

BRepNet: A topological message passing system for solid models

Joseph G. Lambourne
Autodesk Research

Karl D.D. Willis
Autodesk Research

Pradeep Kumar Jayaraman
Autodesk Research

Aditya Sanghi
Autodesk Research

Peter Meltzer
UCL, Computer Science

Hooman Shayani
Autodesk Research

Abstract

Boundary representation (B-rep) models are the standard way 3D shapes are described in Computer-Aided Design (CAD) applications. They combine lightweight parametric curves and surfaces with topological information which connects the geometric entities to describe manifolds. In this paper we introduce BRepNet, a neural network architecture designed to operate directly on B-rep data structures, avoiding the need to approximate the model as meshes or point clouds. BRepNet defines convolutional kernels with respect to oriented coedges in the data structure. In the neighborhood of each coedge, a small collection of faces, edges and coedges can be identified and patterns in the feature vectors from these entities detected by specific learnable parameters. In addition, to encourage further deep learning research with B-reps, we publish the Fusion 360 Gallery segmentation dataset. A collection of over 35,000 B-rep models annotated with information about the modeling operations which created each face. We demonstrate that BRepNet can segment these models with higher accuracy than methods working on meshes, and point clouds.

1. Introduction

Boundary representation (B-rep) models are the de facto standard for describing 3D objects in commercial Computer Aided Design (CAD) software. They consist of collections of trimmed parametric surfaces along with the adjacency relationships between them [44]. Prismatic shapes can be represented using lightweight primitive curves and surfaces while free-form objects can be defined using NURBS [33]. Although this makes the representation both compact and expressive, the complexity of the data structures and limited availability of labelled datasets has presented a high barrier to entry for researchers.

The problem of segmenting B-rep models, based on

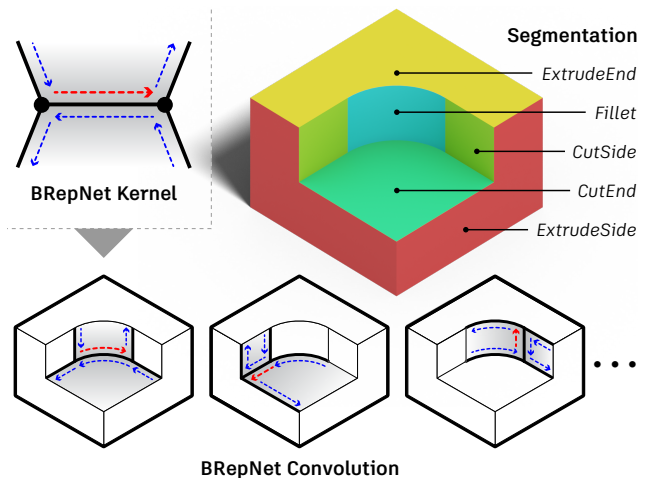


Figure 1: BRepNet convolutional kernels are defined with respect to topological entities called *coedges* (dashed arrows). Feature vectors from a small collection of faces (grey), edges (black) and coedges (blue) adjacent to each coedge (red) are multiplied by the learnable parameters in the kernel. The hidden states arising from the convolution can then be pooled to perform face segmentation.

learned patterns, is of particular interest as it allows the automation of many laborious manual tasks in CAD, Computer Aided Engineering (CAE) and Computer Aided Process Planning (CAPP) [6, 45, 1, 38]. Currently these require a user to repeatedly select groups of faces and/or edges as input for the modeling or manufacturing operation. Examples include model simplification in preparation for finite element analysis [12] and segmenting a model according to the manufacturing process or machining toolpath strategy required to make the object [1, 45].

In addition, parametric feature history is often lost when models are exchanged between different CAD applications [23] and many commercial CAD systems use segmentation algorithms to recover this information [5, 13].

Although attempts were made in the 90s to apply neural networks to the task of B-rep segmentation [19, 11, 30, 14, 41, 38], the absence of machine learning frameworks and large labelled datasets caused progress to stall until very recently [20, 10]. In this paper we introduce BRepNet, a novel neural network architecture designed specifically to operate directly on the faces and edges of B-rep data structures and take full advantage of the topological relationships between them. In addition, we hope to revitalize interest in the problem of B-rep segmentation with the publication of the *Fusion 360 Gallery* segmentation dataset. For the first time we provide a collection of over 35,000 3D models, in multiple representations, annotated with segmentation labels revealing the modeling operations used to create them.

The BRepNet approach is motivated by the observation that in convolutional neural networks for image processing, the weights operate on pixels with known locations within the filter window. A similar arrangement can be achieved with B-reps, where a small collection of faces, edges and coedges can be identified at well defined locations relative to each coedge in the data structure (see Figure 1). Feature vectors can be extracted from these neighbouring entities and concatenated in a known order, allowing convolution to take place as a matrix/vector multiplication [18, 21]. As in image convolution, specific entities relative to each coedge map to specific learnable parameters in our convolutional kernels, allowing patterns in the input data to be easily recognized [32, 9]. The key contributions are as follows:

- We introduce BRepNet, a network architecture using a novel convolution technique which takes full advantage of the topological information the B-rep stores.
- We publish the *Fusion 360 Gallery* segmentation dataset that contains over 35,000 segmented 3D models in B-rep, mesh, and point cloud format.
- We provide experimental results on the *Fusion 360 Gallery* segmentation task, including ablation studies and comparisons to other representations and methods.

Our results demonstrate that direct use of B-rep data solves the *Fusion 360 Gallery* segmentation problem with higher performance and parameter efficiency than other techniques based on point cloud and mesh representations.

2. Related work

Historically, the task of B-rep segmentation has focused on the detection of form features (connected regions of a model with a characteristic shape or pattern with some significance [38]). Feature detection has been an active area of research since the mid 1970s [19], with a range of different heuristic approaches investigated [25, 40, 3, 22, 37, 29].

