Computer-Aided Design 147 (2022) 103226

Contents lists available at ScienceDirect

Computer-Aided Design

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

Hierarchical CADNet: Learning from B-Reps for Machining Feature Recognition

Andrew R. Colligan [a], Trevor T. Robinson [a,*], Declan C. Nolan [a], Yang Hua [a], Weijuan Cao [b]

[a] Queen's University Belfast, Belfast, UK
[b] Hangzhou Dianzi University, Hangzhou, China

ARTICLE INFO

Article history:
Received 25 August 2021
Received in revised form 24 January 2022
Accepted 28 January 2022

Keywords:
Machining feature recognition
3D deep learning
Hierarchical graph convolution network
Computer-aided process planning (CAPP)
B-Rep
CAD

ABSTRACT

Deep learning approaches have been shown to be capable of recognizing shape features (e.g. machining features) in Computer-Aided Design (CAD) models in certain circumstances, yet still have issues when the features intersect, and in exploiting the geometric and topological information which comprises the boundary representation (B-Rep) of the typical CAD model. This paper presents a novel hierarchical B-Rep graph shape representation which encodes information about the surface geometry and face topology of the B-Rep. To learn from this new shape representation, a novel hierarchical graph convolutional network called Hierarchical CADNet has been created, which has been shown to outperform other state-of-the-art neural architectures on feature identification, including machining features that intersect, with improvements in accuracy for some more complex CAD models.

1. Introduction

The product development process (PDP) encompasses all the different processes required to develop an idea from being a concept to producing a finished product. Processes in the PDP can include: specification, design, analysis, testing and manufacturing. For the design and mock-up of a product, engineers will predominantly utilize a mechanical Computer-Aided Design (CAD) system (e.g. CATIA V5/6 [1] or Siemens NX [2]). Within the CAD system, there is a suite of tools that enable the creation and modification of the design, contributing to the ability to create models of complex engineering components.

Once a design has been finalized from subsequent analysis and testing steps in the PDP, the CAD model and corresponding engineering drawings are progressed to the manufacturing department. Engineers then use a Computer-Aided Manufacturing (CAM) system to develop code to control the manufacturing tools to produce the physical product [3].

Computer-Aided Process Planning (CAPP) is focused on linking CAD and CAM systems, where it is used to evaluate the manufacturing processes and conditions required to adapt a CAD design into a physical product. One such method of achieving this, is through automatic feature recognition (AFR) [3]. AFR techniques attempt to interpret the CAD model to identify machining/manufacturing features. Within the CAD model, these machining features are a compositions of certain B-Rep geometric and topological entities (e.g. surfaces, curves, points and B-Rep faces, edges, vertices) which coalesce into semantic features relating to the manufacturing of the part. Some common examples of machining features include: holes, pockets and slots [4].

While a CAD model feature tree can help in the identification of features, the task of recognizing these machining features is not trivial, especially where a feature exists as a consequence of multiple features in proximity (rather than being a feature modeled in its own right), or when being identified from neutral file formats such as a STEP file, where the model is stored based on the composition of the B-Rep model rather than based on the features included. Instances when these machining features intersect also pose a difficult challenge in terms of identification [5]. Therefore, machine learning (ML) and more recently deep learning (DL), have been utilized to help solve this problem. Most of the previous DL approaches have not learned from the boundary representation (B-Rep) directly, but have instead converted it into an alternative shape representation (e.g. a voxel or point cloud representation). The success of the approach can be heavily reliant on the resolution used to construct the alternative representation, more details on which are given in the next section. Also, these do not capitalize on the learning of B-Rep geometric and topological information that could be beneficial for machining feature recognition.

Herein, a novel 3D shape representation to learn from B-Rep CAD models is proposed. This representation consists of a hierarchical graph structure in which one level represents the B-Rep face adjacency graph, and the second level is based on a graph representing the facets of a triangular mesh of the CAD model

* Corresponding author.
E-mail address: t.robinson@qub.ac.uk (T.T. Robinson).

https://doi.org/10.1016/j.cad.2022.103226

boundary. The rationale is that the face adjacency graph encodes the topology of the CAD model, while the facet graph encodes the geometry of the surfaces. This offers the ability to learn from both the geometry and topology of CAD models. To learn from this hierarchical graph, a new hierarchical graph convolutional network called Hierarchical CADNet has been created. This graph neural network can learn from both the geometry and topology of the B-Rep through its hierarchical structure. Edge convexity between the B-Rep faces can also be included and learned. Importantly, each sample in the dataset is not constrained to having the same size and shape.

The proposed shape representation and network is applied to the task of machining feature recognition, where a classification is made for each B-Rep face. A dataset creation tool and dataset are also provided to help in the learning of machining features from CAD models with intersecting machining features.

2. Related work

2.1. 3D shape deep learning

Machine learning is the science of getting computers to perform a task without being explicitly programmed what to do. For the ML algorithm to do this, it must be designed to identify patterns and make predictions about observed data. Deep learning is a sub-class of machine learning, where deep neural networks with multiple hidden neural layers are trained on large-scale datasets consisting of thousands to millions of pieces of data.

Many advancements in deep learning have revolved around 2D data such as 2D convolutional neural networks (CNNs) for image recognition [6]. However, recently more emphasis has been given to 3D data, and in particular 3D shapes. Unlike 2D image data which has a regular grid-like structure, 3D shape can be unordered and irregular. This is an issue as previous deep learning methodologies developed for 2D, like CNNs, require each data sample to have the same input size and tensor shape. Therefore, alternative methods are needed for learning on 3D shapes.

Some approaches have utilized the advances of image descriptors and large-scale image databases for 3D data. These multi-view approaches [7,8] take images of the 3D shape from multiple views and then use a 2D CNN to compress the images into a single descriptor and make a prediction. Multi-view methods do not exploit the intrinsic information of a 3D representation and therefore research in the field has sought to learn from 3D shapes directly.

Wu et al. [9] proposed representing the 3D shape as a voxel representation. A voxel grid is analogous to a 2D pixel grid with additional depth information, where a voxel which lies within the bounds of the shape is denoted with a "1" and where it is outside a "0". A 3D CNN can be utilized for this voxel data due to its regular structure, where it differs from 2D CNNs by using a 3D cube for its filter. [10,11] also used voxels to represent shape, and [12] encoded some surface information such as intersected surface area and local curvature into the representation. All these voxel methods have a significant drawback in that as the voxel resolution is increased, the memory and computational expense scale cubically. This means that most voxel methods are limited in resolution, to allow them to fit in the memory of commercially available GPUs. As the resolution of the voxel representation is limited, it can have difficulty representing larger-scale 3D shapes which include finer grain features.

Octrees, like voxels, are spatial occupancy enumerations where a set of spatial cells discretize the 3D space; each with specific occupancy information. Yet, octrees differ as their data is stored in hierarchical tree structures, in which each level uses exponentially smaller cubes. This means sparser regions of the shape can be represented with lower tree levels and therefore fewer cubes compared to denser areas, making the representation more efficient. Riegler et al. [13] and Wang et al. [14,15] both learned on octree structures, with Riegler et al. [13] showing that only the voxels near the boundaries of the shape activated the neurons of the network and therefore most of the information about the shape is described by its surface. Each approach is limited by its ability to approximate curved surfaces.

Point clouds represent a shape as a set of points in space. Within the computer science community, there is a lot of interest in being able to learn from point cloud data, as it is the 3D shape representation produced by 3D scanning technology. Each point in the point cloud can hold coordinate information, as well as other input features such as the surface normal. While the point cloud could be converted into the previous 3D shape representations already discussed, this creates voluminous data and can lead to noise in the data. PointNet [16], PointNet++ [17] and DGCNN [18] instead proposed methods of learning from the point clouds directly. The issue with point cloud approaches is that smaller features can be under-sampled.

Some approaches have attempted to learn from surface meshes such as MeshCNN [19], MeshNet [20] &[21] which operate on the edges and facets of the meshes, respectively.

Lastly, geometric deep learning has been a field within the 3D shape deep learning space, which has looked at learning from heterogeneous data structures such as graphs and manifolds. [22–24] have all looked at mapping individual patches of the manifold to alternate representations that are more responsive to the filters of the CNN. Each patch behaves like a local coordinate system on the manifold surface, giving directionality information about the manifold to the neural network. A major advantage of these geometric approaches is that they do not require a fixed input size across the entire dataset.

2.2. Learning from mechanical CAD models

As previously stated, a requirement of a CAPP system is the ability to recognize machining features. While traditional approaches to this are algorithmic [4], there have been several approaches that have attempted to use machine learning and deep learning to achieve this. Most commercial mechanical CAD systems generally represent 3D shapes as a boundary representation (B-Rep). This is a continuous mathematical structure where the shape is described by a set of surfaces denoting the boundary of the shape. In most modern mechanical CAD systems, the boundaries are modeled using spline-based surfaces. The data structure for the CAD model consists of both geometric and topological entities. Due to the continuous nature, it is difficult to perform learning on the B-Rep shape directly.

Previous approaches have attempted to use an attribute adjacency graph (AAG) [5]. This is a graphical representation of the B-Rep where the graph $G = \{N, A, T\}$, consists of N nodes, A arcs and T attributes. For each node in N, there is a corresponding face. Where this face shares a common edge with another face, it will be defined by a unique arc in A. Each arc in A will have an attribute T encoding the edges convexity. By decomposing the AAG based on where there are concave edges, sub-graphs can be constructed which represent the machining features of the CAD model. These sub-graphs are then converted into semantic vectors called feature representation vectors (FRVs). These FRVs are described by heuristic techniques. Pande et al. [25] and Onwubolu [26] each suggested different FRVs to use for the input to simple feedforward neural networks for classification. These approaches demonstrated low levels of success at identifying intersecting and complex machining features.

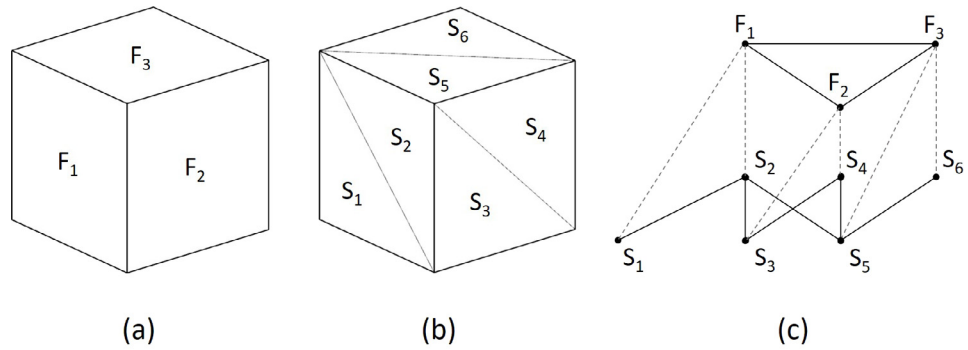


Fig. 1. Hierarchical B-Rep graph structure (a) B-Rep, (b) facet mesh & (c) hierarchical graph.

