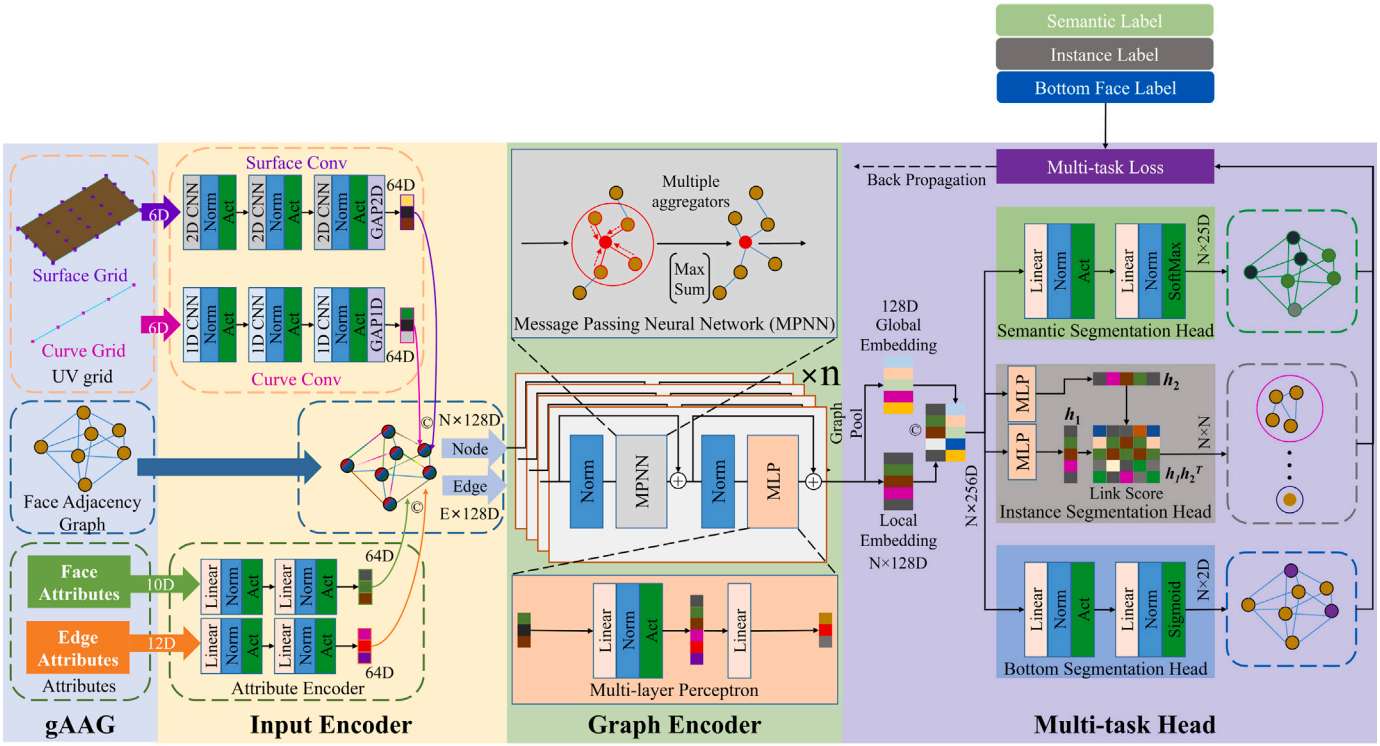

Fig. 4. The architecture of AAGNet. The N denotes the number of nodes, the E denotes the number of edges, D denotes the dimension of the input vector, the GAP denotes the Global Average Pooling layer, the © denotes concatenate operator, the Norm denotes normalization, the Act denotes nonlinear activation function, and the ⊕ denotes add operator.

that indicate that adding MLPs after GNN layers could alleviate over-smooth and improve the performance. In addition, all the AAGNet blocks except the first one have shared parameters.

And message passing is a key procedure in GNNs, which consists of three main steps: (1) Message: Each node in the graph computes a message for each of its neighbors. The message is a function of the node feature, the neighbor feature, and the edge feature. (2) Aggregate: Each node aggregates the messages it receives from its neighbors using a permutation-invariant function, such as sum, mean, max, or min. The aggregation function ensures that the order of the messages does not affect the outcome. (3) Update: Each node updates its feature as a function of its current feature and the aggregated messages. The update function can be any differentiable function, such as a linear layer, a neural network, or an activation function. By applying message passing iteratively, each node can incorporate information from its local neighborhood and eventually from the entire graph. Inspired by the Principal Neighborhood Aggregation Layer (PNALayer) [79], which can capture diverse moments of the node and edge features by employing various combinations of aggregation methods, we adopt PNALayer as the Message passing neural network (MPNN) that can be formulated as follows:

$$h_i' = M(h_i, \oplus_{(i,j)\in E} A(h_i, e_{i,j}, h_j)) \tag{3}$$

where h_i and $e_{i,j}$ denote the node embeddings and edge embeddings, respectively, of a graph with a set of edges E. i and j are the indices of the source node and the destination node of an edge. $\oplus$ represents the concatenation operators. A and M are Aggregating and Mixing MLPs, which compute the output features by concatenating the input features. Multiple aggregators collect messages from neighboring nodes and modify the collected messages in various ways. In the AGGNet, we use max and sum as message aggregators.

3.4.2. Multi-task head

After feature extraction by the GNN encoder, an average pooling operation is applied over the nodes in the graph. This operation pools the $N \times 256D$ shaped graph embeddings into a $1 \times D$ shaped global graph feature vector, which is then concatenated with the local features of each node. The resulting concatenated feature vector is then passed through a multi-task head, which can simultaneously predict several properties of the nodes in the graph. These properties include the probability of each node belonging to a specific class, the probability of links existing between each pair of nodes, and the probability of each node being a bottom surface.

Semantic segmentation head aims to classify each node (face) into one of the predefined machining feature classes, such as hole, slot, pocket, etc. The semantic segmentation head is a Softmax classifier. The output of the semantic segmentation head is a $N \times 25D$ vector of probabilities for each node, indicating the confidence of belonging to each class. The loss function for the semantic segmentation head is the cross-entropy (CE) loss between the predicted probabilities and the ground truth labels. The total loss for the semantic segmentation head is the mean of the losses over all nodes, which can be expressed as:

$$L_s = -\frac{1}{N}\sum_{i=1}^{N}\sum_{i=1}^{K} t_i \log(p_i) \tag{4}$$

where K is the number of class, which is 25 in MFInstSeg dataset. p_i denotes the predicted probability vector of a node belonging to class i, and t_i is the one-hot vector label, when $label = i$, $\hat{y}_i = 1$, otherwise $\hat{y}_i = 0$. N is the number of nodes.

Instance segmentation head aims to group nodes (faces) that belong to the same machining feature instance. The instance segmentation head consists of two MLPs followed by a sigmoid activation function. The head is composed of two multi-layer perceptrons (MLPs) f_1 and f_2 that output node representations h_1 and h_2. These representations are combined with dot product to yield a symmetric matrix S of link scores for each pair of nodes, indicating the probability of being connected in the same instance. The sparse link score matrix S can be computed by:

$$h_1 = f_1(h)$$
$$h_2 = f_2(h) \tag{5}$$
$$S = h_1 h_2^T$$

where h is the output embeddings from the graph encoder.

The loss function for the instance segmentation head is the binary cross-entropy (BCE) loss, which measures the distance between the predicted link scores S and the ground truth adjacency matrix A. The ground truth adjacency matrix is a binary matrix that denotes the existence of a connection between two nodes, and total instance segmentation loss L_i is computed as the mean of the BCE losses over all node pairs:

$$\mathrm{BCE}(t,p) = -t\log(p) - (1-t)\log(1-p)$$
$$L_i = \frac{1}{N^2}\sum_{i,j=1}^{N} \mathrm{BCE}(A_{ij}, S_{ij}) \tag{6}$$

where A_{ij} is the true adjacency value between node i and j, S_{ij} is the predicted link score between node i and j, and N is the number of nodes.

Bottom face segmentation head aims to identify nodes (faces) that are bottom faces. The bottom face segmentation head consists of a MLPs followed by a sigmoid activation function. The output of the bottom face segmentation head is a vector of probabilities for each node, indicating the likelihood of being a bottom face. The loss function for the bottom face segmentation head is the binary cross-entropy (BCE) loss between the predicted probabilities and the ground truth labels. And the total bottom face loss segmentation is computed as the mean of losses over all nodes as:

$$L_b = \frac{1}{N}\sum_{i=1}^{N} \mathrm{BCE}(t_i, p_i) \tag{7}$$

where N is the number of nodes, t_i is the binary true label, and p_i is the predicted probability.

3.4.3. Loss function

To facilitate the joint optimization of the three tasks of semantic segmentation, instance segmentation and bottom face segmentation, we devise a multi-task loss function that combines the losses of each task with different weights. The multi-task loss function can be formulated as:

$$L = \alpha L_s + \beta L_i + \gamma L_b \tag{8}$$

where α, β and γ are the hyper-parameters that regulate the relative significance of each task. To avoid the arduous process of searching for hyper-parameters, we simply assign α, β and γ to be identical as 1. The multi-task loss function enables our model to learn a shared representation for different tasks and leverage the complementary information among them. In our experiment, the AdamW optimizer [80] is employed to minimize the multi-task loss function and update the model parameters.

4. Experimental results and discussion

This section presents extensive experimental results on various open-source machining feature recognition datasets, such as MFCAD and MFCAD++ segmentation dataset, to assess the performance of the proposed AAGNet. Some current datasets like MFCAD and MFCAD++ only provide semantic segmentation labels, which cannot be used for evaluating instance segmentation tasks. Moreover, Zhang et al. [41] contribute a machining feature instance segmentation dataset, but only give point cloud format of CAD models, which cannot be evaluated for B-Rep-based methods. Therefore, we introduce the MFInstSeg dataset containing over 60,000 STEP files with 24 machining feature classes and their semantic segmentation, instance segmentation, and bottom face segmentation labels. The AAGNet is compared with some well-known graph encoders on the MFInstSeg dataset. More details about MFCAD, MFCAD++, and MFInstSeg are given in Appendix C.

Table 6
Face-level semantic segmentation performance on MFCAD test set. θ denotes the number of parameters of each network. The bold number is the best result in each column.