Early neural networks. Neural networks were first employed by Prabhakar *et al.* [34] with a number of extensions and refinements made over the years [31, 14, 41]. In these early works the B-rep structure is first converted to a face adjacency graph with node features extracted from the B-rep faces and attributes for the arcs extracted from the B-rep edges. Heuristics are then used to break the graph into small connected components which are passed to the networks individually. These techniques were limited by the computer power of the time and so the networks see only a small part of the B-rep at once.

Voxels. Feature detection methods based on voxels [7, 46] offer some advantages for manufacturability analysis, however the cubic storage complexity puts severe limitations on the size of geometric features which can be detected. As CAD models often contain small but important features, the applicability of these techniques with current GPU hardware is quite limited.

Point clouds. Point cloud segmentation has shown excellent results in recent years [35, 36, 43], but typically requires a large number of points to be uniformly sampled from the (B-rep) objects' surface. Faces with small areas can easily be under-sampled and incorrectly classified.

Meshes. Triangle meshes are another important representation for 3D objects, and a number of authors have proposed convolution strategies which operate on them [42, 18, 9, 28]. MeshCNN [18] operates on the edges of a triangle mesh with convolution carried out by aggregating information from the five edges of two adjacent triangles onto the central edge. Liu *et al.* [28] introduce a convolution scheme which operates on directed triangle edges and use it to generate neural network conditioned subdivision surfaces. Although the data structures for triangle meshes are simple, converting B-reps to high quality manifold meshes requires special meshing procedures. By working directly on the original B-rep topology we can avoid the requirement to generate good quality meshes and operate directly on a more compact representation.

Graphs. B-rep model segmentation can also be viewed as a node classification problem on graphs. Two concurrent unpublished works have applied graph convolution approaches to B-rep segmentation [20, 10]. Jayaraman *et al.* [20] uses convolution layers to create input features from grids of 3D points and normal vectors, while Cao *et al.* [10] uses only planar faces which can be directly represented as feature vectors of length 4. In both cases the B-rep data structure is translated to a face adjacency graph which causes some information about relative topological locations of nearby entities to be lost.

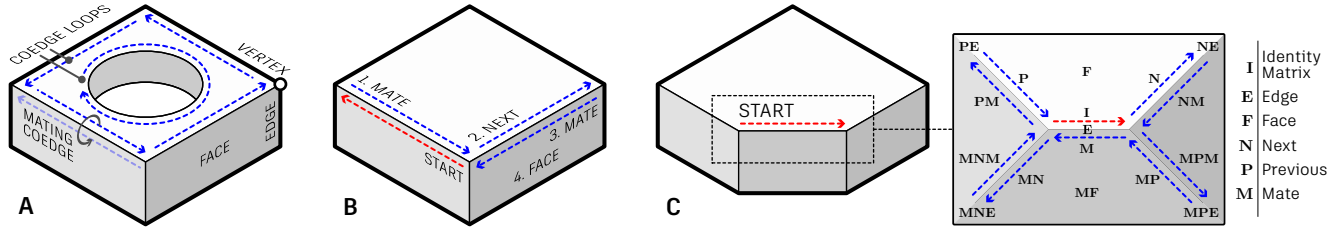


Figure 2: A) B-rep topology comprises faces, edges, loops, coedges and vertices. B) Starting from a given coedge (red), the topology can be traversed by following a sequence of instructions which indicate which entity to move to in the next hop. The instruction sequence $\{mate, next, mate, face\}$ is illustrated. C) The walks from the red coedge to some neighbouring entities are described in terms of products of the incidence matrices \mathbf{N} , \mathbf{P} , \mathbf{M} , \mathbf{F} and \mathbf{E} .

3. Method

3.1. B-rep data structures

Industrial CAD packages have internal data structures which are similar to the partial entity structure described by Lee *et al.* [27]. These structures support the modeling of 2-dimensional manifolds, 3-dimensional volumes and even non-manifold complexes which can arise as intermediate states in boolean operations [44].

A B-rep comprises of faces, edges, loops, coedges and vertices (Figure 2a). A face is a connected region of the model’s surface which may have internal holes [15]. An edge defines the curve where two faces meet and a vertex defines the point where edges meet. Faces have an underlying parametric surface which is divided into visible and hidden regions by a series of boundary loops.

Each loop consists of a doubly linked list of directed edges called coedges, topological entities which are used to represent the adjacency relationships in the B-rep [27]. A coedge stores pointers to the next and previous coedge in the loop, its adjacent or “mating” coedge, its parent face and parent edge. In this work we consider only closed and manifold B-reps where each coedge has exactly one mating coedge, providing sufficient information for the edges in the structure to be traversed in the same way as in the winged edge [8] and QuadEdge [17] data structures.

3.2. Topological walks

By following the pointers which the coedges store, we can walk from a given coedge on the B-rep to entities in its neighborhood. The choice of which pointer to follow at each hop can be thought of as a sequence of instructions which will take us from some starting coedge to a destination coedge. From there we can optionally make one final jump to its owner edge or face. This sequence of instructions defines a topological walk.

An example of a simple topological walk for the instruction sequence: $\{mate, next, mate, face\}$ is shown in Figure 2b. The starting coedge is shown in red and the coedges

traversed during the walk are shown in blue.

For a set of B-rep faces $\mathbf{f} = \{f_1, f_2, \dots, f_{|f|}\}$, edges $\mathbf{e} = \{e_1, e_2, \dots, e_{|e|}\}$, and coedges $\mathbf{c} = \{c_1, c_2, \dots, c_{|c|}\}$, geometric information can be extracted and used to build three input feature matrices $\mathbf{X}^f \in \mathbb{R}^{|f| \times p}$, $\mathbf{X}^e \in \mathbb{R}^{|e| \times q}$ and $\mathbf{X}^c \in \mathbb{R}^{|c| \times r}$ for the face features, edge features and coedge features respectively as described in Section 3.3.