Zhang et al. [27] proposed the first deep learning method for machining feature recognition, by combining a voxel representation and a 3D CNN. The neural network was trained on a dataset consisting of 24 different machining feature classes. Each CAD model in the dataset only contained one machining feature class. Therefore, for the recognition of multiple machining features, auxiliary segmentation algorithms were required. This approach was limited by those of the other voxel approaches already discussed. Also, the segmentation algorithms tended to struggle with highly intersecting features. Peddireddy et al. [28] used a voxel representation and 3D CNN to identify machining features, but also aimed to recognize machining processes. Ning et al. [29] used the AAG to segment the machining features and a voxel 3D CNN method for classification. Ma et al. [30] utilized a modified version of PointNet to learn from a point cloud version of the CAD model. Again, this method relied on the same auxiliary segmentation algorithms as Zhang et al. [27]. Shi et al. [31] instead used a multi-view method that had the novelty of using sectional views as well as the traditional auxiliary views.

Recently, there has been work representing a B-Rep graphically, where the segmentation of the B-Rep can be related to a node/vertex classification problem. Cao et al. [32] proposed a graph neural network for learning from the B-Rep with the application being machining feature recognition. The approach operated on the B-Rep face adjacency graph directly using a graph neural network. Each vertex in the adjacency graph represented a B-Rep face's planar equation in the CAD model. The method was limited to CAD models with planar faces. UV-Net [33] looked at learning from the geometry of u and v parameters of the surfaces and the topology of the face adjacency graph. Each surface was represented as a 2D grid of sampled point coordinates from the UV space and edges were discretized into 1D grids in U space. 1D and 2D CNNs learned from these grids respectively and a graph neural network learned from the face adjacency graph. BRepNet [34] used the coedges to perform topological walks across the B-Rep to encapsulate information on the mating coedges, faces and edges. Lately, AutoMate [35] has utilized the learning of B-Rep topological entities for automatic mating of CAD assemblies. Both BRepNet and AutoMate do not learn from the geometric surface of the B-Rep.

3. Hierarchical graph representation

The B-Rep shape representation, as used in most mechanical CAD systems, is difficult to be the direct input for neural network architectures due to its continuous nature [33]. However, the B-Rep structure congregates much rich information (i.e., surface geometry, edge convexity and face topology) which is useful for machining feature recognition. Representing this topological information as input features for alternative shape representation neural networks, like point clouds and voxels, is not possible due to the intractable number of input features for each CAD model from the differing number of B-Rep entities. By representing the topological information graphically, a compact descriptor of this information can be produced and operated on by a graph neural network. Hence, the hierarchical B-Rep graph shape representation is proposed as shown in Fig. 1.

The hierarchical B-Rep graph is composed of two levels: a B-Rep face adjacency graph level and a mesh facet graph level. The first level describes the B-Rep face adjacency, and therefore captures elements of the topology of the CAD model. It can also define the edge convexity between the faces as will be discussed in Section 4.2. Each vertex in the graph corresponds to B-Rep face with a set of features attributed to it. These vertex features are as follows:

- Machining feature class label,
- Face type (planar, cylindrical, conical etc.),
- Face area,
- Face centroid coordinates.

The mesh facet graph level is used to describe the geometry of the surface of the CAD model. It is constructed from the facets of a triangular mesh, where each vertex in the graph corresponds to a facet in the mesh. The mesh could be constructed by converting the CAD model to the standard triangle language (STL) file format [36], for example, as is tested during later experiments in this work. Each vertex of the graph encodes components of the planar equation of the facet, given by

$$ax + by + cz + d = 0. \tag{1}$$

This planar equation is determined by an arbitrary point $P_0 = (x_0, y_0, z_0)$ on the surface of the facet plane and the normal of the facet $N = (a, b, c)$ where at least one of the coefficients a, b or c must be non-zero. Given point P_0 and normal N, the d coefficient is calculated as

$$d = -(ax_0 + by_0 + cz_0). \tag{2}$$

Between the vertices of the different graph levels, there are persistent links symbolizing which facets belong to each B-Rep face. Within this representation, a B-Rep face can have more than one facet attributed to it.

4. Neural architecture

4.1. Graph theory

A graph G consists of a set of vertices or nodes $v_i \in V$ linked by a set of edges $e_{ij} = (v_i, v_j) \in E$, where $G = (V, E)$. The topology of the graph is described by an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ where n

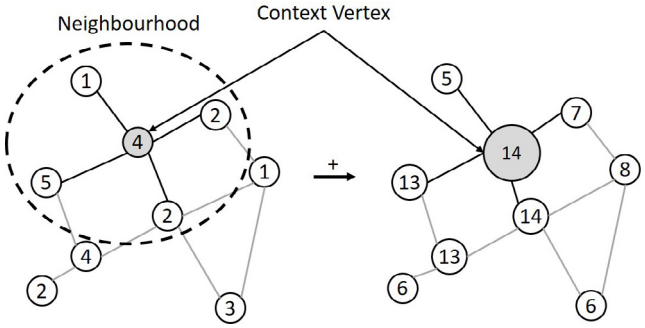


Fig. 2. Spatial graph convolution.

is the number of vertices in the graph. If there is an edge between vertices v_i and v_j then

$$\boldsymbol{A}_{ij} = \begin{cases} 1 & \text{if edge between } i \text{ and } j, \\ 0 & \text{otherwise}. \end{cases} \tag{3}$$

Within graph theory, a graph can be said to be directed or undirected. A directed graph consists of oriented edges where the connections between vertices are ordered, and the resulting adjacency matrix is asymmetric. An undirected graph does not record edge orientation, and the vertex pairs are unordered, providing a symmetric adjacency matrix. Another concept within graph theory is loops or self-loops where $e_{ii} = (v_i, v_i)$. This is when an edge's end-vertices are identical [37].

In the approach described here, no constraints are placed on the adjacency matrix $\boldsymbol{A}$ apart from what is given in Eq. (3). It is assumed that the hierarchical graphs are undirected. Self-loops are also permissible and can represent situations in a CAD model where a B-Rep face is adjacent with itself, e.g., a hole feature with one B-Rep face and one bounding edge along the length of the hole.

4.2. Spatial graph convolutional layer

Graph neural networks (GNNs) [38] are the class of neural networks that learn from graphs. Within this class, there is a subset of GNN called graph convolutional networks (GCNs) [38] that take advantage of the properties of convolutional layers for graphical structures. A convolution is a linear operation that involves the multiplication of a filter with an input to find a given pattern in the data. These convolutional layers have been highly successful for tasks involving 2D image data due to the properties of local connectivity and shift invariance. Local connectivity means that a neuron in a subsequent neural layer has a connection to only a small, localized region of neurons in the previous layer (e.g., a vertex neighbor). This allows for a reduction in the number of parameters in the neural network. Shift invariance means that a convolution operation should have the same meaning no matter its location in the graph or the graph's size.

The different graph convolutions can be divided into three types: spectral, geometric and spatial. For this work, a spatial graph convolution was chosen due to it having the beneficial property of being able to operate on heterogeneous graphs. This means that graph samples with differing numbers of vertices and structures can be learned from, which is necessary for learning from the B-Rep, where the number of entities can differ. The spatial graph convolution works by aggregating a vertex's features and its neighbor's features to produce a new context vertex as shown in Fig. 2, where the filters define the influence of these features in the output.

The spatial graph convolution first proposed by Such et al. [39] was chosen for the neural architecture herein. This convolutional layer looks at learning a cascade of small filters $\boldsymbol{H}$ that could capture the information like a single large filter. This was achieved by using multiple consecutive convolutional layers that through many neighbor vertex hops, vertex information could be diffused across the graph.

The graph filter $\boldsymbol{H}$ is given by

$$\boldsymbol{H} \approx h_0\boldsymbol{I} + h_1\boldsymbol{A}, \tag{4}$$

where $\boldsymbol{I}$ is the identity matrix of the graph (allowing for each vertex's own features to be included in the aggregation), $\boldsymbol{A}$ is the adjacency matrix, and h_0 and h_1 are scalar coefficients that are learned by the network to regulate the impact of the vertex features and neighborhood features respectively during the convolution. For a given vertex signal of a graph $\boldsymbol{V}_{in}$, the convolved output signal $\boldsymbol{V}_{out}$ is

$$\boldsymbol{V}_{out} = h_0\boldsymbol{I}\boldsymbol{V}_{in} + h_1\boldsymbol{A}\boldsymbol{V}_{in} + \boldsymbol{b}, \tag{5}$$

where bias, $\boldsymbol{b}$, is a learned parameter used to shift the activation function in a positive or negative direction. This allows the model to have a greater range of values in which a neuron can be considered activated or not, adding flexibility and reducing overfitting.

The convolutional layer described above assumes that there is only one type of edge in the adjacency matrix $\boldsymbol{A}$. This means that the edge convexity cannot be learned using this graph convolution. An alternative adjacency tensor $\mathcal{A} \in \mathbb{R}^{n\times n\times L}$ can be created where L is the number of adjacency matrix slices in the tensor corresponding to the different edge types $(\boldsymbol{I}, \boldsymbol{E}_1, \boldsymbol{E}_2 \cdots \boldsymbol{E}_{L-1})$. In this paper, three edge types are used: convex, concave and other. Eq. (4) can be extended so that the graph filter represents the convex combination of each different edge type tensor slice as

$$\boldsymbol{H} \approx \sum_{l=1}^{L} h_l\mathcal{A}_l. \tag{6}$$

Each vertex may have many features attributed to it, where C indicates the total number of vertex features. As there are two graph levels in the hierarchical graph with differing numbers of vertices and vertex features, different adjacency matrices and filters are required for each level. In this paper, the vertices of the B-Rep face adjacency graph level will be denoted as $\boldsymbol{V}_1 \in \mathbb{R}^{N\times C}$ and adjacency tensor as $\mathcal{A}_1 \in \mathbb{R}^{N\times N\times L}$. The vertices of the facet graph level are denoted as $\boldsymbol{V}_2 \in \mathbb{R}^{M\times C}$ and the adjacency tensor as $\mathcal{A}_2 \in \mathbb{R}^{M\times M\times 1}$ as there is only one edge type in this graph level. To allow for the filtering of multiple vertex features $h_l \in \mathbb{R}^C$, $\boldsymbol{H}$ is converted to a tensor of $\boldsymbol{H} \in \mathbb{R}^{N\times N\times C}$ for B-Rep level 1 and $\boldsymbol{H} \in \mathbb{R}^{M\times M\times C}$ for facet level 2. A given tensor slice of $\boldsymbol{H}$ can be found using

$$\boldsymbol{H}^{(c)} \approx \sum_{l=1}^{L} h_l{}^{(c)}\mathcal{A}_l. \tag{7}$$

The convolution operation used to filter the vertex signal $\boldsymbol{V}_{in}$ with multiple vertex features is

$$\boldsymbol{V}_{out} = \sum_{c=1}^{C} \boldsymbol{H}^{(c)}\boldsymbol{V}_{in}{}^{(c)} + \boldsymbol{b}. \tag{8}$$

To allow for the learning of a multitude of features in each convolutional layer, multiple filters J are used. Therefore, filter $\boldsymbol{H} \in \mathbb{R}^{N\times N\times C\times J}$ and the vertex output signal $\boldsymbol{V}_{out}$ for each graph level become

$$\mathbf{V}_{1,\text{out}}^{(\text{NxJ})} = \underbrace{\left(\boldsymbol{H}_1^{(\text{N}\times\text{N}\times\text{C}\times\text{J})}\mathbf{V}_{\text{in}}^{(\text{N}\times\text{C})}\right)}_{\text{N}\times\text{J}} + \boldsymbol{b}_1^{(N\times J)}, \tag{9}$$