Network	Accuracy (%)	mIOU (%)	θ
UV-Net	99.95 ± 0.02	99.87 ± 0.03	1.23M
PointNet	32.13 ± 7.92	7.15 ± 5.22	0.87M
PointNet++	91.35	–	1.42M
DGCNN	82.50 ± 2.46	67.70 ± 4.73	0.98M
GNN	–	93.60	0.53M
MeshCNN	99.89 ± 0.01	99.70 ± 0.06	2.29M
Hierarchical CADNet	99.90	–	6.60M
AAGNet	**99.99 ± 0.01**	**99.99 ± 0.01**	**0.38M**

4.1. Comparative results on MFCAD

Cao et al. [61] contributed a synthetic machining feature segmentation dataset called MFCAD, which consists of 15,488 CAD models with 16 planar machining feature classes. In this task, the neural network is designed to predict the category of machining feature that each B-Rep surface entity of the CAD model belongs to. Therefore, accuracy is used to evaluate the performance of the classifiers, which is defined as follows:

$$\text{Accuracy} = \frac{\text{Number of correctly classified faces}}{\text{Total number of faces}} \tag{9}$$

Another widely used metric for semantic segmentation is the mean Intersection Over Union (mIOU), which measures the similarity and diversity of two sets of samples. It is defined as the ratio of the intersection to the union of the sample sets. It is computed by taking the IOU of each class and averaging them:

$$\text{mIOU} = \frac{1}{C}\sum_{c=1}^{C} \frac{A_c \cap B_c}{A_c \cup B_c} \tag{10}$$

where C is the number of classes, A_c is the set of faces that belong to class c in the ground truth label, and B_c is the set of faces that belong to class c in the predicted label. All of our metric calculations are done using the TorchMetrics API [81].

We implemented AAGNet and trained it on a cloud server with 12x Intel Xeon Gold 6230R CPU @ 2.10 GHz, 32G RAM, NVIDIA V100S-32G using the Deep Graph Library (DGL) v0.93. We set the number of AAGNet blocks to 3 and the hidden embedding to 64. We used a dropout rate of 0.25 for MLPs. We set the MLP ratio to 2, meaning that the hidden dimension of each MLP layer is two times larger than the input dimension. The nonlinear activation function of AGGNet was set to Mish [82]. The grid size of surface and curve is 5×5. We also used drop path with a rate of 0.25 to regularize the network. The batch size is 256, and the learning rate uses a cosine decay scheduler with an initial value of 0.01. The optimizer is AdamW with 0.01 weight decay. Moreover, we used Exponential Moving Average (EMA) to update the parameters of the network by combining the current parameters with the previous EMA values and the EMA decay value is set to 0.998. For the semantic segmentation task, we removed the instance segmentation head and bottom face segmentation head. We used the official dataset splits provided by MFCAD [61] for fair comparison. And the data is partitioned into a 60/20/20% train/validation/test split.

The results on the MFCAD test dataset are shown in Table 6. The results of the UV-Net [14], MeshCNN [57], PointNet [83], DGCNN [84], GNN [61] are reported in [14]. PointNet++ [51], Hierarchical CADNet [13] are obtained from [13]. The reported values are the mean over 10 trials with a fixed random seed 42 and the error bars denote the standard deviation. We trained each network for 350 epochs in accordance with UV-Net [14]. Furthermore, for networks whose predictions are not based on the B-Rep face but a primitive such as a point or edge, per-face voting is used to determine the prediction per B-Rep face. Additional details about network training are presented in Appendix B.

Table 7
Face-level semantic segmentation performance on MFCAD++ test set. θ denotes the number of parameters of each network. The bold number is the best result in each column.

Network	Accuracy (%)	mIOU (%)	θ
PointNet++	85.88	–	1.42M
DGCNN	85.98	–	0.53M
Hierarchical CADNet	97.37	–	9.76M
AAGNet	**99.26 ± 0.02**	**98.66 ± 0.02**	**0.38M**

The Table 6 demonstrates that AAGNet outperforms the other networks in all three metrics, indicating that it is the most effective and efficient network for semantic segmentation on the MFCAD dataset. AAGNet achieves an accuracy of 99.99 ± 0.01% and a mean IOU of 99.99 ± 0.01%, which are significantly higher than the state-of-the-art results reported by previous works. AAGNet also has the smallest number of parameters (0.38M), which means that it is less prone to overfitting and requires less memory and computation. AAGNet is a graph-based network that leverages topological and geometric information of the B-Rep models to learn better features for semantic segmentation. The table also shows that point cloud networks such as DGCNN, PointNet have inferior performance than graph-based networks, suggesting that they are not suitable for machining semantic segmentation on the MFCAD dataset. The mesh-based network MeshCNN have intermediate performance, with varying trade-offs between accuracy, mean IOU, and number of parameters.

4.2. Comparative results on MFCAD++

MFCAD only includes a small number of features in each CAD model, and does not capture the complexity of intersecting machining features. To address this limitation, MFCAD++ [13] is proposed, which consists of more challenging intersecting CAD models. This dataset contains 59,655 CAD models with 24 types of machining features with both planar and non-planar faces. Each CAD model has 3 to 10 machining features, which increases the difficulty and diversity of the dataset. We used the official dataset splits provided by MFCAD++ [13] for fair comparison. And the whole dataset is partitioned into a 70/15/15% train/validation/test split. Since MFCAD and MFCAD++ are both semantic segmentation datasets, the network parameters of AAGNet are kept the same except for the change in output category from 16 to 24, and the training parameters and evaluation metrics are also consistent with the settings in MFCAD. The results on the MFCAD++ test dataset are presented in Table 7. The results of DGCNN, PointNet++, and Hierarchical CADNet are taken from [13].

Table 7 compares the performance of different models on the MFCAD++ dataset. The table shows that the proposed AAGNet achieves the highest accuracy and mIOU scores among all models, with 99.26 ± 0.02% and 98.66 ± 0.02%, respectively. These scores indicate that AAGNet can segment faces with high accuracy, and can capture the fine-grained details of the CAD models. The table also shows that AAGNet has the smallest number of parameters among all models, with only 0.38M. This indicates that AAGNet is a lightweight and efficient model that can run on resource-limited devices. The table demonstrates the effectiveness and superiority of the proposed AAGNet model.

4.3. Comparative results on MFInstSeg

We conducted an experiment to evaluate the performance of our proposed AAGNet for machining feature recognition on the MFInstSeg dataset, which consists of 62,495 STEP files with annotated machining feature instances and its bottom faces. We compared AAGNet with several famous graph encoders, namely Graph Convolutional Network (GCN) [85], DeeperGCN [86], GraphSAGE [87], Graph Isomorphism Network (GIN) [88], Graph Attention Network (GAT) [89] and GATv2 [90], by replacing the graph encoder module of our network with each of them and keeping the rest of the network architecture and training parameters unchanged. Besides, both AAGNet and ASIN share a common multi-task learning paradigm. To accurately assess the advantages of the graph-based method, we also evaluated ASIN on MFInstSeg dataset. More implementation details are presented in Appendix D. We also enabled the instance segmentation head and bottom face segmentation head for multi-task learning. The MFInstSeg dataset was randomly split into 70% training set (43,745 models), 15% validation set (9375 models), and 15% test set (9375 models).

To access the face-level performance of our model on instance segmentation and bottom face segmentation, we use accuracy and mean Intersection over Union (mIoU) as the metrics. Instance segmentation is essentially a link prediction task, where we need to assign each face to a machining feature instance. To measure the quality of the predicted links, we use accuracy and F-1 score, which is the harmonic mean of precision and recall. The F-1 score is defined as follows:

$$\begin{aligned} \text{F-1} &= \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \\ \text{Precision} &= \frac{TP}{TP + FP} \\ \text{Recall} &= \frac{TP}{TP + FN} \end{aligned} \tag{11}$$

where TP (True Positive) is the number of face correctly assigned to an instance, FP (False Positive) is the number of faces incorrectly assigned to an instance, and FN (False Negative) is the number of faces incorrectly unassigned.

One of the main challenges of feature recognition is to deal with complex and intersecting features, which requires the network to not only identify the feature types, but also locate the faces that constitute the features. Therefore, it is important to evaluate the performance from feature level. Feature-level evaluation measures how well the network can recognize and localize the feature types and instances. As suggested in [46], we use F-1 score as the metric for both feature-level recognition and localization. For recognition performance, the true positive value TP for all B-Rep models in test set is calculated as: $TP = \min(y, \hat{y})$ where TP refers to the true positive value which is the number of correctly recognized features in all B-Rep models of test set, $\hat{y}$ is the number of predicted features, and y is the actual number of features. For localization performance, as mentioned in [10], a proposal is considered as true positive only when the IoU between the ground truth and predicted boxes is greater than a given threshold (usually 0.5). However, unlike MsvNet and SsdNet that output bounding boxes for the recognized features, our proposed network outputs the faces of the recognized features, which are a set of face ids. Hence, a proposal is considered as true positive only when the IoU between the predicted face set and the ground truth face set is 1, which means that they are identical. Then, the recognition and localization F-1 scores can be computed according to Eq. (11).

Table 8 compares the face-level performance of various graph neural encoders and a point cloud-based encoder (ASIN) on three segmentation tasks: semantic segmentation, instance segmentation, and bottom face segmentation. We report the accuracy, mIOU, and F-1 score for each task, as well as the number of parameters of each network/encoder. As shown in the table, our proposed AGGNet model outperforms other graph-based networks on semantic segmentation, achieving 99.15 ± 0.03% accuracy and 98.45 ± 0.04% mIOU with only 0.44M parameters. This demonstrates the efficiency and compactness of AGGNet for semantic segmentation. On the other hand, DeeperGCN achieves the highest performance on instance segmentation and bottom face segmentation, reaching 99.94 ± 0.01% accuracy and 98.97 ± 0.02% F-1 score on the former task, and 99.77 ± 0.01% accuracy and 98.57 ± 0.03% mIOU on the latter task. However, DeeperGCN has a much larger number of parameters (2.77M) than AGGNet, indicating a trade-off between performance and complexity for different graph-based networks. We also compared our graph-based methods

Table 8
Face-level semantic segmentation, instance segmentation, and bottom face segmentation performance on MFInstSeg test set. θ denotes the number of parameters of each network. The bold number is the best result in each column.