The next, previous and mating adjacency relationships between coedges can be written as three matrices:

$$\mathbf{N}, \mathbf{P}, \mathbf{M} \in \{0, 1\}^{|c| \times |c|} \quad (1)$$

Here $\mathbf{N}_{ij} = 1$ indicates that c_j is the next coedge in the loop from c_i and $\mathbf{M}_{ij} = 1$ when coedge c_j is the mate of coedge c_i . As each coedge has exactly one next, previous and mating coedge, these matrices simply define permutations on the list of coedges in the B-rep. Also we can see that $\mathbf{P} = \mathbf{N}^{-1} = \mathbf{N}^T$. A matrix defining a topological walk between two coedges can then be built by multiplying \mathbf{N} , \mathbf{P} and \mathbf{M} in the sequence in which the *next*, *previous* and *mate* instructions appear in the walk (Figure 2c).

The relationships between a coedge and its parent face and parent edge can also be represented using incidence matrices $\mathbf{F} \in \{0, 1\}^{|c| \times |f|}$ and $\mathbf{E} \in \{0, 1\}^{|c| \times |e|}$. Here $\mathbf{F}_{ij} = 1$ indicates that coedge c_i is in a loop around face f_j and $\mathbf{E}_{ij} = 1$ indicates that coedge c_i belongs to edge e_j .

The transform $\Psi = \mathbf{F}\mathbf{X}^f$ allow us to construct a matrix $\Psi \in \mathbb{R}^{|c| \times p}$ by copying the i th row of the matrix of face features \mathbf{X}^f to the j th row of Ψ for each coedge c_j with parent face f_i . The matrix \mathbf{E} works in a similar way for edges. Topological walks which terminate on faces or edges can then be represented in matrix form by multiplying the matrix for the walk over the coedges by \mathbf{E} or \mathbf{F} .

3.3. Input feature extraction

Geometric feature information from the faces, edges and coedges of the B-rep are passed into the network in the feature matrices \mathbf{X}^f , \mathbf{X}^e and \mathbf{X}^c . One approach to the extraction of geometric features from B-rep faces is given by [20], where grids of points and normal vectors are sampled

from the parametric surface geometry and compressed into feature vectors using a CNN. In this work we investigate whether the *Fusion 360 Gallery* segmentation problem can be solved without providing the network with any coordinate information, instead using only a small amount of extremely concise information from the B-rep data structure. Using coordinate free input features has the advantage that they are invariant to translation and rotation and protects the intellectual property of CAD operators by not revealing the model geometry, while still allowing the network to perform useful tasks.

For face features, the network is given a one-hot vector encoding of the possible surface types (plane, cylinder, cone, sphere, torus). One additional value is used to indicate a rational NURBS surface [33]. In the case of non-rational B-splines all these values will be zero. We also provide the network with the area of each face. For edge features we provide a one-hot vector encoding of the possible kinds of edge geometry (line, circle, ellipse, helix, intersection curve). We encode edge convexity in three one-hot values (concave edge, convex edge, smooth edge). One additional flag indicates if an edge forms a closed loop. Finally the edge length is added. For coedges, the network is passed a single flag indicating whether the direction of the coedge is the same as the direction of the parametric curve of the edge. The input features are standardized over the training set and the same scaling applied to the validation and test sets. More detail is in the supplementary material.

3.4. Convolution

Convolutional kernels in BRepNet are defined relative to the coedges of the B-rep. As noted by Lui *et al.* [28], because coedges are directed this removes the ambiguity between the faces to the left and right of a coedge and avoids the need to aggregate features using symmetric functions as in [18]. The relative topological locations between a starting coedge and the faces, edges and coedges which will take part in a convolution are defined by topological walks. Each walk can be expressed in matrix form by multiplying the matrices \mathbf{N} , \mathbf{P} , \mathbf{M} , \mathbf{F} and \mathbf{E} in the order in which the *next*, *previous*, *mate*, *face* or *edge* instructions must be executed. An example of a collection of faces, edges and coedges which can be used in a BRepNet kernel is shown in Figure 2c. The products of the matrices required to reach each of the destination entities are marked. The matrices encoding the walks to faces, edges and coedges are arranged in three lists K^f , K^e and K^c receptively.

A forward pass through the network proceeds as follows. We start by initialising the matrices $\mathbf{H}_f^{(0)} = \mathbf{X}^f$, $\mathbf{H}_e^{(0)} = \mathbf{X}^e$ and $\mathbf{H}_c^{(0)} = \mathbf{X}^c$. These three matrices are then passed through a number of the convolution units as shown in Figure 3. Following convolution unit t , the hidden state matrices $\mathbf{H}_f^{(t)}$, $\mathbf{H}_e^{(t)}$ and $\mathbf{H}_c^{(t)}$ are generated. The width of

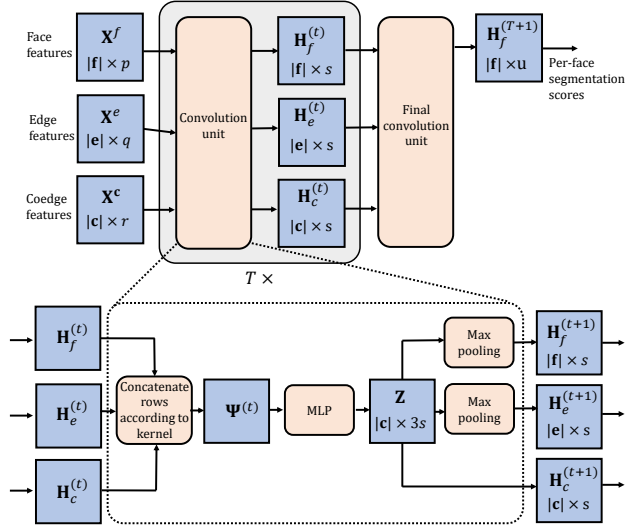


Figure 3: The BRepNet network architecture. Input feature vectors from faces, edges and coedges are passed through a stack of T convolution units to generate hidden states $\mathbf{H}_f^{(t)}$, $\mathbf{H}_e^{(t)}$ and $\mathbf{H}_c^{(t)}$. A final convolution unit generates only the segmentation scores for faces.

these hidden states is defined by a hyper-parameter s . For face classification tasks a final convolution unit generates only matrix $\mathbf{H}_f^{(T+1)} \in \mathbb{R}^{|f| \times u}$ which are the per-face segmentation scores for each of the u classes.