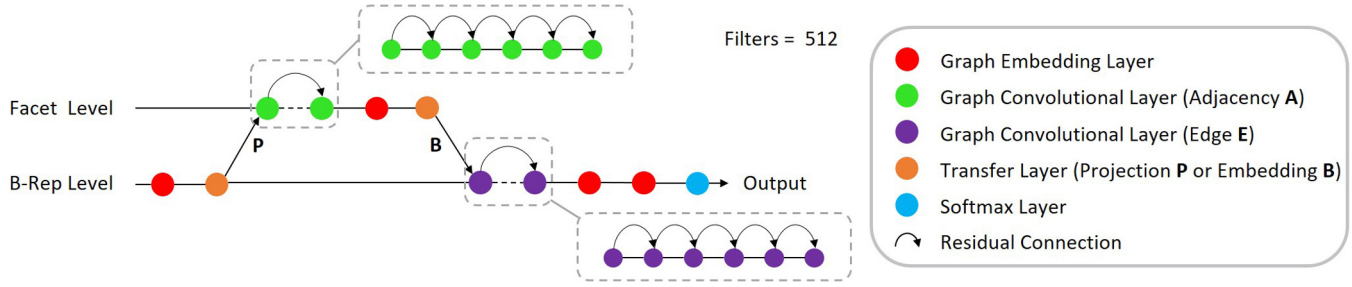


Fig. 3. Hierarchical CADNet neural architecture.

and

$$\mathbf{V}_{2,\text{out}}^{(\text{MxJ})} = \underbrace{\left(\boldsymbol{H}_2^{(\text{M}\times\text{M}\times\text{C}\times\text{J})}\mathbf{V}_{\text{in}}^{(\text{M}\times\text{C})}\right)}_{\text{M}\times\text{J}} + \boldsymbol{b}_2^{(M\times J)}. \tag{10}$$

4.3. Graph pool embedding layer

Within CNNs, there are generally pooling layers, after the convolutional layers to reduce the neural network's sensitivity to the location of features in the input. This is achieved by down sampling and summarizing the output of the convolutional layer to make the network more shift invariant. This reduction in dimensions also enables the convolutional filters to have a larger filter and improve the overall computational efficiency.

Common pooling operations utilized for 2D CNNs are designed to work for grid-like structures and cannot be transferred to heterogeneous graphs. In Hierarchical CADNet, a graph pool embedding proposed by Such et al. [39] is used. The graph pool embedding is a convolutional layer that outputs an embedding matrix of a fixed size of N' or M' vertices. The learned embedding filter is used to construct an embedding matrix as

$$\boldsymbol{V}_{emb}{}^{(N')} = \sum_{c=1}^{C} \boldsymbol{H}_{emb}{}^{(c,N')}\boldsymbol{V}_{in}{}^{(c)} + \boldsymbol{b}. \tag{11}$$

4.4. Transfer layer

The method of learning from a hierarchical graph structure is inspired by Li et al. [40]. Linear operators $\boldsymbol{A}_3 \in \mathbb{R}^{N\times M}$ and $\boldsymbol{A}_4 \in \mathbb{R}^{M\times N}$ are used to allow for information-sharing between the two distinct dimensional features of each graph level. $\boldsymbol{A}_3$ embeds information from facet level 2 to B-Rep level 1 by operating on the facet vertex vectors $\boldsymbol{V}^{(m\times 1)}$, where there is a facet vertex vector for each B-Rep vertex feature. While $\boldsymbol{A}_4$ projects information in the reverse and operates on the B-Rep vertex vector $\boldsymbol{V}^{(n\times 1)}$. With there being a B-Rep vertex vector for each facet vertex feature. Across each graph level, parallel filter-learning is used. These are the learned embedding filters $\boldsymbol{B} = h_3\boldsymbol{A}_3$ and the learned projection filter $\boldsymbol{P} = h_4\boldsymbol{A}_4$. The connections between the different graph level feature spaces are residual sums of each vertex vector, shown as

$$\boldsymbol{V}_1{}^{(N\times 1)} = \boldsymbol{B}^{(N\times M)}\boldsymbol{V}_2{}^{(M\times 1)} + \boldsymbol{V}_1{}^{(N\times 1)} \in \mathbb{R}^N, \tag{12}$$

and

$$\boldsymbol{V}_2{}^{(M\times 1)} = \boldsymbol{P}^{(M\times N)}\boldsymbol{V}_1{}^{(N\times 1)} + \boldsymbol{V}_2{}^{(M\times 1)} \in \mathbb{R}^M. \tag{13}$$

4.5. Softmax layer

A softmax layer in neural networks is used to help decide which class to denote the vertex as. It works by assigning a probability to each class in the dataset. Whichever class has the highest probability is assigned as the predicted class, which in the case of work herein would be the machining feature class of the B-Rep face. The softmax layer equation is

$$\sigma\left(\boldsymbol{V}_{in}\right)_i = \frac{e^{\boldsymbol{V}_{ini}}}{\sum_{k=1}^{K} e^{\boldsymbol{V}_{ink}}}, \tag{14}$$

where $\boldsymbol{V}_{in} \in \mathbb{R}^{K\times N}$, K is the maximum number of machining feature classes and i is a given machining feature class.

4.6. Residual connections

As the number of graph convolutional layers are increased, it was observed that above a certain point, the accuracy of the network decreases. This is due to what is known as the vanishing gradient and over-smoothing, in which the error backpropagated through the network becomes infinitesimally small [41]. Within Hierarchical CADNet, residual connections are implemented as in [42], using

$$\boldsymbol{V}_{out} = \mathbf{K}\left(\boldsymbol{V}_{in}, \mathbf{H}\right) = \mathbf{F}\left(\boldsymbol{V}_{in}, \mathbf{H}\right) + \boldsymbol{V}_{in}, \tag{15}$$

$$\boldsymbol{V}_{out} = \boldsymbol{V}_{out}^{\text{res}} + \boldsymbol{V}_{in}, \tag{16}$$

to help minimize the vanishing gradient. The entire Hierarchical CADNet network architecture with residual connections can be seen in Fig. 3.

The residual connections work by trying to learn an underlying mapping $\boldsymbol{K}$ by fitting another residual mapping $\boldsymbol{F}$. The mapping $\boldsymbol{F}$ learns to transform the vertex input signal $\boldsymbol{V}_{in}$ into a residual vertex signal representation $\boldsymbol{V}_{out}^{res}$ for the next layer. A vertex-wise addition between $\boldsymbol{V}_{in}$ and $\boldsymbol{V}_{out}^{res}$ gives $\boldsymbol{V}_{out}$.

5. Dataset creation

5.1. Issues with current machining feature datasets

To train a deep learning algorithm for a particular task, a large-scale dataset is required with characteristics comparable to the data it will be applied to. There have been large CAD model datasets proposed such as ABC dataset [43] and MCB dataset [44], however, they do not contain the necessary labels required for a machining feature recognition task, and obtaining these labels would be a non-trivial task. For the machining feature recognition task, there has been a couple of datasets put forward. Zhang et al. [27] provided a dataset with 24 different machining features classes as seen in Fig. 4. In that work, each CAD model only contains one machining feature class and so it cannot be used for learning to differentiate intersecting machining features.

Cao et al. [32] described a method of automatically generating CAD models with a random sampling of the 24 machining features described by Zhang et al. [27] and provided the MFCAD dataset [45]. The MFCAD dataset only consisted of planar machining features and contained few instances with intersecting machining features. It was limited in the number of machining features that could be applied per sample and the types of shapes that could be produced. The main reasons for these issues with the dataset creation tool were as follows:

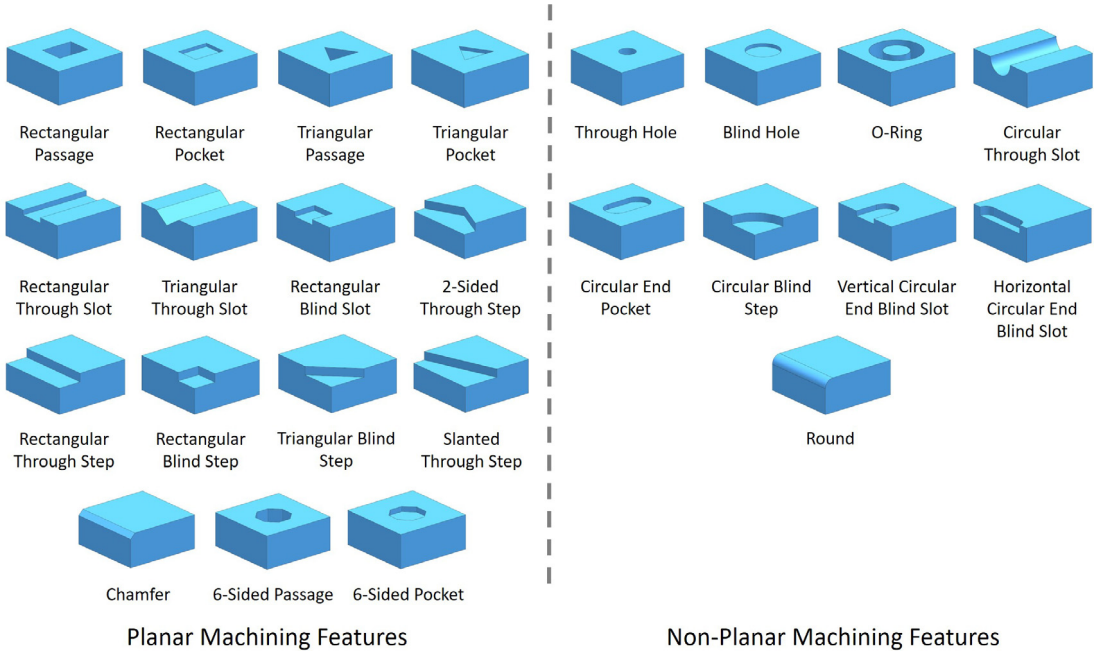


Fig. 4. Machining feature classes: planar and non-planar.

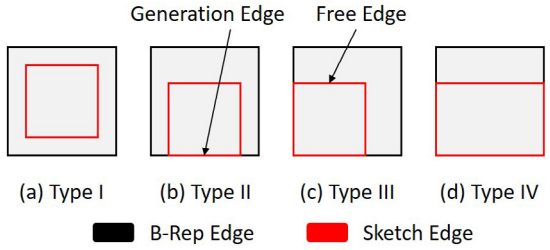


Fig. 5. Sketch types.

1. The stock model on which the machining features are applied is always cubic in shape.
2. The addition of some machining features can result in the CAD model being split into multiple solids.
3. Pockets and passages which lay within the bounds of the B-Rep faces are unable to intersect with one another.
4. Certain machining features produced in the models could not be manufactured using conventional milling operations.

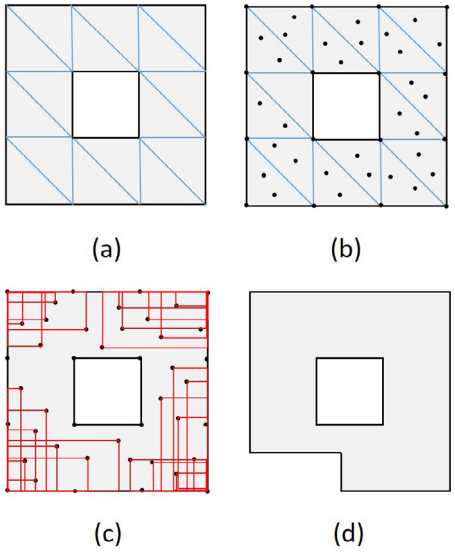


Fig. 6. Machining feature generation process for sketch type III (a) tessellation, (b) point sampling, (c) bounding box generation & (d) applied machining feature.