Network/Encoder	Semantic segmentation		Instance segmentation		Bottom face segmentation		θ
	Accuracy (%)	mIOU (%)	Accuracy (%)	F-1 (%)	Accuracy (%)	mIOU (%)	
GCN	90.62 ± 0.11	84.44 ± 0.18	99.61 ± 0.01	92.53 ± 0.05	96.41 ± 0.08	79.26 ± 0.48	0.57M
DeeperGCN	99.03 ± 0.02	98.31 ± 0.01	**99.94 ± 0.01**	**98.97 ± 0.02**	**99.77 ± 0.01**	**98.57 ± 0.03**	2.77M
GraphSAGE	97.69 ± 0.06	95.70 ± 0.14	99.82 ± 0.01	96.64 ± 0.17	99.48 ± 0.04	96.83 ± 0.24	1.09M
GIN	98.14 ± 0.03	96.52 ± 0.06	99.83 ± 0.01	96.76 ± 0.08	99.54 ± 0.01	97.17 ± 0.03	0.90M
GAT	95.91 ± 0.12	92.66 ± 0.18	99.86 ± 0.1	97.43 ± 0.06	98.92 ± 0.03	93.46 ± 0.16	1.24M
GATv2	95.90 ± 0.20	93.03 ± 0.36	99.80 ± 0.01	96.35 ± 0.20	98.95 ± 0.06	93.64 ± 0.36	0.51M
AGGNet	**99.15 ± 0.03**	**98.45 ± 0.04**	**99.94 ± 0.01**	98.84 ± 0.07	99.75 ± 0.01	98.47 ± 0.05	**0.44M**
ASIN (120 epochs)	86.46 ± 0.45	79.15 ± 0.82	98.29 ± 0.07	73.20 ± 0.86	98.25 ± 0.14	89.65 ± 0.78	6.14M

Table 9
Feature-level recognition and localization performance on MFInstSeg test set. θ denotes the number of parameters of each network. The bold number is the best result in each column.

Network/Encoder	Recognition F-1 (%)	Localization F-1 (%)	θ
GCN	89.47 ± 0.10	69.24 ± 0.17	0.57M
DeeperGCN	**98.78 ± 0.01**	**96.79 ± 0.08**	2.77M
GraphSAGE	96.01 ± 0.11	86.50 ± 0.47	1.09M
GIN	96.96 ± 0.10	89.58 ± 0.34	0.90M
GAT	95.42 ± 0.13	87.16 ± 0.44	1.24M
GATv2	95.42 ± 0.25	86.32 ± 1.09	0.51M
AGGNet	98.71 ± 0.11	95.88 ± 0.43	**0.44M**
ASIN (120 epochs)	83.35 ± 0.83	60.14 ± 2.61	6.14M

with ASIN, which is also a feature instance segmentation network based on PointNet. In this experiment, ASIN was trained for 120 epochs on the MFInstSeg dataset, while our graph-based methods were trained for 100 epochs. However, ASIN achieves much lower performance than our networks on all three tasks. For example, ASIN only achieves 86.46 ± 0.45% accuracy and 79.15 ± 0.82% mIOU on semantic segmentation, while our AGGNet achieves better performance with fewer parameters. This shows that graph-based networks are more suitable for machining feature recognition than point cloud-based networks, as they can explicitly utilize the topological information between different B-Rep entities.

The Table 9 compares the feature-level recognition and localization performance of different network architectures on the MFInstSeg test set. The F-1 score is used as the evaluation metric for both tasks. The table also shows the number of parameters of each network, which reflects its complexity and efficiency. Among the networks, DeeperGCN achieves the highest F-1 scores for both recognition (98.78 ± 0.01%) and localization (96.79 ± 0.08%). This network has a relatively large number of parameters (2.77M), which may indicate its high expressive power and ability to capture complex feature relationships. However, AGGNet also performs very well, with F-1 scores of 98.71 ± 0.11% and 95.88 ± 0.43% for recognition and localization, respectively, while having the smallest number of parameters (0.44M). On the contrary, ASIN, which uses a point cloud paradigm, has the lowest F-1 scores for both recognition (83.35 ± 0.83%) and localization (60.14 ± 2.61%). This network also has the largest number of parameters (6.14M). Moreover, ASIN has a complex post-processing step that involves mean-shift clustering, while our graph-based methods only need simple post-processing.

Feature-level evaluation is more important than face-level evaluation for machining feature recognition, as it focuses on the recognized feature instances rather than the individual faces that constitute them. In real-world CAPP applications, we are more interested in identifying the management, geometric, and process attributes of the machining features than the geometric and topological information of each face entity. Therefore, feature-level evaluation can provide more meaningful insights into whether the network accurately recognizes and localizes the machining features. However, feature-level evaluation metrics are non-differentiable and cannot be directly used as optimization objectives for network training. Thus, we must consider face-level evaluation metrics for training, as they can directly assess networks' effectiveness. By combining feature-level and face-level evaluation metrics, we can thoroughly evaluate different networks' machining feature recognition performance.

4.4. Ablation experiments

4.4.1. Ablation study on input features

In this subsection, we investigate the impact of different input features on the performance of our model. We consider different types of input schemes: using both UV grids and attributes of surfaces and curves, without the attributes, and without the UV grids of surfaces and curves. The results on MFInstSeg test dataset is shown in Table 10.

The ablation study is a method to investigate the contribution of different components of a complex input by removing them one by one and measuring the impact on the performance. In this case, we removed three inputs of AAGNet: face attributes, edge attributes, and UV grid. We compared the performance of the full model and the ablated models on three tasks: semantic segmentation, instance segmentation, and bottom face segmentation. And we used four metrics: accuracy, mean intersection over union (mIOU), F-1 score, and accuracy. Table 10 shows that removing any of the inputs leads to a decrease in performance on all tasks and metrics. The most significant drop is observed when removing the UV grid feature, which indicates that it is the most important feature for MFInstSeg. The least significant drop is observed when removing the face attributes, suggesting that the face attributes is less important for distinguishing different machining feature instance. Besides, the impact of removing UV grid on bottom face segmentation is particularly prominent, which can be explained by the fact that the normal vector of grid points has a significant influence on bottom face recognition.

4.4.2. Ablation study on network structure

In this section, we examine how different network structures affect the performance of our model. We consider two aspects of the network structure: the encoder depth and the encoder width. The encoder depth is the number of layers in the graph encoder. The encoder width is the dimension of the latent vector that represents each node. We vary these parameters and evaluate our model on MFInstSeg test dataset.

Encoder depth. Table 11 shows the effect of different number of graph encoder layers on three segmentation tasks. The main observation from the table is that increasing the depth of the graph encoder improves the accuracy and mIOU scores for all three tasks, but the improvement is marginal and diminishes after four layers. The F-1 score for instance segmentation also increases with depth, but reaches the highest value at five layers. A possible implication of the table is that using more than four layers for graph encoder may not be worth the computational cost, as it does not significantly improve the performance on any of the tasks. However, this may depend on the complexity and size of the input graphs and the desired level of accuracy.

Table 10
Ablation study with input features of AAGNet on the MFInstSeg. The bold number is the best result in each column. The values in parentheses indicate the changes relative to using all inputs.

Input	Semantic segmentation		Instance segmentation		Bottom face segmentation	
	Accuracy (%)	mIOU (%)	Accuracy (%)	F-1 (%)	Accuracy (%)	mIOU (%)
Full	**99.15 ± 0.03**	**98.45 ± 0.04**	**99.94 ± 0.01**	**98.84 ± 0.07**	**99.75 ± 0.01**	**98.47 ± 0.05**
No Face Attr.	99.07 ± 0.02 (−0.08)	98.30 ± 0.02 (−0.15)	99.92 ± 0.01 (−0.02)	98.59 ± 0.03 (−0.25)	99.72 ± 0.01 (−0.03)	98.28 ± 0.04 (−0.19)
No Edge Attr.	98.75 ± 0.01 (−0.40)	97.69 ± 0.04 (−0.76)	99.90 ± 0.01 (−0.04)	98.09 ± 0.01 (−0.75)	99.62 ± 0.01 (−0.13)	97.69 ± 0.06 (−0.78)
No UV Grid	96.90 ± 0.06 (−2.25)	94.19 ± 0.09 (−4.26)	99.87 ± 0.01 (−0.07)	97.57 ± 0.02 (−1.27)	97.23 ± 0.01 (−2.52)	83.53 ± 0.05 (−14.94)

Table 11
Effect of different number of graph encoder layers. The bold number is the best result in each column.

Depth	Semantic segmentation		Instance segmentation		Bottom face segmentation	
	Accuracy (%)	mIOU (%)	Accuracy (%)	F-1 (%)	Accuracy (%)	mIOU (%)
2	98.99 ± 0.01	98.18 ± 0.02	99.92 ± 0.01	98.45 ± 0.03	99.67 ± 0.01	97.95 ± 0.06
3	99.15 ± 0.03	98.45 ± 0.04	99.94 ± 0.01	98.84 ± 0.07	99.75 ± 0.01	98.47 ± 0.05
4	**99.21 ± 0.02**	**98.54 ± 0.04**	99.94 ± 0.01	98.93 ± 0.02	**99.76 ± 0.01**	**98.54 ± 0.04**
5	99.15 ± 0.01	98.43 ± 0.02	**99.95 ± 0.01**	**99.03 ± 0.02**	99.76 ± 0.01	98.51 ± 0.05

Table 12
Effect of different dimension of graph encoder. The bold number is the best result in each column.

Width	Semantic segmentation		Instance segmentation		Bottom face segmentation	
	Accuracy (%)	mIOU (%)	Accuracy (%)	F1(%)	Accuracy (%)	mIOU (%)
32	97.81 ± 0.04	96.11 ± 0.06	99.85 ± 0.01	97.41 ± 0.01	99.22 ± 0.01	95.29 ± 0.08
64	99.15 ± 0.03	98.45 ± 0.04	99.94 ± 0.01	98.84 ± 0.07	99.75 ± 0.01	98.47 ± 0.05
128	**99.18 ± 0.02**	**98.51 ± 0.02**	**99.95 ± 0.01**	**99.09 ± 0.02**	**99.78 ± 0.01**	**98.62 ± 0.02**

Encoder width. Table 12 shows the effect of different dimensions of the graph encoder on the three tasks: semantic segmentation, instance segmentation, and bottom face segmentation. We varied the width of the graph encoder from 32 to 128 and evaluated the performance metrics: accuracy, mIOU, and F-1 score. We can see that increasing the width of the graph encoder leads to slightly better results on all tasks and metrics. However, the improvement is marginal and comes at the cost of higher computational complexity and lower inference speed. Therefore, we chose a width of 64 as a trade-off between accuracy and efficiency for our graph encoder.