Inside each convolution unit three processes take place. First we build up a matrix Ψ where

$$\begin{aligned} \Psi^f &= \begin{matrix} |K^f| \\ \parallel \\ K_i^f \mathbf{H}_f^{(t)} \\ i=1 \end{matrix} & \Psi^e &= \begin{matrix} |K^e| \\ \parallel \\ K_i^e \mathbf{H}_e^{(t)} \\ i=1 \end{matrix} \\ \Psi^c &= \begin{matrix} |K^c| \\ \parallel \\ K_i^c \mathbf{H}_c^{(t)} \\ i=1 \end{matrix} & \Psi &= \Psi^f \parallel \Psi^e \parallel \Psi^c \end{aligned} \quad (2)$$

This procedure populates the i th row of Ψ with the concatenated hidden state vectors of the entities defined by the kernel with starting coedge c_i .

Each row of Ψ is then passed through a multi-layer perceptron (MLP) with parameters $\Theta^{(t)}$ and ReLU nonlinearities. The input to the first layer of the MLP depends on the number of columns of Ψ while all other MLP layers have a size 3 times the width of the hidden states s .

Following the MLP, we generate a matrix $\mathbf{Z} \in \mathbb{R}^{|c| \times 3s}$. The rows of \mathbf{Z} are associated with coedges in the B-rep. A simple architecture would pass the single matrix \mathbf{Z} to the subsequent convolution units, however we observe that this simple approach gives poor performance on B-rep models where faces have multiple loops (e.g. a face with a hole). In this case the edges of the B-rep do not form a connected graph and information cannot flow between the loops of

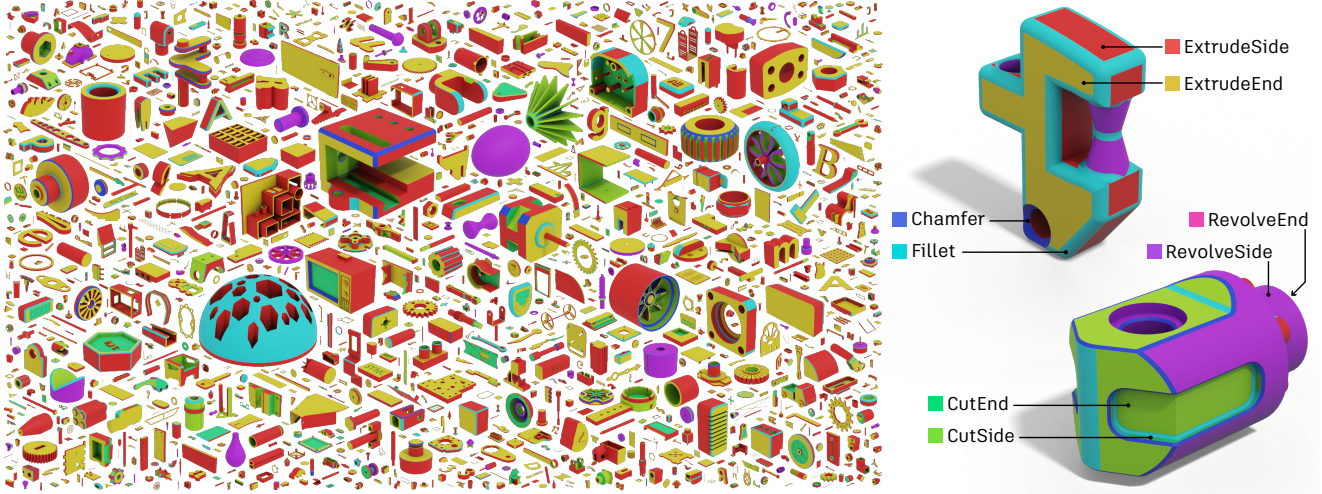


Figure 4: An overview of 3D models from the Fusion 360 Gallery segmentation dataset (left). Each 3D model is labeled according to the CAD modeling operations used to create it (right).

multi-loop faces. The performance of the network is greatly enhanced by pooling information from the coedges onto their parent faces and edges in each convolution unit. This allows information to flow from the coedges in one loop onto the parent face, making it accessible to coedges in another loop in subsequent layers. To apply this pooling the matrix \mathbf{Z} is first split into 3 sub-matrices of size $|c| \times s$.

$$\mathbf{Z} = \begin{bmatrix} \mathbf{H}_c^{(t+1)} & \mathbf{Z}^f & \mathbf{Z}^e \end{bmatrix} \quad (3)$$

$\mathbf{H}_c^{(t+1)}$ is the matrix of hidden states for the coedges in the next layer, which requires no further processing. To build the i th row of the matrix $\mathbf{H}_f^{(t+1)}$ we apply element wise max pooling over the rows of \mathbf{Z}^f corresponding to the coedges with parent face f_i . The matrix $\mathbf{H}_e^{(t+1)}$ is built in a similar way by max pooling the pairs of rows of \mathbf{Z}^e corresponding to coedges with the same parent edge.

A diagram showing the matrices and operations performed in each convolution unit are shown in Figure 3.

3.5. Face classifications

The per-face segmentation scores for each class u_i can then be calculated as follows. In the final convolution unit, the last layer of the MLP has just $|u|$ neurons and produces only the matrix $\mathbf{Z}^f \in \mathbb{R}^{|c| \times |u|}$. The matrix of segmentation scores, $\mathbf{H}_f^{(T+1)} \in \mathbb{R}^{|f| \times |u|}$, is then built by pooling the coedge feature vectors onto their parent faces as before. A cross-entropy loss can then be used to train the network.