5.2. MFCAD dataset creation tool

In this work, the dataset creation procedure from Cao et al. [32] is used as a starting point. In [32], four different sketch types were defined to produce each machining feature as seen in Fig. 5. Each differs by the number of generation and free edges, where a generation edge lies on the outer boundary of the B-Rep face the feature is being added to, and the free edge lies within the outer boundary of the B-Rep.

To generate a new machining feature, a face in the CAD model is randomly selected. The face is tessellated into triangular facets as seen in Fig. 6a. Points are sampled within each facet as shown in Fig. 6b. From the sampled points and the vertices of the facets, sketch bounding boxes are created, where a number of edges of the bounding box will align with the edges of the B-Rep faces according to the desired sketch type (Fig. 6c). Within the process, one bounding box is randomly chosen, and the height and width are randomly altered where the range of parameters is between a user-defined lower bound variable and the maximum parameters of the bounding box. The final bounding box is then used to create the machining feature (Fig. 6d).

5.3. MFCAD++ dataset creation tool improvements

The MFCAD++ dataset tool can produce 25 different classes including the 24 machining features in Fig. 4 and the stock face. PythonOCC [46] is used for the dataset generation and the Numba Python library [47] is utilized for computation acceleration.

The stock CAD model size is altered from being a $10 \times 10 \times 10$ cube, to a cuboidal billet with dimensions ranging between 10–50. The x, y and z dimensions are randomly selected from this range for each CAD model sample.

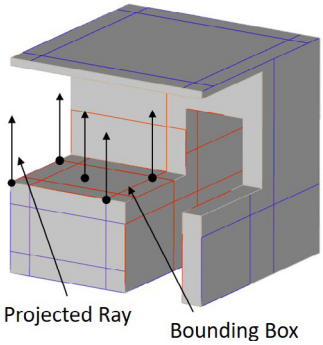


Fig. 7. Check if new machining feature can be manufactured by projecting rays to locate perpendicular faces. Red *Bounding Boxes* = Cannot be Manufactured & Blue *Bounding Boxes* = Can be Manufactured.

Before creating the dataset, a random set of machining feature sequences are generated from the different possible sequence permutations. We propose to group the different machining features into 5 distinct groups and reorder the feature creation sequences into the order given below:

1. Steps
2. Slots
3. Through passages, pockets or holes
4. Blind passages, pockets or holes
5. Transition features (chamfers and rounds).

By having the step and slot features at the start of the sequences, the outer boundaries of the B-Rep faces are defined in the CAD model. Therefore, the type I sketch bounding boxes only need to be calculated once, saving on computational costs. With the MFCAD dataset creation process, the sketch bounding boxes are created each time a new inner bound machining feature is applied. This meant the new bounding boxes would not intersect with the previous inner bound machining features and therefore these types of machining features would not intersect in the dataset. The feature sequence length was set to 3–10 machining features.

To prevent the creation of multiple solids after adding a new machining feature, the CAD model is checked to determine if the new feature resulted in more than one solid. If so, the operation is reversed, and so the feature is not added.

Given the desire to have a dataset representative of parts useful for manufacturing research, another check is to ensure that the machining features within the CAD model can be machined using conventional milling manufacturing methods, and that they maintain their semantic meaning. This means that machining features should not be generated from sketch bounding boxes that milling operations would not be able to reach. This situation occurs when there is a face directly opposite the sketch bounding box, for example. Therefore, before a sketch bounding box is chosen, the potential bounding boxes in the CAD model should be checked that they adhere to this rule.

To test if a bounding box can be used for sketch generation, five points are selected. Specifically, four from the bounding box vertices and the last being the centroid of the box. Rays are fired in the direction of the face normal to determine if any of the rays intersect any of the facets of the tessellated faces of the CAD model. If so, then the bounding box is not considered for sketch generation. Fig. 7 shows invalid machining bounding boxes in red and valid ones in blue.

Another instance where a machining feature may lose its semantic meaning is when a concave edge is selected as a generation edge. Fig. 8 shows a circular blind step generated on a concave edge where the feature can no longer be considered a step. The solution used to mitigate this, is to filter out any concave edges from the scope of generation edges.

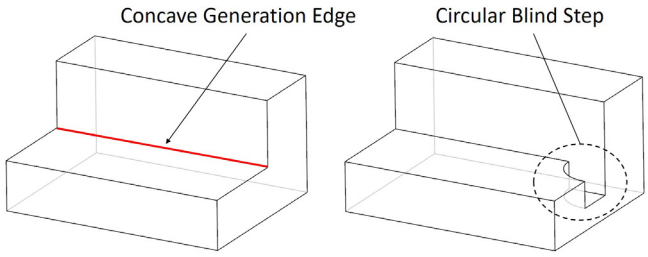


Fig. 8. Generation of a machining feature using a concave edge.

6. Results and discussion

A range of experiments was performed to find the viability of Hierarchical CADNet for learning from the B-Rep and the application of machining feature recognition. Hierarchical CADNet is implemented using TensorFlow 2 [48] and run on a NVIDIA GeForce RTX 2080Ti GPU. Each training batch contains a varying number of hierarchical graphs with the sum of the number of vertices (i.e. B-Rep faces and facets) in all the graphs being equal to or less than 10,000 vertices. Weight initialization is performed using Xavier initialization [49]. Batch normalization [50] and dropout [51] are used after each convolutional layer, where further information on their influence on the network is given Appendix C. In each experiment, 512 filters are used for each graph convolutional layer. A learning rate with decay is utilized with an initial learning rate of 0.01, as well as the ADAM optimizer [52] and cross-entropy loss [53]. The dropout rate is set to 0.3. For each dataset tested, besides the MFCAD dataset, a 70:15:15 training/validation/test split is used and the number of epochs is set to 100, where one epoch signifies that the entire dataset has been passed through the network once.

Within each experiment, two versions of Hierarchical CADNet are tested. One utilizes the edge convexity information denoted as Hierarchical CADNet (Edge) and the second uses only the B-Rep face adjacency matrix denoted Hierarchical CADNet (Adj). A different number of convolutional layers are used for each dataset (more details of the selected numbers are given in Appendix C).

6.1. Single machining feature dataset

As previous machining feature recognition approaches like FeatureNet [27] could not be trained to perform a segmentation task, the work herein is compared with them on a dataset containing CAD models with only one class of machining feature, where a classification task is performed. FeatureNet provided the dataset used in their paper, however, only STL versions of the CAD models are available and not the original B-Reps. Therefore, a new comparable dataset is created with 60,000 CAD models (24 machining feature classes with 2,500 samples per class). For each CAD model sample, the networks would need to learn to classify which machining feature is present in the CAD model.

To allow Hierarchical CADNet to be able to predict a single machining feature, a global average pooling layer [54] is added before the softmax layer in the neural architecture. This pooling layer reduces the number of neurons in the network to the same number of hierarchical graphs in the batch being passed to the network. The results from this experiment are shown in Table 1, where the Hierarchical CADNet architectures shown used 7 graph convolutional layers. For FeatureNet [27], Peddireddy [28] and

Table 1
Classification accuracy and runtime results for single machining feature dataset.

Network	Accuracy (%)	Test time (s)	Number of parameters
FeatureNet [27]	94.11	17.36	3.17M
Peddireddy et al [28]	90.58	17.35	**2.99M**
MSVNet [31]	99.81	26.72	128.86M
Hierarchical CADNet (Adj)	99.99	**6.36**	7.67M
Hierarchical CADNet (Edge)	**100.00**	6.67	11.34M

MSVNet [31], a voxel resolution of 64^3 is utilized, with this being the voxel resolution FeatureNet obtained its highest classification accuracy. MSVNet is pre-trained on the ImageNet dataset [55] and is set to use 12 section cuts.

From Table 1, it can be seen that each version of Hierarchical CADNet outperforms the previous networks on both classification accuracy and test time. The test time for Hierarchical CADNet (Adj) is better than that of the edge convexity version as it did not need to operate on the costly adjacency tensor $\mathcal{A}$. Hierarchical CADNet (Edge) is able to correctly classify each sample in the test set. This shows the strength of operating on both geometric and topological information and its ability to outperform previous voxel and multi-view methods.

6.2. MFCAD dataset

Cao et al. [32] operated on CAD models with planar surfaces, therefore, to compare Hierarchical CADNet with the approach in [32], the MFCAD dataset is used [45]. This dataset consists of 15,488 CAD models, where each CAD model has been generated from a stock cube and 16 planar machining feature classes as can be seen in Fig. 4. Examples from the MFCAD dataset can be found in Appendix D, where it can seen that each CAD model is relativity simplistic, lacking in intersecting machining features. The dataset is split 60:20:20 as per the original paper.

A segmentation task is performed on the MFCAD dataset where the neural networks attempt to predict the machining feature label for each B-Rep face in the CAD model. To evaluate this, the accuracy per B-Rep face is calculated as

$$\textit{Accuracy per Face} = \frac{\textit{Correct \# Faces}}{\textit{Total \# Faces}}. \tag{17}$$

For networks where the prediction is not based on the B-Rep face but a primitive such as a point or edge, per-face voting is used to find the prediction per B-Rep face. The accuracies in terms of the given primitive type are also given. The mean Intersection over Union (IoU) is often used as a metric in segmentation tasks. However, as each machining feature class is represented sparsely across the dataset, where a sample may not contain most of the machining feature classes; this inflates the mean IoU score as these classes will be represented as being correct in the metric.

The results on the MFCAD dataset are shown in Table 2. For the UV-Net [33] and MeshCNN [19], the results are referenced from [33] and 350 epochs are used for training each network as was used in [33].

For each hierarchical CADNet architecture, four graph convolutional layers are used. PointNet++ and DGCNN are trained with 2048 points with additional surface normal information at each point. The number of parameters in each network architecture are also given. More information on network configurations is given in Appendix A.

As can be seen in Table 2, Hierarchical CADNet achieves lower but still comparable results on the MFCAD dataset, where it was 0.05% lower in accuracy per B-Rep face than the highest accuracy achieved by Cao et al. [32] and UV-Net. This may be due to the increased number of parameters of the network in comparison to the others, requiring more data for sufficient training. In terms of performance on individual machining features, the two features with the lowest F1 score [56] are the rectangular through step and the slanted through step with scores of 99.34% and 99.38%, respectively. The precision is higher for these machining features in comparison to the networks recall score. Therefore, when the network makes a prediction on a B-Rep face pertaining to either rectangular or slanted through steps, it has a high probability of being correct. However, as the recall is lower, this means that B-Rep faces that belong to these classes, are often being misclassified.

Table 2
Segmentation accuracy per face and primitive, and number of parameters for MFCAD dataset.

Network	Accuracy per face (%)	Accuracy per primitive (%)	Number of parameters
UV-Net [33]	**99.95**	–	1.23M
PointNet++ [17]	91.35	94.95	1.42M
DGCNN [18]	90.99	71.00	0.53M
MeshCNN [19]	99.89	**98.52**	2.29M
Cao et al [32]	**99.95**	–	**0.53M**
Hierarchical CADNet (Adj)	98.24	–	4.49M
Hierarchical CADNet (Edge)	99.90	–	6.60M

Table 3
Segmentation accuracy per face and primitive, and number of parameters for MFCAD++ dataset.