4.5. Comparative case study and discussion

Existing graph-based feature recognition approaches, such as UV-Net and hierarchical CADNet, are limited to semantic segmentation that recognizes distinct categories of faces within a CAD model. And, they cannot separate individual machining feature instances, which requires detecting the faces of each instance and assigning a unique label to each instance. Consequently, the results obtained by these methods need further processing (e.g., clustering) to extract machining feature instances before the subsequent process planning. To address this limitation, we proposed the AAGNet, a novel approach that can simultaneously identify and locate machining feature instances.

Some existing methods, such as SsdNet and ASIN, can also achieve or approximate machining feature instance segmentation. Therefore, we compare the proposed AAGNet with them through several cases to highlight its advantages.

Shi et al. [10,46,47] proposed a series of multiple-view-based approaches, namely MSVNet, SsdNet, and RefineDet. MSVNet first employs a traditional unsupervised segmentation algorithm to segment machining features from the intersecting models represented by multiple sectional views. Then, MsvNet classifies the segmented features. MSVNet is a time-consuming two-stage approach and its performance largely hinges on the unsupervised segmentation algorithm in the first stage, which usually underperforms. Both SsdNet and RefineDet are single-stage feature recognition methods inspired by 3D target detection. They locate and classify targets simultaneously and output a set of 3D bounding box coordinates and classification results. Although these methods address the time-consuming problem of the two-stage algorithm, the output 3D bounding box coordinates are not appropriate for subsequent process planning, as the machined faces corresponding to the machining feature are needed. Moreover, the multiple views of these methods are derived from voxel, which implies that the 3D model needs to be converted to the voxel representation first.

Zhang et al. [41] introduced a point-cloud-based machining feature recognition approach called Associatively Segmenting and Identifying Network (ASIN). ASIN first samples each surface to obtain a point cloud, which is then fed into PointNet for feature extraction. The feature vectors are then fed into a multi-task head to perform the following tasks: grouping the faces with high similarities into unidentified machining features, predicting the semantic class for each face to determine the category of each machining feature, and recognizing the bottom faces of the machining features. The process of AAGNet is analogous to ASIN, but ASIN is based on point cloud and does not consider the topological relationships between B-Rep entities, which can result in inefficient learning. Moreover, ASIN requires padding the input model with zeros to a fixed number of faces (64 by default) for batch training, which restricts its capability to handle models with more than 64 faces. This limitation can be solved by adjusting the hyperparameter before training, but this would entail retraining the network for different input sizes, which reduces the flexibility and scalability of ASIN for variable inputs.

The comparative case study with MsvNet, SsdNet, and ASIN is shown in Fig. 5 and recognition time in Table 13. The results of MsvNet, SsdNet, and ASIN are obtained from [41] and more details are given in Appendix E. Note that as [41] did not provide STEP files for part A and part B, our test samples were redrawn from the shapes in the original paper. As can be seen from the visualized results in Fig. 5, MsvNet fails to detect the rectangular pocket feature in Part A, and the rectangular through slots in Part B. For both simple samples A and B, SsdNet, ASIN, and the proposed AAGNet successfully identify all the machining features from intersecting parts. For the complex intersecting part C with 11 machining features, only the proposed AAGNet can handle this case and correctly recognize the feature instances. MsvNet and SsdNet are based on voxel, the 3D model has to be voxelized into a grid of discrete cubic cells before being fed into the network. In the voxel representation, the accurate representation of complex 3D models with surfaces and curves requires a high discrete resolution, which is

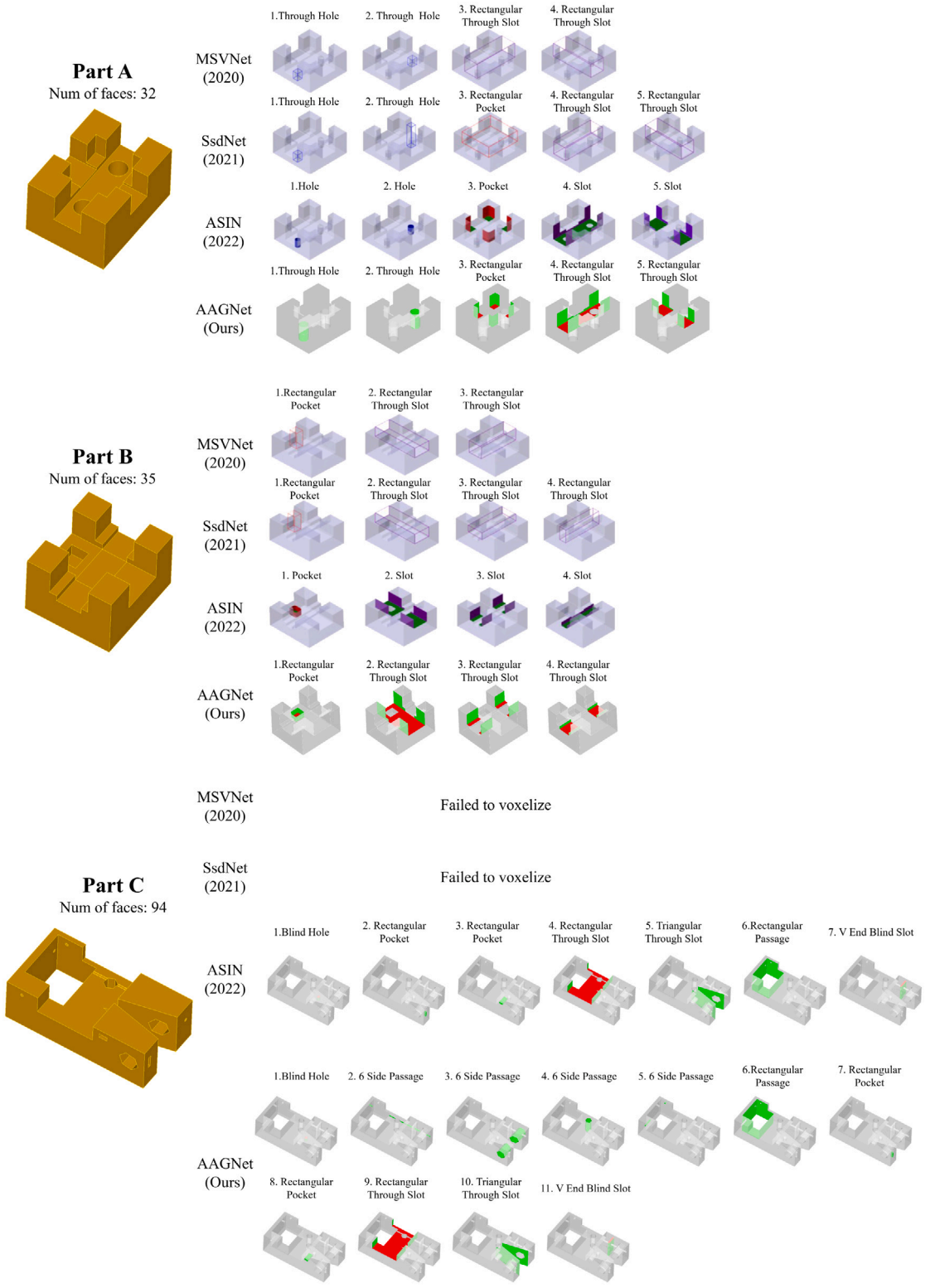

Fig. 5. The comparative case study with MsvNet, SsdNet, ASIN, and AAGNet.

constrained by the capacity of the computer's memory. Therefore at the commonly used voxelization resolution of 64^3, most of the detailed geometric information of the part C has been lost and cannot be used for machining feature recognition (see Fig. 6), which is one of the drawbacks of the voxel representation. ASIN successfully recognized one blind hole, two rectangular pockets, one rectangular through slot, one triangular through slot, one rectangular passage, and a V End Blind Slot in part C, but failed to recognize four six-side passages, possibly due to the larger number of faces involved in these features. In the dataset mentioned in [9], the feature categories are more coarse-grained, such as slot, hole, pocket, etc., but our MFInstSeg dataset contains more fine-grained feature categories, which exhibit high inter-class diversity and low intra-class diversity, making the classification more challenging. This demonstrates the advantages of our proposed AAGNet, which

Table 13
The recognition time of different networks.

		Data conversion (s)	Pre-processing (s)	Feedforward (s)	Post-processing (s)	Total (s)
Part A	MsvNet	0.66	0.82	0.01	0.15	1.64
	SsdNet	0.66	0.52	0.01	0.11	1.30
	ASIN	0.05	0.00	0.05	0.31	0.41
	AAGNet	0.16	0.00	0.02	0.01	**0.19**
Part B	MsvNet	0.59	0.71	0.01	0.11	1.42
	SsdNet	0.59	0.45	0.01	0.10	1.15
	ASIN	0.05	0.00	0.05	0.29	0.39
	AAGNet	0.17	0.00	0.02	0.01	**0.20**
Part C	ASIN	0.15	0.00	0.05	0.12	**0.22**
	AAGNet	0.61	0.00	0.05	0.02	0.68

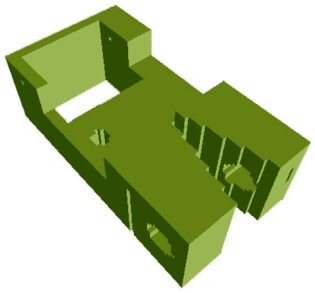

Fig. 6. The voxelized part C using a resolution of 64^3.

can handle fine-grained and intersecting features with variable face numbers and topology by using a graph-based representation and a multi-task learning framework.

We measured and compared the detailed recognition time of four different networks: MsvNet, SsdNet, ASIN, and AAGNet, on sample Part A, B, and C. The experiments were performed on a Windows 10 64 bit PC with Ryzen R9 7950X CPU, 128 GB memory, and NVIDIA GeForce RTX 4090 GPU. The recognition time consists of four components: data conversion, pre-processing, feedforward, and post-processing. Table 13 shows the breakdown of the recognition time for each network and each part.