4. Fusion 360 Gallery segmentation dataset

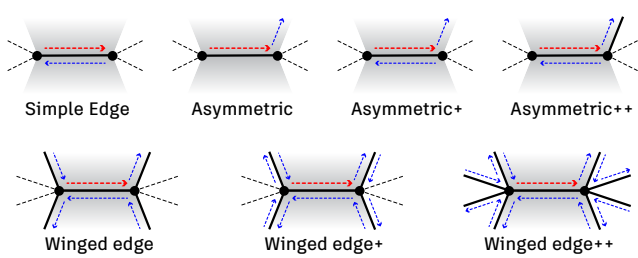
In this section we introduce, to our knowledge for the first time, a dataset containing segmentation information

for B-rep models and the corresponding triangle meshes and point clouds. The *Fusion 360 Gallery* segmentation dataset is produced from designs submitted by users of the CAD package Autodesk Fusion 360 and is segmented based on the CAD modeling operations used to create each face. This modeling history information is not available in existing datasets [24, 47, 2] and goes beyond *what* was designed, providing insights into *how* people design 3D models.

The segmentation dataset contains a total of 35,858 3D models with per-face, per-triangle, and per-point segment labels provided for the B-rep, mesh and point cloud representations (Figure 4, left). For segmentation we use a small subset of the most common CAD modeling operations: *extrude*, *chamfer*, *fillet*, and *revolve*. In order to create a segmentation which contains as much information as possible about the CAD modeling operations, we subdivide extrude operations into additive (i.e. adding) and subtractive (i.e. cutting) extrusion operations, and further divide the faces created by extrude and revolve into side and end faces. This gives a set of eight labels for each face: *ExtrudeSide*, *ExtrudeEnd*, *CutSide*, *CutEnd*, *Fillet*, *Chamfer*, *RevolveSide*, and *RevolveEnd* (Figure 4, right). Further details on the dataset are provided in the supplementary material.

5. Experiments

In this section we perform experiments to examine the following important network capabilities. First we show that loop ordering information is useful for solving a B-rep segmentation problem. We study how performance is affected when the incidence relations in the matrices \mathbf{N} and \mathbf{P} are withheld from our architecture and explore a range of kernel configurations to find which one is optimal. We ana-



Kernel	s	$ \Theta $	Accuracy %	IoU %
Simple edge	120	359k	91.02 ± 0.20	74.03 ± 0.55
Asymmetric	120	359k	91.66 ± 0.17	75.01 ± 0.54
Asymmetric+	113	358k	91.82 ± 0.13	75.06 ± 0.71
Asymmetric++	107	359k	92.05 ± 0.07	75.69 ± 0.38
Winged edge	84	359k	92.52 ± 0.15	77.10 ± 0.54
Winged edge+	75	357k	92.50 ± 0.15	76.86 ± 0.47
Winged edge++	63	358k	92.50 ± 0.12	77.14 ± 0.44
ECC	153	360k	90.36 ± 0.23	72.08 ± 0.50

Figure 5: Different BRepNet kernel configurations (left) for which the accuracy and IoU are compared (right). The accuracy and IoU for the Edge-Conditioned Convolution (ECC) graph network [39] discussed in Section 5.5 is also shown. MLP width s is adjusted to keep the total number of parameters $|\Theta|$ in the network to around 360k.

lyze the features passed to the network, identify the key information used to generate the segmentation and provide insights on why these features are important. To demonstrate the advantages of learning based approaches we compare with a traditional rule-based feature recognition algorithm. Next, we compare BRepNet performance against an Edge-Conditioned Convolution (ECC) graph network [39, 16]. This architecture is chosen as it can ingest the same input features as BRepNet, but employs no special techniques to exploit the manifold nature of the B-rep. As such this comparison shows the gains which can be made when specific kernel weights operate on specific neighbouring nodes.

Finally, as B-rep models can be converted to meshes and point clouds we compare against networks using these representations. We investigate the advantages of working directly with the B-rep data structure and challenges of using approximations to the true geometry.

The data is divided into a 70/15/15% train/validation/test split. In each of the experiments above the networks are trained for 50 epochs and the weights with the smallest validation loss are recorded. The performance of these trained models on the test set is then evaluated. The reported values are the average over 10 runs with different random seeds and the error bars are computed as the standard deviation.

5.1. Evaluation metrics

We use accuracy and intersection over union (IoU) to evaluate network performance. Due to data imbalance in the *Fusion 360 Gallery* segmentation dataset, the IoU metric is useful for providing insight into the performance on the rarer classes. Rather than computing IoU values for individual B-rep models and then averaging as in [35], we consider the entire collection of all faces in the test set at once. This methodology is referred to as “part IoU” in [26] and avoids the special case when a B-rep model has no faces, either predicted or in the ground truth, for a given class.

5.2. Choice of kernel

The BRepNet architecture provides a flexible framework for defining the relative topological locations of the entities which make up a convolutional kernel. Here we study how the choice of these entities affects network performance. Figure 5, left shows the range of different kernel configurations used in the experiments. The corresponding topological walks are included in the supplementary material. As the number of parameters in the MLP is dependent on the number of entities in the kernel, we adjust the hyper-parameter s to keep the total number of network parameters as close as possible to 360k. This decouples the effect of aggregating information from a wider region of the B-rep and the effects of increasing network capacity. For each kernel configuration we train a network with two convolutional units, each with a two layer MLP. Figure 5, right shows the accuracy and IoU for each kernel configuration along with the values of s and the corresponding number of parameters.

The ability of the network to exploit loop ordering information can now be evaluated. The “simple edge” and “asymmetric” kernels are carefully chosen to have the same number of faces, edges and coedges, allowing them to be compared directly without any adjustments in the MLP width. The “simple edge” arrangement contains only an edge and its two adjacent faces and coedges, giving it information similar to a face adjacency graph, but withholding information regarding the order in which coedges are arranged around the loop. The “asymmetric” kernel includes the next coedge in the loop in place of the mating coedge, allowing the kernel to observe patterns like contiguous smooth edges. We observe 0.98% improvement in IoU when moving from the “simple edge” to “asymmetric” kernels. While this improvement is less than 2 standard deviations, a Welch’s unequal variances t-test gives a P value of 0.0012 for this result, indicating that the coedge ordering information is useful for the segmentation task.