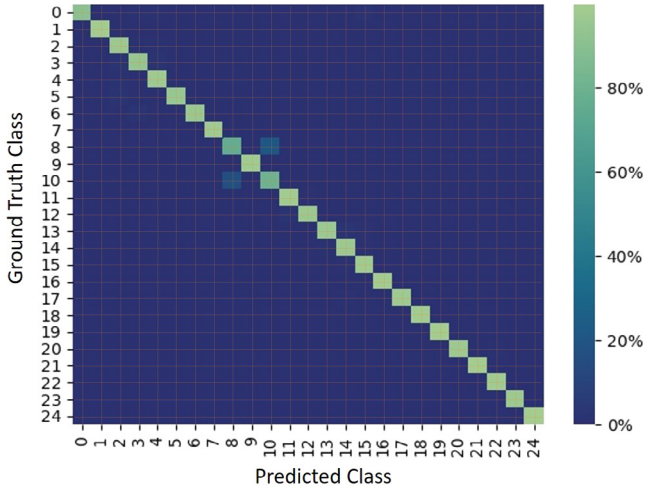
Network	Accuracy per face (%)	Accuracy per primitive (%)	Number of parameters
PointNet++ [17]	85.88	**93.48**	1.42M
DGCNN [18]	85.98	91.28	**0.53M**
Hierarchical CADNet (Adj)	96.52	–	6.62M
Hierarchical CADNet (Edge)	**97.37**	–	9.76M

6.3. MFCAD++ dataset

A new machining feature dataset called MFCAD++ is created using the improved dataset creation tool as described in Section 5.1. This consists of 59,655 CAD models with machining features with both planar and non-planar faces. Each CAD model contains between 3 to 10 machining features. Samples from the datasets can be seen in Appendix D and details on how it differs from MFCAD are shown in Appendix B.

Table 3 shows the results for the MFCAD++ dataset. Six graph convolutional layers are used for each level in the Hierarchical CADNet architectures. As the UV-Net [33] code is not publicly available at the time of writing, its result on the MFCAD++ dataset could not be established. The results of all methods were lower than on MFCAD dataset. This is believed to be due to the added complexity in the dataset, with intersecting machining features and varying stock billet sizes. Some examples of these intersections are shown in Appendix D.

It can be seen that the architecture described in this work outperforms the others on this dataset. When comparing the different versions of Hierarchical CADNet, it is evident that learning from the edge convexity improves performance for the machining feature recognition task. The results of the Hierarchical CADNet architecture using edge convexity for the test set are visualized



Idx	Machining Features	Idx	Machining Features
0	Chamfer	13	Triangular Pocket
1	Through Hole	14	Rectangular Pocket
2	Triangular Passage	15	6-Sided Pocket
3	Rectangular Passage	16	Circular End Pocket
4	6-Sided Passage	17	Rectangular Blind Slot
5	Triangular Through Slot	18	Vertical Circular End Blind Slot
6	Rectangular Through Slot	19	Horizontal Circular End Blind Slot
7	Circular Through Slot	20	Triangular Blind Step
8	Rectangular Through Step	21	Circular Blind Step
9	2-Sided Through Step	22	Rectangular Blind Step
10	Slanted Through Step	23	Round
11	O-Ring	24	Stock
12	Blind Hole		

Fig. 9. Confusion matrix for segmentation results on the test set of the MFCAD++ dataset using the Hierarchical CADNet (Edge) architecture.

in the confusion matrix shown in Fig. 9, where correct predictions lie on the diagonal of the matrix. It can be seen that the architecture struggles again between the rectangular through step and the slanted through step. To further understand why this could be an ablation study is performed on the network's input features, the results for which are given in Appendix C. The only differentiating aspect between the two machining feature classes is the slanted face. This difference should be able to be learned from the normal information of the mesh level in the hierarchical B-Rep graph. However, from the ablation study, it is clear that the network is insufficiently learning the mesh information in comparison to the B-Rep topological information. Therefore, performance could be boosted by tailoring the neural architecture on the mesh level for learning from mesh data or by directly learning the dihedral angle between the faces on the B-Rep level.

6.4. More complex test cases

Several more complex test cases were produced to see how well the Hierarchical CADNet trained on the MFCAD++ dataset generalized for these new CAD models not seen in the dataset and showcase how Hierarchical CADNet can perform well on types of models, that other voxel methods struggle with. These test cases differ from those in the MFCAD++ dataset by having a larger number of machining features per sample, more chamfered or filleted edges and large aspect ratios in the machining features and stock billet. The results for each test case can be seen in Fig. 10. FeatureNet is chosen as the comparative method, where the network is trained on the original FeatureNet dataset and had a test accuracy of 98.01% (similar to the original paper). Along with Hierarchical CADNet (Edge), another version of Hierarchical CADNet is used called Hierarchical CADNet (Sub). In this version, the hierarchical B-Rep graphs of the MFCAD++ dataset are decomposed into sub-graphs using the edge convexity and used to train Hierarchical CADNet. The rationale with this, is to simplify the task, making Hierarchical CADNet more robust to shapes that varied from those in the MFCAD++ dataset. For both Hierarchical CADNets, accuracy is given in a per B-Rep face manner, and for FeatureNet, the accuracy is given per machining feature. It can be seen that test cases 4 and 5, that both CAD models are not able to be voxelized using a resolution of 64^3. This is due to the large aspect ratio difference between the x, y and z dimensions. It is found that a voxel resolution of 256^3 or higher is needed to voxelize these shapes.

Hierarchical CADNet (Sub) can be seen to outperform Hierarchical CADNet (Edge) on the majority of the test cases, especially on test cases 4 and 5 which have a thin sheet shape. This is assumed to be due to the fact that these kinds of shape do not appear in the MFCAD++ dataset and the network is having difficulty generalizing for them. It is anticipated that if the dataset was supplemented with more shapes like these, the network would improve its performance.

7. Conclusion

This paper has:

- Introduced an improved, automated method of constructing CAD models with machining features. The method allows for the generation of CAD models with intersecting machining features.
- Presented a new machining feature dataset called MFCAD++, which contains more complex CAD models than the previous MFCAD dataset as well as non-planar machining features.
- Proposed a new hierarchical B-Rep graph structure for encoding information about the B-Rep CAD model.
- Presented an encoding structure that represents the surface geometry and face topology of B-Rep CAD models, while also being able to be used as the input representation into a graph neural network.
- Presented a novel hierarchical graph convolutional network called Hierarchical CADNet, which can learn from these hierarchical B-Rep graph structures. This new approach can achieve comparable or better results on several datasets compared to the state-of-art frameworks.
- Showed the benefits of learning the B-Rep face topology and edge convexity information for the task of machining feature recognition.

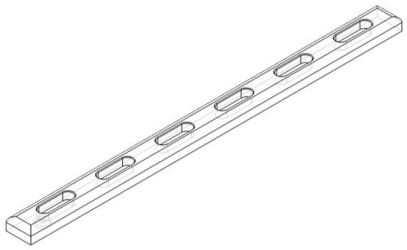
The MFCAD++ dataset [57] is available here: https://pure.qub.ac.uk/en/datasets/mfcad-dataset.

Code relating to the paper is available here: https://gitlab.com/qub_femg/machine-learning.

8. Future work

It would be interesting to compare Hierarchical CADNet's performance on other 3D deep learning tasks such as shape retrieval. For this datasets such as the ABC dataset [43] or MCB dataset [44] could be used. In terms of improving performance, the ability to

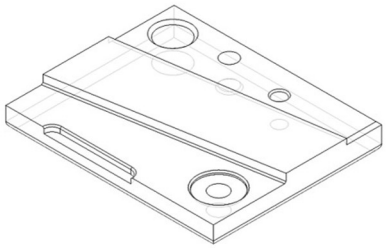
1



Machining Features = 36

Network	Accuracy (%)
Hierarchical CADNet (Edge)	92.54
Hierarchical CADNet (Sub)	100.00
FeatureNet	0.00

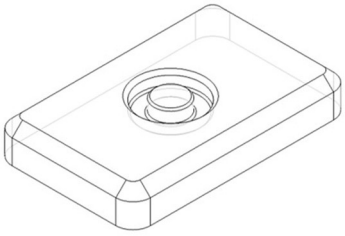
2



Machining Features = 23

Network	Accuracy (%)
Hierarchical CADNet (Edge)	85.71
Hierarchical CADNet (Sub)	82.26
FeatureNet	17.39

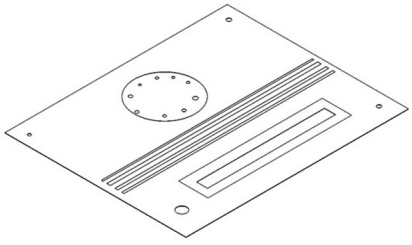
3



Machining Features = 17

Network	Accuracy (%)
Hierarchical CADNet (Edge)	100.00
Hierarchical CADNet (Sub)	92.31
FeatureNet	0.00

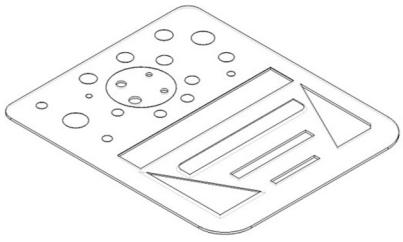
4



Machining Features = 19

Network	Accuracy (%)
Hierarchical CADNet (Edge)	38.00
Hierarchical CADNet (Sub)	98.00
FeatureNet	Failed to Voxelize

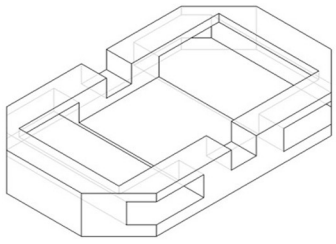
5



Machining Features = 44

Network	Accuracy (%)
Hierarchical CADNet (Edge)	57.33
Hierarchical CADNet (Sub)	92.31
FeatureNet	Failed to Voxelize

6



Machining Features = 8

Network	Accuracy (%)
Hierarchical CADNet (Edge)	43.33
Hierarchical CADNet (Sub)	83.33
FeatureNet	0.00

Fig. 10. Results for different test cases.

Table 4
Average number of machining features per CAD model.

Dataset	Average number of machining features per CAD model
MFCAD	4
MFCAD++	7

Table 5
Average number of B-Rep faces per CAD model.

Dataset	Average number of faces per CAD model
MFCAD	22
MFCAD++	30

learn the dihedral angles between the edges could help better the differentiation between machining features such as slanted and rectangular through steps. The hierarchical graph levels could also be extended to encompass levels for B-Rep edge or vertex entities. Other graph neural architectures could be utilized for the mesh level to see if more geometric information could be extracted and learnt from, however, these would need to be able to work on heterogeneous data structures. Alternatively, how the mesh is described could be changed allowing for better encoding of surface continuity.

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.

Acknowledgment

Author Andrew R. Colligan is a Ph.D. researcher funded through Department for the Economy (DfE) Northern Ireland.

Appendix A. Network configurations

- **FeatureNet** [27]**:** Batch size of 40. Implementation: https://gitlab.com/qub_femg/machine-learning/featurenet-tensorflow-2.
- **MVSNet** [31]**:** Batch size of 64. Implementation: https://github.com/PeizhiShi/MsvNet.
- **PointNet++** [17]**:** Batch size of 64. Implementation: https://github.com/rusty1s/pytorch_geometric/blob/master/examples/pointnet2_segmentation.py.
- **DGCNN** [18]**:** Batch size of 16. Implementation: https://github.com/rusty1s/pytorch_geometric/blob/master/examples/dgcnn_segmentation.py.
- **Cao et al.** [32]**:** Batch size of 32. Implementation: https://gitlab.com/qub_femg/machine-learning/cadnet.