From Table 13, we can observe that: (1) All the networks perform feedforward on GPU, so the feedforward time is relatively small and stable across different parts. (2) For MsvNet and SsdNet, they use voxel as input, so their data conversion time is the time spent on converting STL to voxel using Binvox [91]. This time is affected by the voxel resolution, which we choose as 64^3. We can see that voxelization is the most time-consuming pre-processing step among these four networks. (3) For ASIN, the data conversion is to discretize each face entity using Open CASCADE, and obtain point cloud data with coordinates and normal vectors. This is a simple process, so the data conversion time is very fast for ASIN. (4) For AAGNet, the data conversion involves checking the input STEP file, building the face adjacency graph, extracting the extended attributes for each face and edge, and sampling grids on the UV coordinate of each face and edge. This process is more complex than point cloud sampling, so the data conversion time is slower than point cloud sampling. (5) For MsvNet, the pre-processing includes slicing the input voxel into multiple 2D images, and applying an unsupervised selective search algorithm to segment them. This is the most time-consuming pre-processing step among all the networks. (6) For SsdNet, the pre-processing only involves slicing the input voxel into multiple 2D images, so the pre-processing time is slightly faster than MsvNet. (7) For ASIN and AAGNet, there is no pre-processing step, as all the data processing work is done in the data conversion step. (8) MsvNet, SsdNet, and ASIN have fixed-length input, so their feedforward time does not increase with the model complexity. However, our AAGNet can accept variable-length graph input, so we can observe that its feedforward time is longer for more complex Part C than for simpler Part A. (9) MsvNet and SsdNet share a similar 3D soft non-maximum suppression for post-processing, so their post-processing time is not very fast. (10) ASIN's post-processing involves mean-shift clustering, which is more complicated, so its post-processing time is slightly longer. (11) AAGNet's post-processing only requires a simple loop to traverse the model output, so its post-processing is very fast.

We also can observe that the proposed AAGNet has a slight advantage in terms of recognition speed over MsvNet and SsdNet. It also outperforms ASIN in most cases, except for Part C where ASIN is faster than AAGNet. However, this is because ASIN failed to recognize four six-side passages in Part C, which reduced the number of proposals to be processed in post-processing. If we consider the recognition accuracy as well as speed, AAGNet is still superior to ASIN.

5. Conclusion and future work

In this paper, we proposed a novel graph representation for machining feature recognition, called geometric Attributed Adjacency Graph (gAAG), which integrates the topology of face adjacency graph, the geometry of UV grid, and the extended attributes of faces and edges from neutral B-Rep. We also introduced machining feature instance segmentation (MFInstSeg) dataset, a large-scale and diverse dataset containing over 60,000 STEP files with 24 machining feature classes and their annotations. To exploit the rich information of gAAG for feature recognition, we developed AAGNet, a novel graph neural network that performs multi-task learning of semantic segmentation, instance segmentation, and bottom face segmentation on gAAG. Our approach was evaluated on various open-source datasets, including MFCAD, MFCAD++, and MFInstSeg, and demonstrated its superior accuracy and efficiency over existing methods. Moreover, we implemented a prototype of feature recognition software based on our approach for subsequent process planning (see Appendix F). Our work shows the potential of applying AAGNet in CAPP systems for machining feature recognition.

However, some limitations and challenges still exist and need to be addressed in future work. Some possible research directions are as follows:

- The proposed AAGNet does not consider process attributes, such as roughness, tolerance, and machining accuracy, due to the limitations of synthetic dataset. These attributes are important for subsequent process planning and should be incorporated in the future study.
- Most of the current open-source machining feature recognition datasets are synthetic, which may not reflect the real-world scenarios. We are working on building a large open-source real CAD part dataset with machining feature instance labels to evaluate the generalization ability of our approach.
- The graph encoders we currently use are general-purpose graph networks, which may not be optimal for B-Rep data structure and machining feature instance segmentation task. Future work will focus on designing more advanced graph network architectures that are tailored for machining feature recognition.

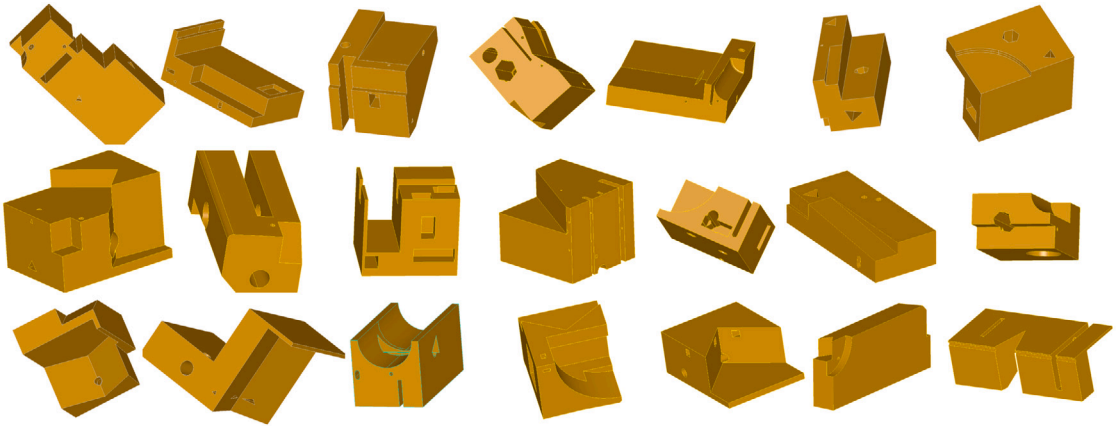

Fig. 7. Some generated examples from MFInstSeg.

Therefore, we will continue to improve our framework and explore new directions in machining feature recognition.

The MFInstSeg is available at https://aistudio.baidu.com/aistudio/datasetdetail/211864?lang=en.

The AAGNet is available at https://github.com/whjdark/AAGNet.

CRediT authorship contribution statement

Hongjin Wu: Conceptualization, Methodology, Formal analysis, Data creation, Software, Visualization, Writing – original draft. **Ruoshan Lei:** Resources, Data analysis. **Yibing Peng:** Validation, Supervision, Resources, Funding acquisition, Writing – review & editing. **Liang Gao:** Validation, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

We have shared the link to my data at the appendix

Acknowledgments

The computation was completed in the HPC Platform of Huazhong University of Science and Technology.

Funding

This work was supported by the National Key Research and Development Program of China (No. 2022YFB3304100)

Appendix A. MFInstSeg examples

Fig. 7 presents the visual overview of the MFInstSeg dataset. Fig. 8 displays the 24 machining feature classes of the MFInstSeg dataset. The MFInstSeg consists of 62,495 randomly created CAD models using Open CASCADE's python interface stored in a STEP file, and also includes machining feature instance labels for each CAD model stored in a JSON file. The labels include semantic segmentation labels which are expressed in python dictionary form as follows: $\{F_i : C_{F_i}\}$ where the F_i indicates the index of the face entity i in the B-Rep and the C_{F_i} indicates the class index of the face entity F_i. They also include instance segmentation labels, which are represented as dense adjacency matrices, as shown in Eq. (1). And the bottom semantic segmentation labels, which are also expressed in python dictionary form as follows: $\{F_i : 0 \text{ or } 1\}$ if the F_i is a bottom face, the label is set to 1.

Appendix B. Implementation details on MFCAD and MFCAD++ dataset

We adopted the results of PointNet, PointNet++, DGCNN, GNN, MeshCNN, UV-Net, and Hierarchical CADNet from [13,14]. We provide the implementation details of each model below:

- PointNet implementation: https://github.com/fxia22/pointnet.pytorch.
- PointNet++ implementation: https://github.com/rusty1s/pytorch_geometric/blob/master/examples/pointnet2_segmentation.py.
- DGCNN implementation: https://github.com/AnTao97/dgcnn.pytorch.
- GNN implementation: https://gitlab.com/qub_femg/machine-learning/cadnet.
- MeshCNN implementation: https://github.com/ranahanocka/MeshCNN.
- UV-Net implementation: https://github.com/AutodeskAILab/UV-Net.
- Hierarchical CADNet implementation: https://gitlab.com/qub_femg/machine-learning/hierarchical-cadnet.

Appendix C. Available datasets

- MFCAD Original: https://github.com/hducg/MFCAD. We also provide the derived gAAG in a JSON format file with UV-grids and attributes stored as node and edge features, which is available at https://aistudio.baidu.com/aistudio/datasetdetail/207373?lang=en.
- MFCAD++ Original: https://gitlab.com/qub_femg/machine-learning/mfcad2-dataset. With derived gAAG: https://aistudio.baidu.com/aistudio/datasetdetail/207819?lang=en. Note that since some STEP models in MFCAD++ have some topological errors, we performed data cleaning to obtain a better quality training set. Therefore, the number of our training, validation and testing samples is slightly smaller than the original MFCAD++, but the impact on the experimental results is negligible.
- MFInstSeg: https://aistudio.baidu.com/aistudio/datasetdetail/211864?lang=en.

Appendix D. Implementation details on MFInstSeg dataset

We conducted all experiments on a cloud server equipped with 12x Intel Xeon Gold 6230R CPU @ 2.10 GHz, 32G RAM, and NVIDIA V100S-32G. We implemented our models using PyTorch v1.13 and the Deep Graph Library (DGL) v0.93. The training hyperparameters were as follows: The batch size was 256, and the learning rate followed a

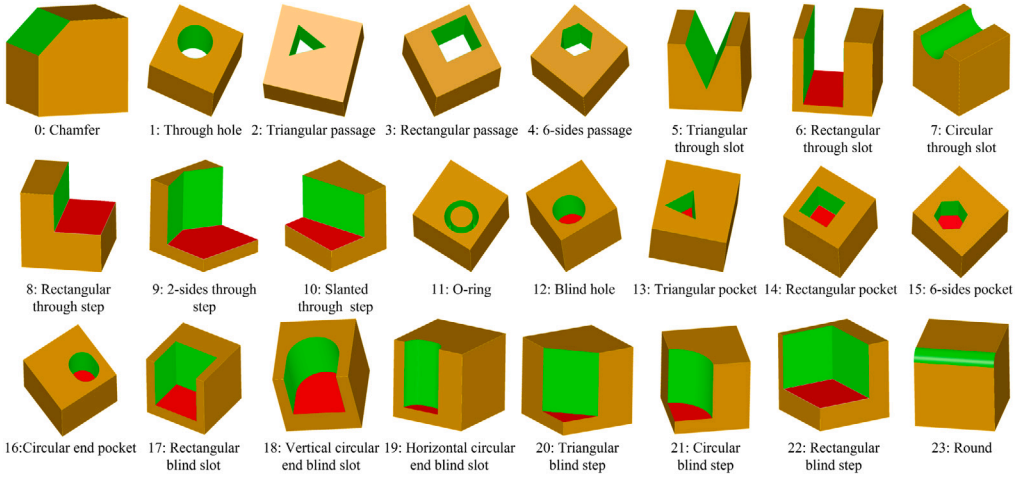

Fig. 8. Machining features used in the MFInstSeg dataset. The green faces are the non-bottom feature faces, red faces are the bottom feature faces, and brown faces are the stock faces.