The “winged edge” kernel configuration is similar to the *half-flaps* described by Liu *et al.* [28]. It achieves an accuracy of 92.52% and an IoU of 77.10%, over 5 standard

deviations above the IoU value achieved by the “simple edge” kernel. Adding additional entities to the kernel results in only very marginal gains as shown in the table at the right of Figure 5. This can be understood intuitively as the “winged edge” kernel includes a compact set of topological entities immediately adjacent to a given edge. When the kernel is expanded beyond this size, the locations at which the topological walks terminate become dependent on the B-rep topology in the vicinity of the edge. For example the “winged edge++” kernel configuration includes walks like NMN and MPM which will evaluate to the same entity when walking around vertices of valance 3 but distinct entities when the vertex has valance 4 or higher. The “winged edge” kernel lies at a sweet spot containing enough entities to allows patterns in local regions of the B-rep topology to be recognized, while being small enough not to be adversely affected by differences in the topology.

5.3. Ablation studies on input features

Here we identify which of the input features described in Section 3.3 play an important role in the results for the segmentation. The network is trained with groups of input features removed and the resulting IoU values are shown in Figure 6. The “winged edge++” kernel configuration is employed in these experiments and the hyper parameters are as described in Section 5.2. We see that removing the one-hot encodings for surface type reduces IoU by 3.7% and removing curve type information reduces the IoU by 3.9%. These large reductions in performance are expected as the surface and curve type information is the primary way geometric information is fed to the network.

We also observe a 4.6% reduction in IoU when edge convexity is removed. Edge convexity is well known to be useful for the detection of form features and was used in a large number of early neural networks [34, 31, 41, 38]. Joshi *et al.* [22] offers an insight into how edge convexity could be useful with the observation that a face with all convex edges cannot be a part of a concave feature (*CutSide* or *CutEnd*).

Removing other input features have much smaller effects. Without the edge length feature the IoU only decreases by 0.7% and removing the face area feature causes just a 0.4% IoU decrease. Hence we conclude that edge convexity, curve type and surface type are the primary pieces of information used by the network in segmentation.

5.4. Heuristic method comparison

Many CAD modeling packages use rule-based algorithms for the detection of form features. Here we compare against the feature recognition capabilities of Autodesk Shape Manager (ASM) [4], an industry standard CAD kernel used in numerous commercial products.

As ASM cannot detect the *RevolveEnd* segment type, we omit this when computing the ASM average IoU result.

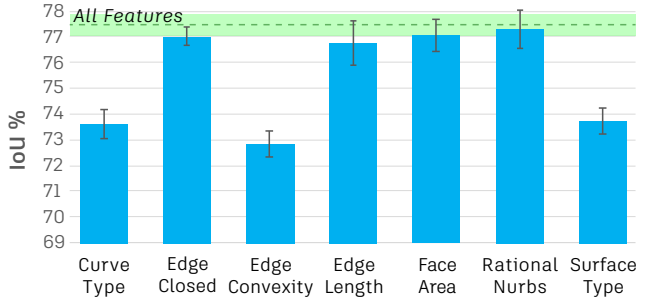


Figure 6: Effect on IoU of removing groups of input features from the network.

ASM does not identify any modeling feature type for 13% of the faces in the dataset and we consider these faces to be incorrectly classified.

The results for ASM feature recognition on the *Fusion 360 Gallery* segmentation task are shown in Table 1. While BRepNet achieves an IoU value 27% higher than ASM, this does not reflect the results qualitatively. When features are recognized by the ASM algorithm, the faces identified are always geometrically consistent with the type of feature found. The confusion is between classes where the actual modeling technique used is ambiguous. For example, a designer may choose to create a cylinder with an extruded circle or a revolved rectangle. The higher accuracy achieved on the classification task by BRepNet shows the network is capable of learning the *most likely* modeling technique a designer will employ rather than simply identifying one of the possible solutions.

5.5. Edge-conditioned convolution graph network

In this section we compare BRepNet performance with an Edge-Conditioned Convolution (ECC) graph network as described in [39]. As discussed in Section 5.3, an important indicator for the class of a face is the convexity of its surrounding edges. As this architecture allows edge attributes to affect the messages passed between faces it is well suited to *Fusion 360 Gallery* segmentation task. The B-rep topology is translated into a face adjacency graph with the faces represented as nodes connected by pairs of directed arcs with opposite orientations. We use the face features X^f as input node features. As the directed arcs map 1:1 with the coedges in the B-rep, we create the attribute vectors for each directed arc by concatenating the corresponding coedge features with the features of its parent B-rep edge.

We match the hyper-parameters of the network to those of BRepNet as closely as possible. Two edge-conditioned convolution layers are used, with the edge specific weight matrices computed by two-layer MLPs. The width of the first MLP input is defined by the number of face features and all subsequent widths were set to 153. This gave the

	Per-face accuracy	Per-face IoU	Per-entity accuracy	Per-entity IoU	$ \Theta $
BRepNet	92.52 ± 0.15	77.10 ± 0.54	-	-	359,100
ECC	90.36 ± 0.23	72.08 ± 0.50	-	-	359,558
ASM	64.57	49.53	-	-	-
PointNet++	74.00 ± 0.84	34.78 ± 2.23	82.78 ± 0.30	36.49 ± 1.67	1,403,784
MeshCNN	62.99 ± 0.37	20.59 ± 0.41	73.81 ± 0.51	24.07 ± 0.42	2,279,720

Table 1: Accuracy, IoU and number of model parameters, $|\Theta|$, for a variety of different networks. The BRepNet results are for the “winged edge” kernel configuration. The per-entity accuracy and IoU columns refer to per-edge accuracy in the case of MeshCNN and per-point accuracy in the case of PointNet++.