Appendix B. Dataset statistics

In this appendix, more details about the MFCAD++ dataset will be discussed. One way to measure the complexity of the CAD models, allowing for a comparison between the MFCAD and MFCAD++ datasets, is by looking at the number of machining features and B-Rep faces per CAD model. In Table 4, the average number of machining features per CAD model is given, where it is shown that the MFCAD++ dataset had on average three more machining features than the original MFCAD dataset.

When comparing the average number of B-Rep faces per CAD model as shown in Table 5, again the MFCAD++ dataset can be seen to have more complexity with eight more B-Rep faces on average per CAD model than the MFCAD dataset.

The distribution of each edge type in the MFCAD++ dataset is shown in Fig. 11. Most of the edges in the dataset (67%) are convex, with 31% of edges being concave. This is because concave edges will only exist due to the creation of certain machining features. Only a small percentage (2%) are the "other" edge type

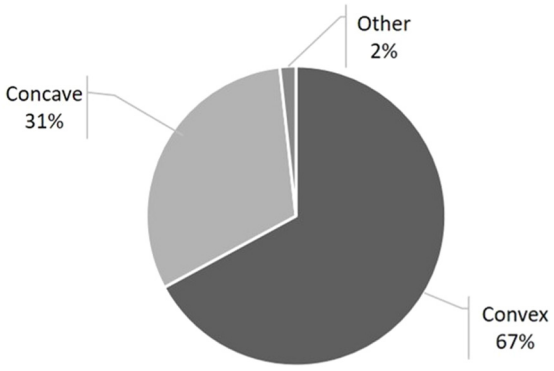


Fig. 11. Distribution of edge types in MFCAD++ dataset.

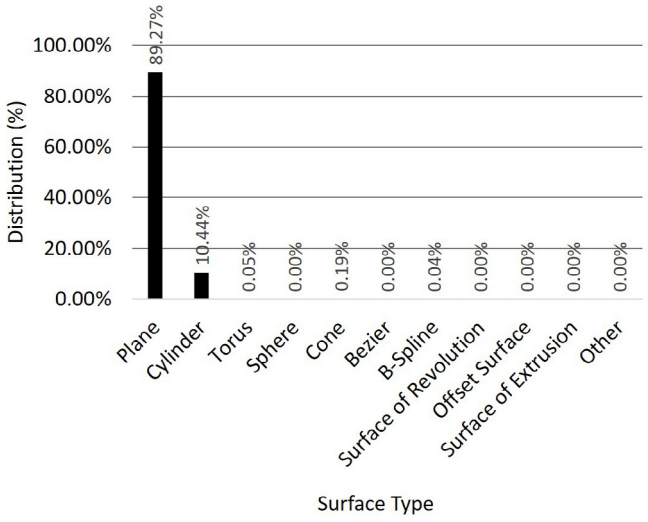


Fig. 12. Distribution of surface types in MFCAD++ dataset.

and shows that there is a small likelihood of this edge type being created using the dataset generation tool.

Fig. 12 shows the distribution of the different surface types within the MFCAD++ dataset. Most of the B-Rep faces (89.27%) have a planar surface type. This is due to only the stock billet only using a cuboidal shape and 62.5% of the machining feature classes being planar in nature. The next most common surface type is the cylinder which can be seen in hole and circular machining features. As can be seen there is a lack of other surface types, with six types not being represented in the dataset.

Fig. 13 highlights the distribution of the machining feature classes in the MFCAD++ dataset, where for an equal distribution each class would have a distribution of 4.2%. The machining feature classes that lie below this distribution are either slots or the round feature. A reason for a lack of round/fillet features in the dataset is due to PythonOCC's filleting tool lacking robustness. This means that many machining feature sequences that contained round/fillet features failed to be generated. The generation of the slot features can be assumed to also be an issue.

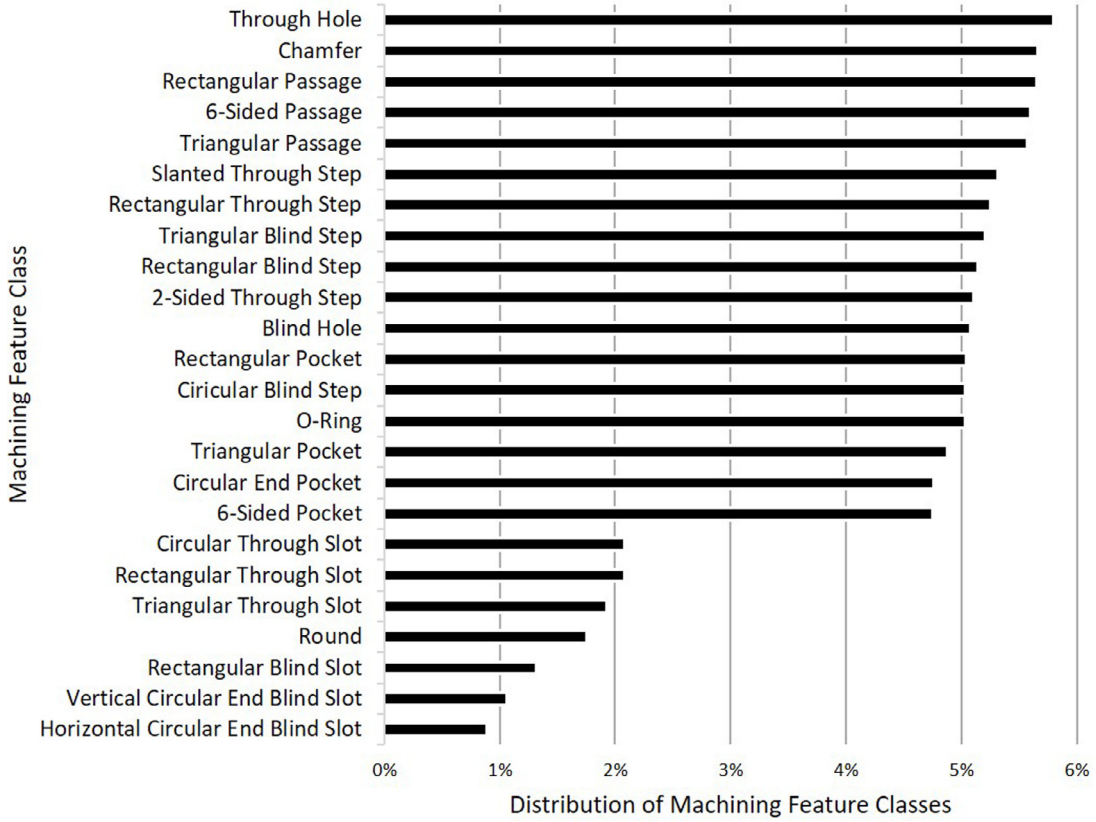


Fig. 13. Distribution of machining feature classes in MFCAD++ dataset.

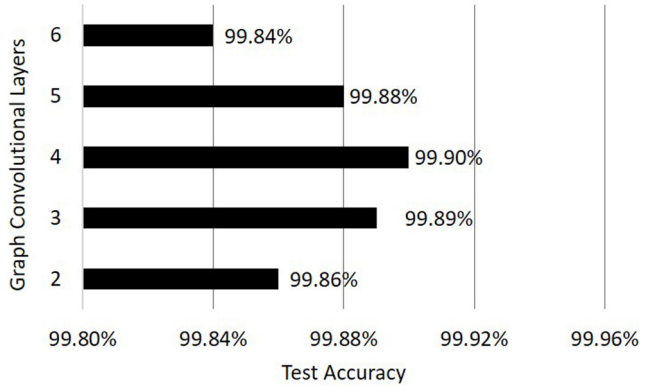


Fig. 14. Differing the number of graph convolutional layers for the MFCAD dataset.

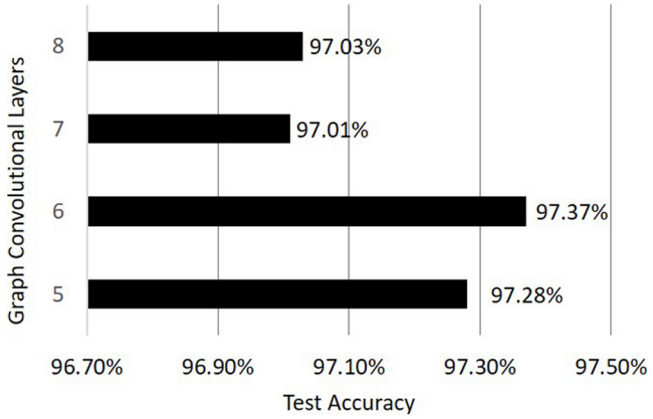


Fig. 15. Differing the number of graph convolutional layers for the MFCAD++ dataset.

Appendix C. Experiments

C.1. Differing the number of graph convolutional layers

A study is performed for each dataset to see which number of graph convolutional layers achieves the highest performance. The number of layers is set the same for each graph level and further study is needed to determine if having a different number of graph convolution layers on each graph level achieves superior results.

Fig. 14 shows the effects of differing the number of graph convolutional layers for the MFCAD dataset. As the number of graph convolutional layers increase, so does the performance on the test set. This performance peaks at four graph convolutional layers and then decreases after that. Therefore, four graph convolution layers are chosen for this dataset.

In Fig. 15, the graph relates to the MFCAD++ dataset. As for the MFCAD dataset, performance on the test set peaks after a certain number of graph convolutional layers and after that number the test accuracy decreases. Unlike with the MFCAD dataset, the number of graph convolutional layers that achieves the highest performance is six. It is assumed this, is because the dataset size and split is different, meaning that there is less data in the training set to learn from, therefore using less graph convolutional layers reduces the case of overfitting.

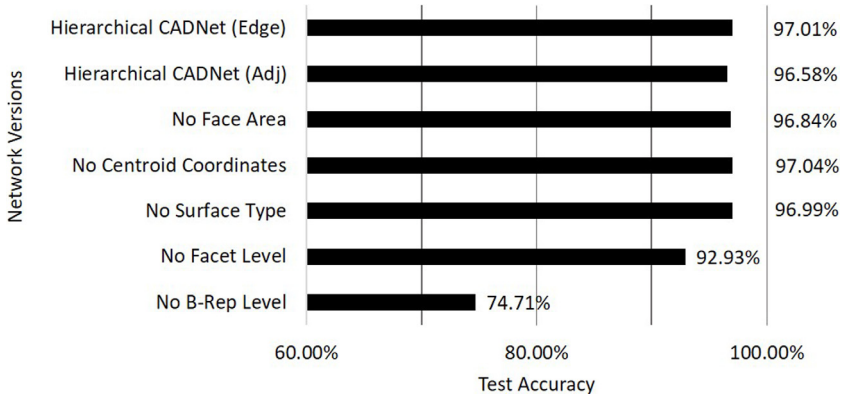


Fig. 16. Ablation study on input features.