cosine decay scheduler with an initial value of 0.01. The optimizer was AdamW with a weight decay of 0.01. We also applied Exponential Moving Average (EMA) to update the network parameters by combining the current parameters with the previous EMA values. The EMA decay was set to 0.998. For the semantic segmentation task, we discarded the instance segmentation head and the bottom face segmentation head from the model. We partitioned the MFInstSeg dataset into 70/15/15% for training/validation/testing. The detailed settings of the different models are given below:

- We implemented the Graph Convolutional Network (GCN) model based on the code available at https://github.com/dmlc/dgl/tree/master/examples/pytorch/gcn. We used a four-layer GCN architecture with 128-dimensional hidden embeddings. Each GCN layer consisted of a graph convolution operation for propagating the node features through the graph and a rectified linear unit (ReLU) for applying the nonlinearity.
- We implemented the DeeperGCN model based on the code available at https://github.com/dmlc/dgl/tree/master/examples/pytorch/deepergcn. The model consists of 7 layers of Generalized Message Aggregators (GENs) with hidden node and edge dimensions of 256 and 512, respectively. We applied a batch normalization before each layer and rectified linear unit (ReLU) activation. We also used dropout with a rate of 0.2 to prevent overfitting.
- We implemented the GraphSAGE model following the code available at https://github.com/dmlc/dgl/tree/master/examples/pytorch/graphsage. We used a four-layer GraphSAGE architecture with 128-dimensional hidden embeddings. Each GraphSAGE layer consisted of a max-neighbor-pooling aggregator for aggregating the node features from the neighborhood and a rectified linear unit (ReLU) for applying the nonlinearity.
- We implemented the Graph Isomorphism Network (GIN) model based on the code available at https://github.com/dmlc/dgl/tree/master/examples/pytorch/gin. We used a four-layer GIN architecture with 128-dimensional hidden embeddings. Each GIN layer consisted of a two-layer multilayer perceptron (MLP) for updating the node features and a batch normalization layer for regularization. We used the max-neighbor-pooling as the aggregation function and the rectified linear unit (ReLU) as the activation function for each layer.
- We implemented the Graph Attention Network (GAT) model following the code available at https://github.com/dmlc/dgl/tree/master/examples/pytorch/gat. We used three GAT layers with 4, 4, and 6 attention heads, respectively. The hidden dimension of each layer was set to 64. We applied the exponential linear unit (ELU) as the activation function for each layer.
- We implemented the GATv2 model following the code available at https://github.com/dmlc/dgl/tree/master/examples/pytorch/gatv2. We used a three-layer GATv2 architecture with 8 attention heads for each layer. Each GATv2 layer consisted of a multi-head attention mechanism for updating the node features and an exponential linear unit (ELU) for applying the nonlinearity. The hidden dimension of each layer was set to 64. We applied dropout with a rate of 0.25 to both the node features and the attention weights. We also used residual connections to enhance the information flow between layers.
- We implemented the AAGNet model and opened the available implementation at https://github.com/whjdark/AAGNet. Our AAGNet model consisted of three blocks, each containing a message passing neural network (MPNN) layer with Principal Neighborhood Aggregation (PNAConv) and a multilayer perceptron (MLP) with an expansion rate of two. The MPNN layer employed both sum and max aggregators to combine the node and edge features. The hidden dimensions of the node and edge features were 64 for each block. We applied dropout with a probability of 0.25 to the MLPs and droppath with a probability of 0.25 to the connections between blocks. We also incorporated residual connections to facilitate the information flow across layers. The nonlinear activation function of AAGNet was Mish. The grid size of the surface and curve representations was 5×5.
- We reproduced the Associatively Segmenting and Identifying Network (ASIN) model based on the publicly available code at https://github.com/HARRIXJANG/ASIN-master. To adapt it to the MFInstSeg dataset and the face-level and feature-level evaluation metrics, we modified the code accordingly. We also increased the hyper-parameter of the number of input faces from 64 to 128 to handle larger B-Rep models. For a fair comparison, we trained the model for 120 epochs, which is 20 more epochs than other networks, as ASIN needed more training time to converge. We observed that the training process of ASIN was unstable, and the instance segmentation head failed to learn effectively, so we tuned the loss weight ratio from 1:1:3 to 1:1:10 for semantic segmentation, bottom face segmentation, and instance segmentation to mitigate this issue.

Appendix E. Network details on test samples

We adopted the results of MsvNet, SsdNett, and ASIN from [9]. We provide the details of each network below:

- MsvNet: https://github.com/PeizhiShi/MsvNet.
- SsdNet: https://github.com/PeizhiShi/SsdNet.
- ASIN: https://github.com/HARRIXJANG/ASIN-master.

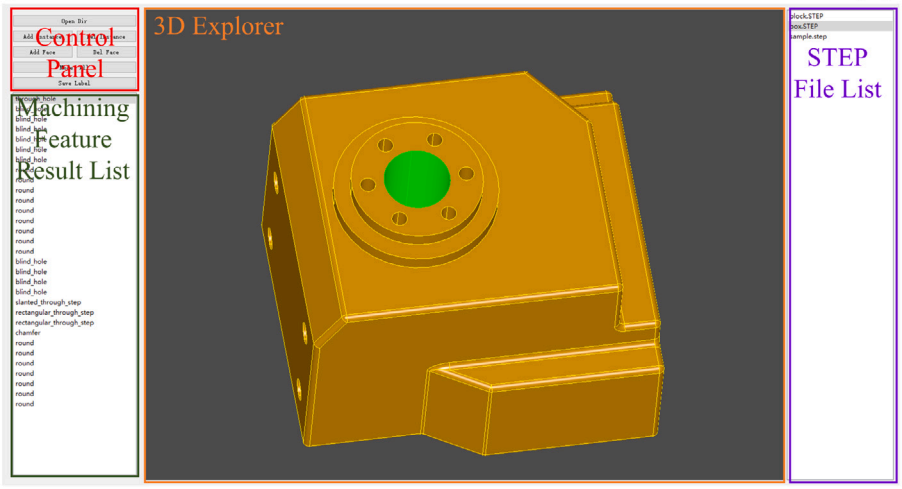

Fig. 9. The prototype software for machining feature recognition.

Appendix F. Prototype software

We developed a prototype software for process planning using PyQt 5.15.7 and PyTorch 1.13. The software is still in an early stage of development and requires further testing and improvement. The user interface (UI) of the prototype software consists of several components, such as a control panel, a result list, a file list, and a 3D explorer. The 3D explorer displays the input 3D part and the faces of the selected feature instance. The UI of the prototype software is illustrated in Fig. 9.

References

[1] Y. Shi, Y. Zhang, K. Xia, R. Harik, A critical review of feature recognition techniques, Comput.-Aided Des. Appl. 17 (5) (2020) 861–899, http://dx.doi.org/10.14733/cadaps.2020.861-899.

[2] T. Xu, J. Li, Z. Chen, Automatic machining feature recognition based on MBD and process semantics, Comput. Ind. 142 (2022) 103736, http://dx.doi.org/10.1016/j.compind.2022.103736, URL: https://www.sciencedirect.com/science/article/pii/S0166361522001336.

[3] Autodesk, Shapemanager, 2001, https://en.wikipedia.org/wiki/ShapeManager.

[4] M.R. Henderson, D.C. Anderson, Computer recognition and extraction of form features: a CAD/CAM link, Comput. Ind. 5 (4) (1984) 329–339, http://dx.doi.org/10.1016/0166-3615(84)90056-3.

[5] S. Joshi, T. Chang, Graph-based heuristics for recognition of machined features from a 3D solid model, Comput. Aided Des. 20 (2) (1988) 58–66, http://dx.doi.org/10.1016/0010-4485(88)90050-4.

[6] Y.S. Kim, Volumetric feature recognition using convex decomposition, in: Manufacturing Research and Technology, Vol. 20, Elsevier, 1994, pp. 39–63, http://dx.doi.org/10.1016/B978-0-444-81600-9.50008-0.

[7] J.H. Vandenbrande, A.A. Requicha, Spatial reasoning for the automatic recognition of machinable features in solid models, IEEE Trans. Pattern Anal. Mach. Intell. 15 (12) (1993) 1269–1285, http://dx.doi.org/10.1109/34.250845.

[8] S. Gao, J. Shah, Automatic recognition of interacting machining features based on minimal condition subgraph, Comput. Aided Des. 30 (9) (1998) 727–739, http://dx.doi.org/10.1016/S0010-4485(98)00033-5.

[9] H. Zhang, S. Zhang, Y. Zhang, J. Liang, Z. Wang, Machining feature recognition based on a novel multi-task deep learning network, Robot. Comput.-Integr. Manuf. 77 (2022) 102369, http://dx.doi.org/10.1016/j.rcim.2022.102369.

[10] P. Shi, Q. Qi, Y. Qin, P.J. Scott, X. Jiang, Intersecting machining feature localization and recognition via single shot multibox detector, IEEE Trans. Ind. Inform. 17 (5) (2021) 3292–3302, http://dx.doi.org/10.1109/TII.2020.3030620.

[11] Z. Zhang, P. Jaiswal, R. Rai, FeatureNet: Machining feature recognition based on 3D convolution neural network, Comput. Aided Des. 101 (2018) 12–22, http://dx.doi.org/10.1016/j.cad.2018.03.006.

[12] B.R. Babić, N. Nešić, Z. Miljković, Automatic feature recognition using artificial neural networks to integrate design and manufacturing: Review of automatic feature recognition systems, AI EDAM 25 (3) (2011) 289–304, http://dx.doi.org/10.1017/S0890060410000545.

[13] A.R. Colligan, T.T. Robinson, D.C. Nolan, Y. Hua, W. Cao, Hierarchical CADNet: Learning from B-reps for machining feature recognition, Comput. Aided Des. 147 (2022) 103226, http://dx.doi.org/10.1016/j.cad.2022.103226, URL: https://www.sciencedirect.com/science/article/pii/S0010448522000240.