ECC network a total of 359,558 parameters which is the closest possible match to BRepNet. The accuracy and IoU of the ECC network are shown in Figure 5 and Table 1. The IoU value is more than 5% below what BRepNet can achieve using the ‘winged edge’ kernel. This is expected as this graph network architecture is not specifically designed for convolution on manifolds and does not map specific learnable parameters to specific entities in the convolution. We would expect to see IoU values approximately equal to what BRepNet achieves with the “simple edge” kernel, but on the test set the ECC gives a 2% lower IoU value. We noticed that BRepNet is more stable than the ECC during training, and we believe the poor performance of the ECC on the test set may be partially due to the choice of epoch for which the trained model was recorded for use at test time. In all experiments the model with the lowest validation loss is used for evaluation on the test set.

5.6. Comparison with geometry based methods

In this section we investigate the advantages of working directly with B-rep models rather than converting the geometry to meshes or point clouds. We generate closed and manifold meshes from the B-rep geometry with close to 3000 triangles edges each. Computing meshes which meet this criteria is itself a difficult task requiring specialized meshing algorithms. For 13% of B-rep models the meshing algorithm failed entirely and the B-rep representations of these models were removed from the dataset. Avoiding the requirement to generate these high quality meshes is a major advantage of working directly on B-rep data.

Point cloud data is then generated by random uniform sampling of 2048 points over the surface of each mesh. As the number of points sampled from each face is determined by area, small faces generate a low number of points. The potential to under-sample some faces is a disadvantage for point cloud techniques as small holes and grooves, which can be critical for the function of the part, may be missed.

Two well-known architectures, PointNet++ [36], and MeshCNN [18] are adapted for the face segmentation task. To compare performance between multiple representations we use per-face classification accuracy and IoU as our primary evaluation metrics. The triangles generated from each

B-rep face inherit the label of that face and points inherit the labels of the triangles from which they were sampled. Triangle edges are considered to be owned by the first of the two triangles sharing the edge and edge labels are derived from the faces which generated the triangles. The per-face accuracy and IoU is then evaluated by averaging the segmentation scores for the points or edges derived from each face. This gives a prediction of the class for each face, from which the accuracy and mean IoU can be evaluated as described in Section 5.1. In addition to the per-face metrics we also report the per-point and per-edge accuracy and IoU for PointNet++ and MeshCNN respectively. As for the per-face metrics, the IoU values are computed by considering the points or edges from all bodies together.

Table 1 details the accuracy and IoU results of the segmentation task along with the number of model parameters. Both BRepNet and the ECC easily outperform the geometry based methods by more than 16% accuracy and 37% in IoU with just under 1/4 the number of parameters. This demonstrates the advantages of working directly with the compact B-rep data rather than derived representations.

6. Conclusions

We have presented BRepNet, a neural network architecture which can operate directly on B-rep models. We also introduced the *Fusion 360 Gallery* segmentation dataset and provide benchmark results on the segmentation problem. Our results show that by using the concise surface and edge type information from the B-rep data structure the network can easily outperform existing techniques using point clouds and meshes on the *Fusion 360 Gallery* dataset segmentation task. In addition we demonstrate that by defining convolutional kernels relative to the coedges of the B-rep, the architecture can make use of information about the next and previous coedges in the loops around faces, giving better performance than an edge conditioned convolution network with the same number of parameters. In future work we plan to apply this convolution scheme to other problems where graphs are embedded in 2D manifolds such as polyhedral models, subdivision surfaces, super-pixel segmentations of image data, region growing algorithms on triangle meshes and for learning tasks on Voronoi diagrams.