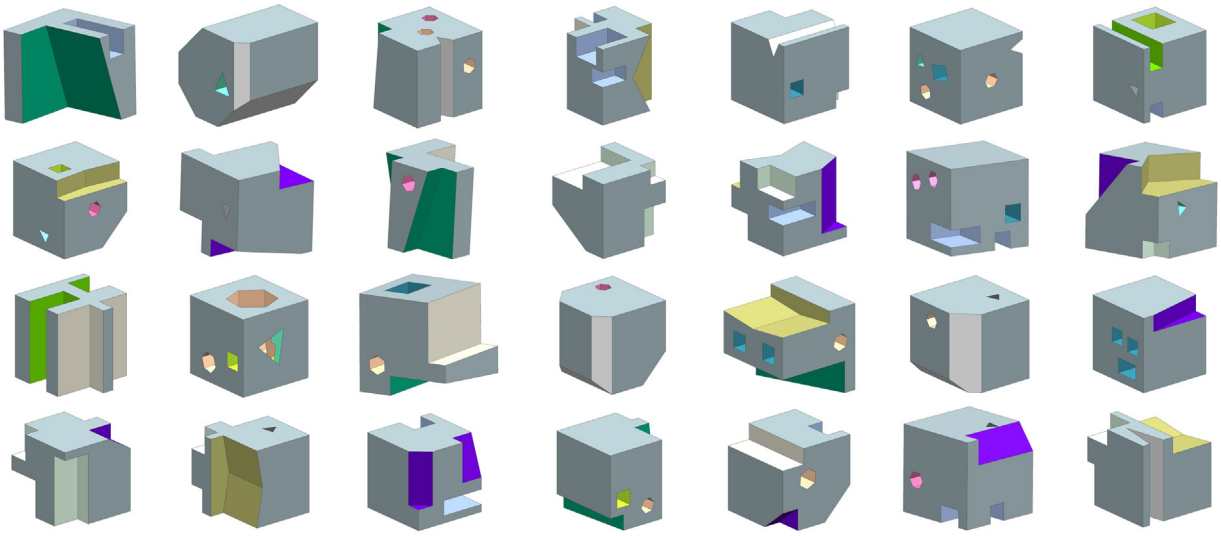


Fig. 17. Samples from MFCAD dataset.

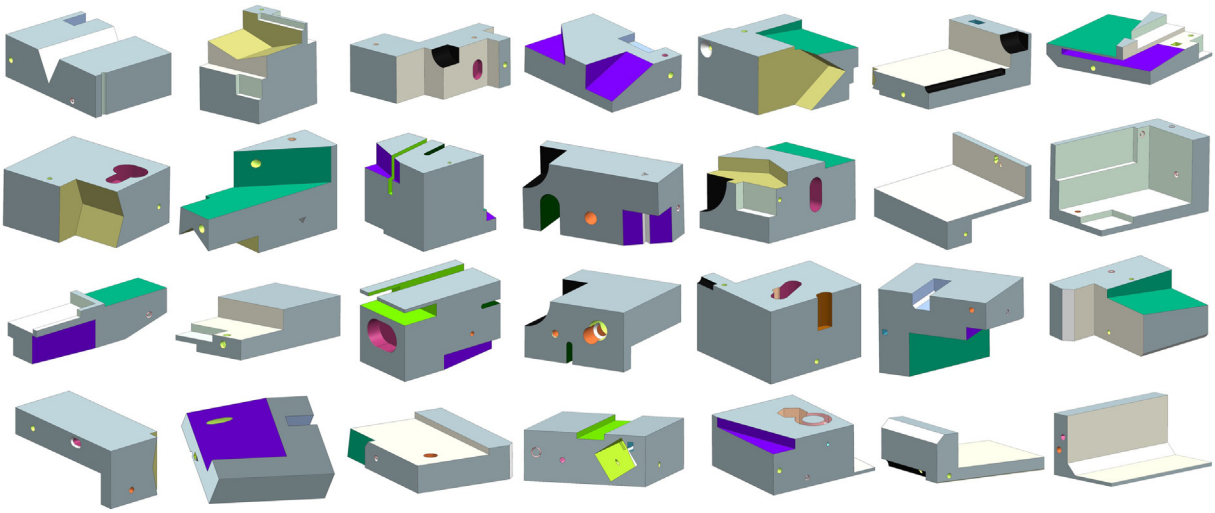


Fig. 18. Samples from MFCAD++ dataset.

C.2. Ablation study on input features

An ablation study is also performed to determine the effect of each input feature and the graph levels on the performance of Hierarchical CADNet on the test set of the MFCAD++ dataset. Fig. 16 shows the results of this study. There is six alternative network versions tested, where the benchmark is Hierarchical CADNet using edge convexity information with seven graph convolutional layers. The input features of the B-Rep graph level can be seen to have the least effect on performance. However, test accuracy

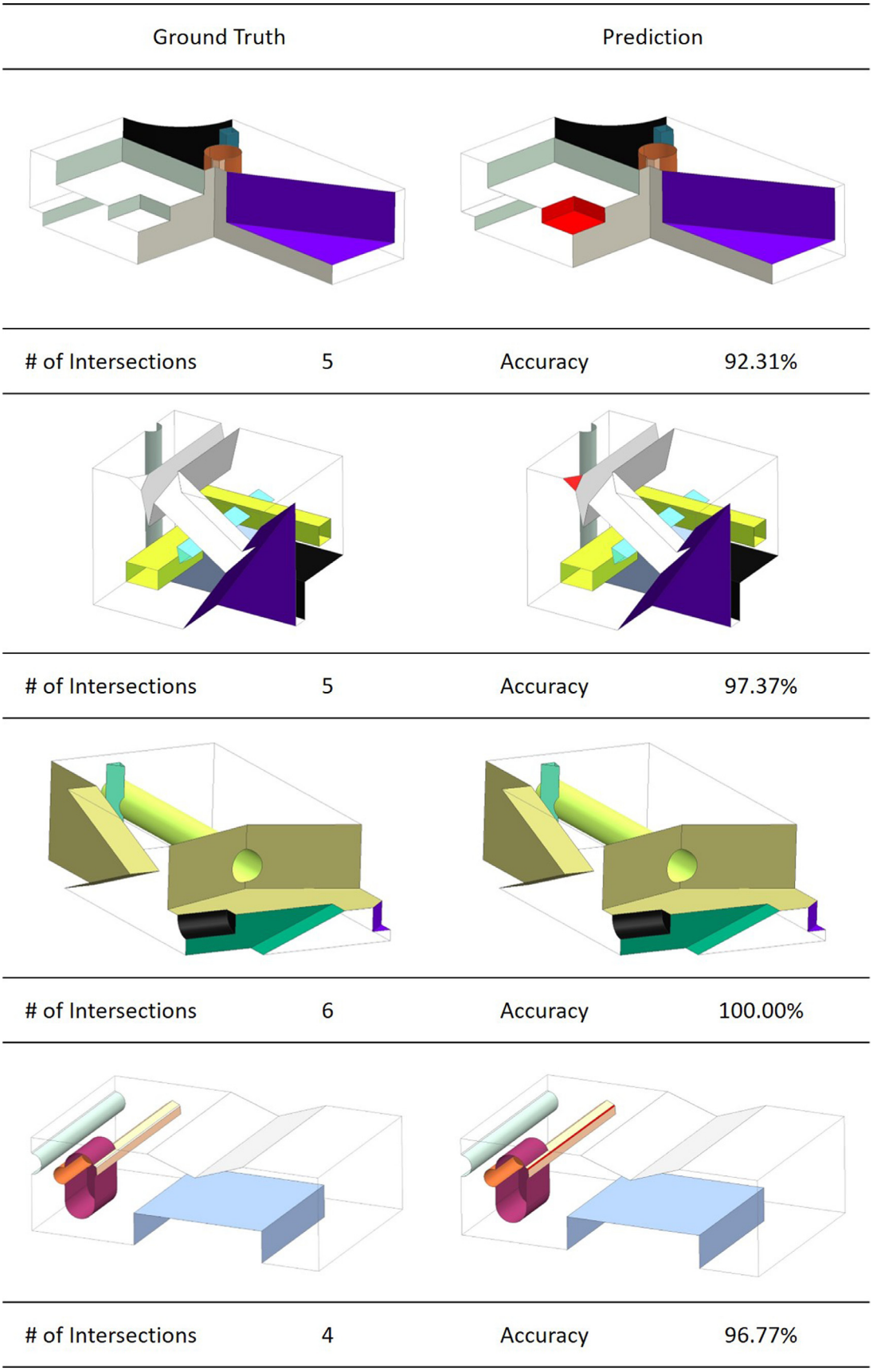


Fig. 19. Examples of classification of intersecting machining features from the test set of the MFCAD++ dataset using Hierarchical CADNet (Edge) (failed B-Rep face predictions are colored in red).. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 6
Effects of using dropout & batch normalization.

Network	Test accuracy (%)
Hierarchical CADNet (Edge)	97.37
No Dropout	96.83
No Batch Normalization	1.81
No Residual Connections	95.70

dramatically decreases by 22.30% once the B-Rep level is removed and by proxy the B-Rep face topology information. This highlights the benefits of learning the information for the machining feature recognition task and Hierarchical CADNet's ability to learn the face topology. It also highlights the fact that more performance could be gained from improving the mesh facet graph level and more information could be extracted from the geometry.

C.3. Dropout, batch normalization and residual connections

In this section, reasoning behind using dropout, batch normalization and residual connections are shown. The results are shown in Table 6, where each network uses edge convexity information and 6 graph convolutional layers on each network level. Dropout is used within the Hierarchical CADNet architecture to add regularization allowing for the trained network to better generalize for the test set and new unseen data. Without dropout, the training accuracy of the network is 1.64% higher than with, though this higher accuracy did not translate to the test set where the accuracy of the network with dropout is 0.53% higher.

Although, batch normalization has a slight regularization effect, the main reason it is used within the network architecture is to help stabilize the learning process. By removing the batch normalization from the network architecture, it can be seen in Table 6, that Hierarchical CADNet fails to learn anything.

Without the use of residual connections, Hierarchical CADNet is more susceptible to the vanishing gradient problem and over-smoothing. This means that the number of graph convolutional layers that can be used within the network architecture is reduced. This is highlighted by the results in Table 6, where without the residual connections, the performance using six graph convolutional layers decreases from 97.37% to 95.70%.

Appendix D. Further details of the datasets

In Figs. 17 and 18, some images of samples from the MFCAD and MFCAD++ datasets are shown respectively.

Fig. 19 gives some examples of intersecting machining features in the test set of the MFCAD++ dataset. The number of times each machining feature intersects with a unique machining feature is given as well as the test accuracies achieved by Hierarchical CADNet using edge convexity information and six graph convolutional layers. Where Hierarchical CADNet has failed to correctly predict the class of a B-Rep face, the B-Rep face has been colored red.