[14] P.K. Jayaraman, A. Sanghi, J.G. Lambourne, K.D. Willis, T. Davies, H. Shayani, N. Morris, UV-net: Learning from boundary representations, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 11703–11712, http://dx.doi.org/10.1109/CVPR46437.2021.01153.

[15] J.G. Lambourne, K.D. Willis, P.K. Jayaraman, A. Sanghi, P. Meltzer, H. Shayani, BRepNet: A topological message passing system for solid models, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 12773–12782, http://dx.doi.org/10.1109/CVPR46437.2021.01258.

[16] G. Corso, L. Cavalleri, D. Beaini, P. Liò, P. Velickovic, Principal neighbourhood aggregation for graph nets, 2020, CoRR abs/2004.05718 arXiv:2004.05718. URL: https://arxiv.org/abs/2004.05718.

[17] I.A. Donaldson, J.R. Corney, Rule-based feature recognition for 2.5D machined components, Int. J. Comput. Integr. Manuf. 6 (1–2) (1993) 51–64, http://dx.doi.org/10.1080/09511929308944555, arXiv:https://doi.org/10.1080/09511929308944555.

[18] M. Al-wswasi, A. Ivanov, A novel and smart interactive feature recognition system for rotational parts using a STEP file, Int. J. Adv. Manuf. Technol. 104 (1–4) (2019) 261–284, http://dx.doi.org/10.1007/s00170-019-03849-1, URL: http://link.springer.com/10.1007/s00170-019-03849-1.

[19] Y. Zhang, F. Wang, T. Chen, User-defined machining feature recognition based on semantic reasoning for B-rep models, Comput.-Aided Des. Appl. 20 (4) (2023) 763–785, http://dx.doi.org/10.14733/cadaps.2023.763-785.

[20] Y. Li, Y. Ding, W. Mou, H. Guo, Feature recognition technology for aircraft structural parts based on a holistic attribute adjacency graph, Proc. Inst. Mech. Eng. B 224 (2) (2010) 271–278, http://dx.doi.org/10.1243/09544054JEM1634.

[21] Y. Shi, Y. Zhang, R. Harik, Manufacturing feature recognition with a 2D convolutional neural network, CIRP J. Manuf. Sci. Technol. 30 (2020) 36–57, http://dx.doi.org/10.1016/j.cirpj.2020.04.001, URL: https://www.sciencedirect.com/science/article/pii/S1755581720300298.

[22] F. Ning, Y. Shi, M. Cai, W. Xu, Part machining feature recognition based on a deep learning method, J. Intell. Manuf. (2021) 1–13, http://dx.doi.org/10.1007/s10845-021-01827-7.

[23] H. Sakurai, Volume decomposition and feature recognition: part 1—polyhedral objects, Comput. Aided Des. 27 (11) (1995) 833–843, http://dx.doi.org/10.1016/0010-4485(95)00007-0.

[24] D.S. Nau, S.K. Gupta, T.R. Kramer, W.C. Regli, G. Zhang, Development of machining alternatives, based on MRSEVs, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Vol. 97645, American Society of Mechanical Engineers, 1993, pp. 47–57, http://dx.doi.org/10.1115/CIE1993-0007.

[25] J. Ferreira, S. Hinduja, Convex hull-based feature-recognition method for 2.5D components, Comput. Aided Des. 22 (1) (1990) 41–49, http://dx.doi.org/10.1016/0010-4485(90)90028-B.

[26] Y. Woo, H. Sakurai, Recognition of maximal features by volume decomposition, Comput. Aided Des. 34 (3) (2002) 195–207, http://dx.doi.org/10.1016/S0010-4485(01)00080-X.

[27] Y. Woo, Fast cell-based decomposition and applications to solid modeling, Comput. Aided Des. 35 (11) (2003) 969–977, http://dx.doi.org/10.1016/S0010-4485(02)00144-6.

[28] B. Babic, N. Nesic, Z. Miljkovic, A review of automated feature recognition with rule-based pattern recognition, Comput. Ind. 59 (4) (2008) 321–337, http://dx.doi.org/10.1016/j.compind.2007.09.001.

[29] H. Yan, C. Yan, P. Yan, Y. Hu, S. Liu, Manufacturing feature recognition method based on graph and minimum non-intersection feature volume suppression, Int. J. Adv. Manuf. Technol. (2023) 1–20, http://dx.doi.org/10.1007/s00170-023-11031-x.

[30] J. Han, A.A. Requicha, Integration of feature based design and feature recognition, Comput. Aided Des. 29 (5) (1997) 393–403, http://dx.doi.org/10.1016/S0010-4485(96)00079-6.

[31] H. Li, Y. Huang, Y. Sun, L. Chen, Hint-based generic shape feature recognition from three-dimensional B-rep models, Adv. Mech. Eng. 7 (2015) http://dx.doi.org/10.1177/1687814015582082.

[32] C. Zhang, K. Chan, Y. Chen, A hybrid method for recognizing feature interactions, Integr. Manuf. Syst. (1998) http://dx.doi.org/10.1108/09576069810202078.

[33] K. Rahmani, B. Arezoo, A hybrid hint-based and graph-based framework for recognition of interacting milling features, Comput. Ind. 58 (4) (2007) 304–312, http://dx.doi.org/10.1016/j.compind.2006.07.001, URL: https://www.sciencedirect.com/science/article/pii/S0166361506001059.

[34] A.K. Verma, S. Rajotia, A hybrid machining feature recognition system, Int. J. Manuf. Res. 4 (3) (2009) 343–361, http://dx.doi.org/10.1504/IJMR.2009.026578.

[35] S.R. Subrahmanyam, A method for generation of machining and fixturing features from design features, Comput. Ind. 47 (3) (2002) 269–287, http://dx.doi.org/10.1016/S0166-3615(01)00154-3, URL: https://www.sciencedirect.com/science/article/pii/S0166361501001543.

[36] S. Prabhakar, M.R. Henderson, Automatic form-feature recognition using neural-network-based techniques on boundary representations of solid models, Comput. Aided Des. 24 (7) (1992) 381–393, http://dx.doi.org/10.1016/0010-4485(92)90064-H.

[37] J. Wang, S. Liu, Hopfield neural network-based automatic recognition for 3-D features, in: Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan), Vol. 3, 1993, pp. 2121–2124 vol.3, http://dx.doi.org/10.1109/IJCNN.1993.714143.

[38] K. Lankalapalli, S. Chatterjee, T. Chang, Feature recognition using ART2: a self-organizing neural network, J. Intell. Manuf. 8 (3) (1997) 203–214, http://dx.doi.org/10.1023/A:1018521207901.

[39] N. Öztürk, F. Öztürk, Hybrid neural network and genetic algorithm based machining feature recognition, J. Intell. Manuf. 15 (3) (2004) 287–298, http://dx.doi.org/10.1023/B:JIMS.0000026567.63397.d5.

[40] L. Ding, J. Matthews, A contemporary study into the application of neural network techniques employed to automate CAD/CAM integration for die manufacture, Comput. Ind. Eng. 57 (4) (2009) 1457–1471, http://dx.doi.org/10.1016/j.cie.2009.01.006, URL: https://www.sciencedirect.com/science/article/pii/S0360835209000114.

[41] Y. Zhang, Y. Zhang, K. He, D. Li, X. Xu, Y. Gong, Intelligent feature recognition for STEP-NC-compliant manufacturing based on artificial bee colony algorithm and back propagation neural network, J. Manuf. Syst. 62 (2022) 792–799, http://dx.doi.org/10.1016/j.jmsy.2021.01.018, URL: https://www.sciencedirect.com/science/article/pii/S0278612521000182.

[42] D. Peddireddy, X. Fu, A. Shankar, H. Wang, B.G. Joung, V. Aggarwal, J.W. Sutherland, M.B.-G. Jun, Identifying manufacturability and machining processes using deep 3D convolutional networks, J. Manuf. Process. 64 (2021) 1336–1348, http://dx.doi.org/10.1016/j.jmapro.2021.02.034, URL: https://www.sciencedirect.com/science/article/pii/S1526612521001201.

[43] D. Maturana, S. Scherer, VoxNet: A 3D convolutional neural network for real-time object recognition, in: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015, pp. 922–928, http://dx.doi.org/10.1109/IROS.2015.7353481.

[44] Y. Zhou, O. Tuzel, VoxelNet: End-to-end learning for point cloud based 3D object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 4490–4499, URL: http://arxiv.org/abs/1711.06396.

[45] D. Peddireddy, X. Fu, H. Wang, B.G. Joung, V. Aggarwal, J.W. Sutherland, M. Byung-Guk Jun, Deep learning based approach for identifying conventional machining processes from CAD data, Procedia Manuf. 48 (2020) 915–925, http://dx.doi.org/10.1016/j.promfg.2020.05.130, URL: https://www.sciencedirect.com/science/article/pii/S2351978920315821. 48th SME North American Manufacturing Research Conference, NAMRC 48.

[46] P. Shi, Q. Qi, Y. Qin, P.J. Scott, X. Jiang, A novel learning-based feature recognition method using multiple sectional view representation, J. Intell. Manuf. 31 (5) (2020) 1291–1309, http://dx.doi.org/10.1007/s10845-020-01533-w.

[47] P. Shi, Q. Qi, Y. Qin, P.J. Scott, X. Jiang, Highly interacting machining feature recognition via small sample learning, Robot. Comput.-Integr. Manuf. 73 (2022) 102260, http://dx.doi.org/10.1016/j.rcim.2021.102260.

[48] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, Ssd: Single shot multibox detector, in: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, the Netherlands, October 11–14, 2016, Proceedings, Part I 14, Springer, 2016, pp. 21–37, URL: http://arxiv.org/abs/1512.02325.

[49] A.R. Colligan, T.T. Robinson, D.C. Nolan, Y. Hua, Point cloud dataset creation for machine learning on CAD models, Comput.-Aided Des. Appl. 18 (4) (2021) 760–771, http://dx.doi.org/10.14733/cadaps.2021.760-771.

[50] X. Yao, D. Wang, T. Yu, C. Luan, J. Fu, A machining feature recognition approach based on hierarchical neural network for multi-feature point cloud models, J. Intell. Manuf. (2022) 1–12, http://dx.doi.org/10.1007/s10845-022-01939-8.

[51] C.R. Qi, L. Yi, H. Su, L.J. Guibas, PointNet++: Deep hierarchical feature learning on point sets in a metric space, in: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 30, Curran Associates, Inc., 2017, URL: https://arxiv.org/abs/1706.02413.