References

- [1] Mazin Al-wswasi, Atanas Valev Ivanov, and Harris G. Makatsoris. A survey on smart automated computer-aided process planning (acapp) techniques. *The International Journal of Advanced Manufacturing Technology*, 97:809–832, 2018. **1**
- [2] Atin Angrish, Benjamin Craver, and Binil Starly. “Fab-Search”: A 3D CAD Model-Based Search Engine for Sourcing Manufacturing Services. *Journal of Computing and Information Science in Engineering*, 19(4), 06 2019. 041006. **5**
- [3] S. Ansaldi, K. De Floriani, and B. Falcidienc. An edge-face relational scheme for boundary representations. *Computer Graphics Forum*, 4(4):319–332, 1985. **2**
- [4] Autodesk. Autodesk shapemanager. <https://en.wikipedia.org/wiki/ShapeManager>. **7**
- [5] Autodesk. *Inventor Feature Recognition*, 2012. **1**
- [6] Bojan R. Babic, Nenad Nesic, and Zoran Miljkovic. A review of automated feature recognition with rule-based pattern recognition. *Comput. Ind.*, 59:321–337, 2008. **1**
- [7] Aditya Balu, Kin Gwn Lore, Gavin Young, Adarsh Krishnamurthy, and Soumik Sarkar. A deep 3d convolutional neural network based design for manufacturability framework. *CoRR*, abs/1612.02141, 2016. **2**
- [8] Bruce G. Baumgart. Winged edge polyhedron representation. Technical report, Stanford University, Stanford, CA, USA, 1972. **3**
- [9] Giorgos Bouritsas, Sergiy Bokhnyak, Stylianos Ploumpis, Stefanos Zafeiriou, and Michael Bronstein. Neural 3d mophable models: Spiral convolutional networks for 3d shape representation learning and generation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct 2019. **2**
- [10] Weijuan Cao, Trevor Robinson, Yang Hua, Flavien Bousuge, Andrew R. Colligan, and Wanbin Pan. Graph representation of 3d cad models for machining feature recognition with deep learning. In *Proceedings of the ASME 2020 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, IDETC-CIE. ASME, 2020. **2**
- [11] Y. H. Chen and H. M. Lee. A neural network system feature recognition for two-dimensional. *International Journal of Computer Integrated Manufacturing*, 11(2):111–117, 1998. **2**
- [12] Florence Danglade, Jean-Philippe Pernot, and Philippe Véron. On the use of machine learning to defeature cad models for simulation. *Computer-Aided Design and Applications*, 11(3):358–368, 2014. **1**
- [13] Dassault. *Solidworks FeatureWorks*, 2019. **1**
- [14] Lian Ding and Yong Yue. Novel ann-based feature recognition incorporating design by features. *Computers in Industry*, 55:197–222, 10 2004. **2**
- [15] Charles M Eastman and Kevin J Weiler. Geometric modeling using the euler operators. *12 p. : ill. Pittsburgh: Institute of Physical Planning, Carnegie Mellon University, February, 1979. includes bibliography*, 01 1979. **3**
- [16] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. **6**
- [17] Leonidas Guibas and Jorge Stolfi. Primitives for the manipulation of general subdivisions and the computation of voronoi. *ACM Trans. Graph.*, 4(2):74–123, Apr. 1985. **3**
- [18] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: a network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019. **2, 4, 8**
- [19] Mark Henderson, Gopal Srinath, Roger Stage, Kim Walker, and William Regli. Boundary representation-based feature identification. *Manufacturing Research and Technology*, 20(C):15–38, 1994. **2**
- [20] Pradeep Kumar Jayaraman, Aditya Sanghi, Joseph Lambourne, Thomas Davies, Hooman Shayani, and Nigel Morris. Uv-net: Learning from curve-networks and solids. *arXiv:2006.10211v1*, 2020. **2, 3**
- [21] Y. Jia. *Learning Semantic Image Representations at a Large Scale*. PhD thesis, University of California, Berkeley, 2014. **2**
- [22] S. Joshi and T.C. Chang. Graph-based heuristics for recognition of machined features from a 3d solid model. *Computer-Aided Design*, 20(2):58 – 66, 1988. **2, 7**
- [23] Byung Chul Kim and Soonhung Han. Integration of history-based parametric translators using the automation apis. *Int. J. Product Lifecycle Management*, 2, 01 2007. **1**
- [24] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9601–9611, 2019. **5**
- [25] L.K. Kyprianou. *Shape Classification in Computer-aided Design*. University Library, 1980. **2**
- [26] Eric-Tuan Le, Iasonas Kokkinos, and Niloy J. Mitra. Going deeper with lean point networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. **6**
- [27] Sang Hun Lee and Kunwoo Lee. Partial entity structure: A compact non-manifold boundary representation based on partial topological entities. In *Proceedings of the Sixth ACM Symposium on Solid Modeling and Applications, SMA '01*, page 159–170, New York, NY, USA, 2001. Association for Computing Machinery. **3**
- [28] Hsueh-Ti Derek Liu, Vladimir G. Kim, Siddhartha Chaudhuri, Noam Aigerman, and Alec Jacobson. Neural subdivision. *ACM Trans. Graph.*, 39(4), July 2020. **2, 4, 6**
- [29] Lujie Ma, Zhengdong Huang, and Yanwei Wang. Automatic discovery of common design structures in cad models. *Comput. Graph.*, 34:545–555, 2010. **2**
- [30] M. Marquez, R. Gill, and A. White. Application of neural networks in feature recognition of mould reinforced plastic parts. *Concurrent Engineering*, 7(2):115–122, 1999. **2**

- [31] Konstantinos Nezis and George Vosniakos. Recognizing 212d shape features using a neural network and heuristics. *Computer-Aided Design*, 29(7):523–539, 1997. 2, 7
- [32] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs, 2016. 2
- [33] Les Piegl and Wayne Tiller. *The NURBS Book*. Springer-Verlag, New York, NY, USA, second edition, 1996. 1, 4
- [34] S. Prabhakar and Mark Henderson. Automatic form-feature recognition using neural-network-based techniques on boundary representations of solid models. *CAD Computer Aided Design*, 24(7):381–393, jul 1992. 2, 7
- [35] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 2, 6
- [36] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017. 2, 8
- [37] Hiroshi Sakurai and David C. Gossard. Recognizing shape features in solid models. *IEEE Computer Graphics and Applications*, 10:22–32, 1990. 2
- [38] Yang Shi, Zhang Yicha, Kaishu Xia, and Ramy Harik. A critical review of feature recognition techniques. *Computer-Aided Design and Applications*, 17:861–899, 01 2020. 1, 2, 7
- [39] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017. 6, 7
- [40] S. Staley, Mark Henderson, and D. Anderson. Using syntactic pattern recognition to extract feature information from a solid geometric database. *Computers in Mechanical Engineering*, 2, 09 1983. 2
- [41] V. Sunil and Sanket Pande. Automatic recognition of machining features using artificial neural networks. *The International Journal of Advanced Manufacturing Technology*, 41:932–947, 04 2009. 2, 7
- [42] Pengyu Wang, Yuan Gan, Panpan Shui, Fenggen Yu, Yan Zhang, Songle Chen, and Zhengxing Sun. 3d shape segmentation via shape fully convolutional networks. *Computers & Graphics*, 70:128–139, Feb 2018. 2
- [43] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *CoRR*, abs/1801.07829, 2018. 2
- [44] K.J. Weiler. *Topological structures for geometric modeling*. Technical report RPI, Center for Interactive Computer Graphics. University Microfilms, 1986. 1, 3
- [45] Yong Yue, Lian Ding, Kemal Ahmet, John Painter, and Mick Walters. Study of neural network techniques for computer integrated manufacture. *Engineering Computations*, 19:136–157, 03 2002. 1
- [46] Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *CoRR*, abs/1812.04202, 2018. 2
- [47] Zhibo Zhang, Prakhar Jaiswal, and Rahul Rai. FeatureNet: machining feature recognition based on 3d convolution neural network. *Computer-Aided Design*, 101:12–22, 2018. 5