References

[1] CATIA™ 3Dexperience® - dassault systèmes® 3D software n.d. 2019, https://www.3ds.com/products-services/catia/. [Accessed 11 May 2019].
[2] NX n.d. 2021, https://www.plm.automation.siemens.com/global/en/products/nx/. [Accessed 9 January 2020].
[3] Al-wswasi M, Ivanov A, Makatsoris H. A survey on smart automated computer-aided process planning (ACAPP) techniques. Int J Adv Manuf Technol 2018;97:809–32. http://dx.doi.org/10.1007/s00170-018-1966-1.
[4] Babic B, Nesic N, Miljkovic Z. A review of automated feature recognition with rule-based pattern recognition. Comput Ind 2008;59:321–37. http://dx.doi.org/10.1016/J.COMPIND.2007.09.001.
[5] Babić BR, Nešić N, Miljković Z. Automatic feature recognition using artificial neural networks to integrate design and manufacturing: Review of automatic feature recognition systems. Artif Intell Eng Des Anal Manuf AIEDAM 2011;25:289–304. http://dx.doi.org/10.1017/S0890060410000545.
[6] Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Adv Neural Inf Process Syst 2012;25(NIPS 2012):1097–105.
[7] Qi CR, Su H, NieBner M, Dai A, Yan M, Guibas LJ. Volumetric and multi-view CNNs for object classification on 3D data. In: 2016 IEEE Conf. Comput. Vis. Pattern Recognit. IEEE; 2016, p. 5648–56. http://dx.doi.org/10.1109/CVPR.2016.609.
[8] Shi B, Bai S, Zhou Z, Bai X. DeepPano: Deep panoramic representation for 3-D shape recognition. IEEE Signal Process Lett 2015;22:2339–43. http://dx.doi.org/10.1109/LSP.2015.2480802.
[9] Wu Zhirong, Song S, Khosla A, Yu Fisher, Zhang Linguang, Tang Xiaoou, et al. 3D ShapeNets: A deep representation for volumetric shapes. In: 2015 IEEE Conf. Comput. Vis. Pattern Recognit. IEEE; 2015, p. 1912–20. http://dx.doi.org/10.1109/CVPR.2015.7298801.
[10] Maturana D, Scherer S. Voxnet: A 3D convolutional neural network for real-time object recognition. In: 2015 IEEE/RSJ Int. Conf. Intell. Robot. Syst. IEEE; 2015, p. 922–8. http://dx.doi.org/10.1109/IROS.2015.7353481.
[11] Sedaghat N, Zolfaghari M, Amiri E, Brox T. Orientation-boosted Voxel nets for 3D object recognition. 2016.
[12] Kim S, gun Chi H, Ramani K. Object synthesis by learning part geometry with surface and volumetric representations. Comput Des 2021;130:102932. http://dx.doi.org/10.1016/J.CAD.2020.102932.
[13] Riegler G, Ulusoy AO, Geiger A. OctNet: Learning deep 3D representations at high resolutions. In: 2017 IEEE Conf. Comput. Vis. Pattern Recognit. IEEE; 2017, p. 6620–9. http://dx.doi.org/10.1109/CVPR.2017.701.
[14] Wang P-S, Liu Y, Guo Y-X, Sun C-Y, Tong X. O-CNN: Octree-based convolutional neural networks for 3D shape analysis. ACM Trans Graph 2017;36:1–11. http://dx.doi.org/10.1145/3072959.3073608.
[15] Wang P, Sun C, Liu Y, Tong X, Wang P-S, Sun C-Y, et al. Adaptive O-CNN: A patch-based deep representation of 3D shapes ACM reference format. ACM Trans Graph 2018;37. http://dx.doi.org/10.1145/3272127.3275050.
[16] Charles RQ, Su H, Kaichun M, Guibas LJ. PointNet: Deep learning on point sets for 3D classification and segmentation. In: 2017 IEEE Conf. Comput. Vis. Pattern Recognit. IEEE; 2017, p. 77–85. http://dx.doi.org/10.1109/CVPR.2017.16.
[17] Li CRQ, Hao Y, Leonidas S, Guibas J. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In: Conf. Neural Inf. Process. Syst.. 2017, 2017.
[18] Wang Y, Sun Y, Liu Z, Sarma SE, Bronstein MM, Solomon JM. Dynamic graph CNN for learning on point clouds. ACM Trans Graph 2018;38:146.
[19] Hanocka R, Hertz A, Fish N, Giryes R, Fleishman S, Cohen-Or D. MeshCNN: A network with an edge. 2018, http://dx.doi.org/10.1145/3306346.3322959.
[20] Feng Y, Feng Y, You H, Zhao X, Gao Y. MeshNet: Mesh neural network for 3D shape representation. Proc AAAI Conf Artif Intell 2019;33:8279–86. http://dx.doi.org/10.1609/aaai.v33i01.33018279.
[21] Yang J, Mo K, Lai Y-K, Guibas LJ, Gao L. DSG-net: Learning disentangled structure and geometry for 3D shape generation. 2021.
[22] Masci J, Boscaini D, Bronstein MM, Vandergheynst P. Geodesic convolutional neural networks on Riemannian manifolds. In: Proc IEEE Int Conf Comput Vis, 2015-February. 2015, p. 832–40.
[23] Boscaini D, Masci J, Rodolà E, Bronstein MM. Learning shape correspondence with anisotropic convolutional neural networks. Adv Neural Inf Process Syst 2016;3197–205.
[24] Boscaini D, Masci J, Rodolà E, Bronstein MM, Cremers D. Anisotropic diffusion descriptors. Comput Graph Forum 2016;35:431–41. http://dx.doi.org/10.1111/cgf.12844.
[25] Sunil VB, Pande SS. Automatic recognition of machining features using artificial neural networks. Int J Adv Manuf Technol 2009;41:932–47. http://dx.doi.org/10.1007/s00170-008-1536-z.
[26] Onwubolu GC. Manufacturing features recognition using backpropagation neural networks. J Intell Manuf 1999;10:289–99. http://dx.doi.org/10.1023/a:1008904109029.
[27] Zhang Z, Jaiswal P, Rai R. FeatureNet: Machining feature recognition based on 3D convolution neural network. Comput Des 2018;101:12–22. http://dx.doi.org/10.1016/J.CAD.2018.03.006.
[28] Peddireddy D, Fu X, Wang H, Joung BG, Aggarwal V, Sutherl JW, et al. Deep learning based approach for identifying conventional machining processes from CAD data. Procedia Manuf 2020;48:915–25. http://dx.doi.org/10.1016/J.PROMFG.2020.05.130.
[29] Ning F, Shi Y, Cai M, Xu W. Part machining feature recognition based on a deep learning method. J Intell Manuf 2021. http://dx.doi.org/10.1007/s10845-021-01827-7.
[30] Ma Y, Zhang Y, Luo X. Automatic recognition of machining features based on point cloud data using convolution neural networks. In: ACM Int. Conf. Proceeding Ser. Association for Computing Machinery; 2019, p. 229–35. http://dx.doi.org/10.1145/3349341.3349407.
[31] Shi P, Qi Q, Qin Y, Scott PJ, Jiang X. A novel learning-based feature recognition method using multiple sectional view representation. J Intell Manuf 2020;31:1291–309. http://dx.doi.org/10.1007/s10845-020-01533-w.

[32] Cao W, Robinson T, Hua Y, Boussuge F, Colligan AR, Pan W. Graph representation of 3D CAD models for machining feature recognition with deep learning. In: 11A 46th Des. Autom. Conf. American Society of Mechanical Engineers; 2020, http://dx.doi.org/10.1115/DETC2020-22355.
[33] Jayaraman PK, Sanghi A, Lambourne J, Davies T, Shayani H, Morris N. UV-Net: Learning from curve-networks and solids. In: IEEE Conf. Comput. Vis. Pattern Recognit. 2021, p. 11703–12, arXiv.
[34] Lambourne Autodesk Research Karl DD, Willis JG, Kumar Jayaraman P, Sanghi A, Meltzer P, Shayani H. BRepNet: A topological message passing system for solid models. In: Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. 2021, p. 12773–82.
[35] Jones B, Hildreth D, Chen D, Baran I, Kim VG, Schulz A. Automate: A dataset and learning approach for automatic mating of CAD assemblies. 2021.
[36] STL (sTereoLithography) file format, binary n.d. 2021, https://www.loc.gov/preservation/digital/formats/fdd/fdd000505.shtml.. [Accessed 8 January 2021].
[37] Rahman MS. Basic graph theory. Cham: Springer International Publishing; 2017, http://dx.doi.org/10.1007/978-3-319-49475-3.
[38] Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS. A comprehensive survey on graph neural networks. IEEE Trans Neural Netw Learn Syst 2019;32:4–24. http://dx.doi.org/10.1109/tnnls.2020.2978386.
[39] Such FP, Sah S, Dominguez M, Pillai S, Zhang C, Michael A, et al. Robust spatial filtering with graph convolutional neural networks. IEEE J Sel Top Signal Process 2017;11:884–96. http://dx.doi.org/10.1109/JSTSP.2017.2726981.
[40] Li M, Perrier E, Xu C. Deep hierarchical graph convolution for election prediction from geospatial census data. In: 33rd AAAI Conf. Artif. Intell. Vol. 33. AAAI Press; 2019, p. 647–54. http://dx.doi.org/10.1609/aaai.v33i01.3301647.
[41] Li Q, Han Z, Wu X-M. Deeper insights into graph convolutional networks for semi-supervised learning. In: 32nd AAAI Conf Artif Intell AAAI 2018. 2018, p. 3538–45.
[42] Li G, Muller M, Thabet A, Ghanem B. DeepGCNs: Can GCNs go as deep as CNNs? In: Proc. IEEE Int. Conf. Comput. Vis., 2019- October. Institute of Electrical and Electronics Engineers Inc.; 2019, p. 9266–75. http://dx.doi.org/10.1109/ICCV.2019.00936.
[43] Koch S, Matveev A, Jiang Z, Williams F, Artemov A, Burnaev E, et al. ABC: A big CAD model dataset for geometric deep learning. 2018.
[44] Kim S, gun Chi H, Hu X, Huang Q, Ramani K. A large-scale annotated mechanical components benchmark for classification and retrieval tasks with deep neural networks. In: Proc. 16th Eur. Conf. Comput. Vis. 2020.
[45] MFCAD: A dataset of 3D CAD models with machining feature labels. n.d. 2021, https://github.com/hducg/MFCAD. [Accessed 14 June 2021].
[46] PythonOCC: Python package for 3D CAD/BIM/PLM/CAM n.d. 2021, https://github.com/tpaviot/pythonocc-core. [Accessed 18 June 2021].
[47] Numba: A high performance python compiler n.d. 2021, http://numba.pydata.org/. [Accessed 18 June 2021].
[48] Tensorflow n.d. 2020, https://www.tensorflow.org/. [Accessed 18 December 2020].
[49] Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. In: JMLR Workshop and Conference Proceedings. 2010, p. 249–56, J. Mach. Learn. Res., 9.
[50] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: 32nd Int. Conf. Mach. Learn. ICML 2015, 1. International Machine Learning Society (IMLS); 2015, p. 448–56.
[51] Srivastava N, Hinton G, Krizhevsky A, Salakhutdinov R. Dropout: A simple way to prevent neural networks from overfitting. J Mach Learn Res 2014;15:1929–58.
[52] Kingma DP, Ba JL. Adam: A method for stochastic optimization. In: 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc. International Conference on Learning Representations, ICLR; 2015.
[53] Categorical cross entropy n.d. 2021, https://www.tensorflow.org/api_docs/python/tf/keras/losses/CategoricalCrossentropy. [Accessed 18 June 2021].
[54] Lin M, Chen Q, Yan S. Network in network. In: 2nd Int. Conf. Learn. Represent. ICLR 2014 - Conf. Track Proc. International Conference on Learning Representations. ICLR; 2014.
[55] Imagenet n.d. 2019, http://www.image-net.org/. [Accessed 16 May 2019].
[56] Precision-recall — scikit-learn 0.24.2 documentation n.d. 2021, https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html. [Accessed 28 June 2021].
[57] Colligan AR, Robinson TT, Nolan DC, Hua Y, Cao W. MFCAD++ Dataset n.d, https://doi.org/10.17034/d1fec5a0-8c10-4630-b02e-b92dc81df823.