[52] Y. He, H. Yu, X. Liu, Z. Yang, W. Sun, Y. Wang, Q. Fu, Y. Zou, A. Mian, Deep learning based 3D segmentation: A survey, 2021, CoRR abs/2103.05423 arXiv:2103.05423. URL: https://arxiv.org/abs/2103.05423.

[53] Y. Ma, Y. Zhang, X. Luo, Automatic recognition of machining features based on point cloud data using convolution neural networks, in: Proceedings of the 2019 International Conference on Artificial Intelligence and Computer Science, in: AICS 2019, Association for Computing Machinery, New York, NY, USA, 2019, pp. 229–235, http://dx.doi.org/10.1145/3349341.3349407.

[54] J.M. Worner, D. Brovkina, O. Riedel, Feature recognition for graph-based assembly product representation using machine learning, in: 2021 21st International Conference on Control, Automation and Systems (ICCAS), 2021, pp. 629–635, http://dx.doi.org/10.23919/ICCAS52745.2021.9649784.

[55] W. Wang, R. Yu, Q. Huang, U. Neumann, SGPN: Similarity group proposal network for 3D point cloud instance segmentation, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Los Alamitos, CA, USA, 2018, pp. 2569–2578, http://dx.doi.org/10.1109/CVPR.2018.00272, URL: https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00272.

[56] X. Wang, S. Liu, X. Shen, C. Shen, J. Jia, Associatively segmenting instances and semantics in point clouds, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Los Alamitos, CA, USA, 2019, pp. 4091–4100, http://dx.doi.org/10.1109/CVPR.2019.00422, URL: https://doi.ieeecomputersociety.org/10.1109/CVPR.2019.00422.

[57] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, D. Cohen-Or, MeshCNN: A network with an edge, ACM Trans. Graph. 38 (4) (2019) http://dx.doi.org/10.1145/3306346.3322959.

[58] Y. Feng, Y. Feng, H. You, X. Zhao, Y. Gao, Meshnet: Mesh neural network for 3d shape representation, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 8279–8286, URL: http://arxiv.org/abs/1811.11424.

[59] M. Naseer, S. Khan, F. Porikli, Indoor scene understanding in 2.5/3D for autonomous agents: A survey, IEEE Access 7 (2019) 1859–1887, http://dx.doi.org/10.1109/ACCESS.2018.2886133.

[60] J.-L. Jia, S.-W. Zhang, Y.-R. Cao, X.-L. Qi, et al., Machining feature recognition method based on improved mesh neural network, Iran. J. Sci. Technol. Trans. Mech. Eng. (2023) 1–14, http://dx.doi.org/10.1007/s40997-023-00610-8.

[61] W. Cao, T. Robinson, Y. Hua, F. Boussuge, A.R. Colligan, W. Pan, Graph representation of 3D CAD models for machining feature recognition with deep learning, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Vol. Volume 11A: 46th Design Automation Conference (DAC), 2020, http://dx.doi.org/10.1115/DETC2020-22355, URL: https://doi.org/10.1115/DETC2020-22355. V11AT11A003.

[62] P. Wang, W.-A. Yang, Y. You, A hybrid learning framework for manufacturing feature recognition using graph neural networks, J. Manuf. Process. 85 (2023) 387–404, http://dx.doi.org/10.1016/j.jmapro.2022.10.075, URL: https://www.sciencedirect.com/science/article/pii/S152661252200768X.

[63] C. Yeo, B.C. Kim, S. Cheon, J. Lee, D. Mun, Machining feature recognition based on deep neural networks to support tight integration with 3D CAD systems, Sci. Rep. 11 (1) (2021) 22147, http://dx.doi.org/10.1038/s41598-021-01313-3.

[64] X. Fu, D. Peddireddy, V. Aggarwal, M.B.-G. Jun, Improved dexel representation: A 3D CNN geometry descriptor for manufacturing CAD, IEEE Trans. Ind. Inform. (2021) 1, http://dx.doi.org/10.1109/TII.2021.3136167.

[65] R. Harik, Y. Shi, S. Baek, Shape terra: mechanical feature recognition based on a persistent heat signature, Comput.-Aided Des. Appl. 14 (2) (2017) 206–218, http://dx.doi.org/10.1080/16864360.2016.1223433, arXiv:https://doi.org/10.1080/16864360.2016.1223433.

[66] C. Jian, Z. Sheng, Y. Lu, M. Lin, M. Zhang, X. Hou, QSCC: A quaternion semantic cell convolution graph neural network for MBD product model classification, IEEE Trans. Ind. Inform. (2023) 1–8, http://dx.doi.org/10.1109/TII.2023.3246066.

[67] V. Miles, S. Giani, O. Vogt, Recursive encoder network for the automatic analysis of STEP files, J. Intell. Manuf. 34 (1) (2023) 181–196, http://dx.doi.org/10.1007/s10845-022-01998-x, URL: https://link.springer.com/10.1007/s10845-022-01998-x.

[68] V. Miles, S. Giani, O. Vogt, Approaching STEP file analysis as a language processing task: A robust and scale-invariant solution for machining feature recognition, J. Comput. Appl. Math. 427 (2023) 115166, http://dx.doi.org/10.1016/j.cam.2023.115166, URL: https://www.sciencedirect.com/science/article/pii/S0377042723001103.

[69] M. Wang, L. Yu, D. Zheng, Q. Gan, Y. Gai, Z. Ye, M. Li, J. Zhou, Q. Huang, C. Ma, Z. Huang, Q. Guo, H. Zhang, H. Lin, J. Zhao, J. Li, A.J. Smola, Z. Zhang, Deep graph library: Towards efficient and scalable deep learning on graphs, 2019, CoRR abs/1909.01315 arXiv:1909.01315. URL: http://arxiv.org/abs/1909.01315.

[70] S. Kim, H.-g. Chi, X. Hu, Q. Huang, K. Ramani, A large-scale annotated mechanical components benchmark for classification and retrieval tasks with deep neural networks, in: A. Vedaldi, H. Bischof, T. Brox, J.-M. Frahm (Eds.), Computer Vision – ECCV 2020, in: Lecture Notes in Computer Science, vol. 12363, Springer International Publishing, Cham, 2020, pp. 175–191, http://dx.doi.org/10.1007/978-3-030-58523-5_11, URL: https://link.springer.com/10.1007/978-3-030-58523-5_11.

[71] S. Koch, A. Matveev, Z. Jiang, F. Williams, A. Artemov, E. Burnaev, M. Alexa, D. Zorin, D. Panozzo, ABC: A big CAD model dataset for geometric deep learning, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 9593–9603, http://dx.doi.org/10.1109/CVPR.2019.00983.

[72] A. Angrish, B. Craver, B. Starly, "FabSearch": A 3D CAD Model-Based Search Engine for Sourcing Manufacturing Services, J. Comput. Inf. Sci. Eng. 19 (4) (2019) 041006, http://dx.doi.org/10.1115/1.4043211.

[73] R. Lei, H. Wu, Y. Peng, MfPointNet: A point cloud-based neural network using selective downsampling layer for machining feature recognition, Machines 10 (12) (2022) http://dx.doi.org/10.3390/machines10121165, URL: https://www.mdpi.com/2075-1702/10/12/1165.

[74] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P.S. Yu, A comprehensive survey on graph neural networks, 2019, CoRR abs/1901.00596 arXiv:1901.00596. URL: http://arxiv.org/abs/1901.00596.

[75] L. Wu, P. Cui, J. Pei, L. Zhao, Graph Neural Networks: Foundations, Frontiers, and Applications, Springer Singapore, Singapore, 2022, p. 725.

[76] W. Yu, C. Si, P. Zhou, M. Luo, Y. Zhou, J. Feng, S. Yan, X. Wang, MetaFormer baselines for vision, 2022, arXiv:2210.13452.

[77] X. Song, R. Ma, J. Li, M. Zhang, D.P. Wipf, Network in graph neural network, 2021, CoRR abs/2111.11638 arXiv:2111.11638. URL: https://arxiv.org/abs/2111.11638.

[78] K. Han, Y. Wang, J. Guo, Y. Tang, E. Wu, Vision GNN: An image is worth graph of nodes, 2022, arXiv:2206.00272.

[79] G. Corso, L. Cavalleri, D. Beaini, P. Liò, P. Velickovic, Principal neighbourhood aggregation for graph nets, 2020, CoRR abs/2004.05718 arXiv:2004.05718. URL: https://arxiv.org/abs/2004.05718.

[80] I. Loshchilov, F. Hutter, Fixing weight decay regularization in adam, 2017, CoRR abs/1711.05101 arXiv:1711.05101. URL: http://arxiv.org/abs/1711.05101.

[81] N.S. Detlefsen, J. Borovec, J. Schock, A.H. Jha, T. Koker, L.D. Liello, D. Stancl, C. Quan, M. Grechkin, W. Falcon, TorchMetrics - measuring reproducibility in pytorch, J. Open Source Softw. 7 (70) (2022) 4101, http://dx.doi.org/10.21105/joss.04101.

[82] D. Misra, Mish: A self regularized non-monotonic activation function, 2020, arXiv:1908.08681.

[83] R.Q. Charles, H. Su, M. Kaichun, L.J. Guibas, PointNet: Deep learning on point sets for 3D classification and segmentation, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 77–85, http://dx.doi.org/10.1109/CVPR.2017.16.

[84] Y. Wang, Y. Sun, Z. Liu, S.E. Sarma, M.M. Bronstein, J.M. Solomon, Dynamic graph CNN for learning on point clouds, ACM Trans. Graph. 38 (5) (2019) http://dx.doi.org/10.1145/3326362.

[85] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, 2016, CoRR abs/1609.02907 arXiv:1609.02907. URL: http://arxiv.org/abs/1609.02907.

[86] G. Li, C. Xiong, A. Thabet, B. Ghanem, DeeperGCN: All you need to train deeper GCNs, 2020, arXiv:2006.07739.

[87] W.L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, 2018, arXiv:1706.02216.

[88] K. Xu, W. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks? 2018, CoRR abs/1810.00826 arXiv:1810.00826. URL: http://arxiv.org/abs/1810.00826.

[89] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, 2018, arXiv:1710.10903.

[90] S. Brody, U. Alon, E. Yahav, How attentive are graph attention networks?, 2022, arXiv:2105.14491.

[91] P. Min, Binvox, 2004–2019, http://www.patrickmin.com/binvox or https://www.google.com/search?q=binvox. Accessed: 2023-08-10.