Computer-Aided Design 166 (2024) 103629

Contents lists available at ScienceDirect

Computer-Aided Design

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

Extending Point-Based Deep Learning Approaches for Better Semantic Segmentation in CAD

Gerico Vidanes [a,*], David Toal [a], Xu Zhang [b], Andy Keane [a], Jon Gregory [c], Marco Nunez [c]

[a] University of Southampton, UK
[b] Falmouth University, UK
[c] Rolls Royce plc., UK

ARTICLE INFO

Keywords:
Point cloud
Deep learning
Feature recognition
Computer-aided design
Graph neural networks

ABSTRACT

Geometry understanding is a core concept of computer-aided design and engineering (CAD/CAE). Deep neural networks have increasingly shown success as a method of processing complex inputs to achieve abstract tasks. This work revisits a generic and relatively simple approach to 3D deep learning – a point-based graph neural network – and develops best-practices and modifications to alleviate traditional drawbacks. It is shown that these methods should not be discounted for CAD tasks; with proper implementation, they can be competitive with more specifically designed approaches. Through an additive study, this work investigates how the boundary representation data can be fully utilised by leveraging the flexibility of point-based graph networks. The final configuration significantly improves on the predictive accuracy of a standard *PointNet++* network across multiple CAD model segmentation datasets and achieves state-of-the-art performance on the *MFCAD++* machining features dataset. The proposed modifications leave the core neural network unchanged and results also suggest that they can be applied to other point-based approaches.

1. Introduction

3D deep learning (DL) continues to mature towards applicability for real use cases that add value to society. An explosion of pioneering works brought countless new methods for processing 3D data in different formats [1–4]. More recently, theoretical frameworks are being developed to consolidate the research landscape [5] and work is being done to utilise these methods in applied science [6,7] and engineering [8–11]. Building on this, the current work focuses on automated feature recognition for computer-aided design (CAD) — commonly formulated as the task of semantic segmentation within the DL field. This capability is a critical building block in computational engineering applications, for example in process planning [12] or design optimisation [13]. While a system for automated feature recognition is useful in itself, it also serves as a stage for developing a DL system that is able to learn internal representations that are useful for arbitrary downstream tasks — commonly referred to as 'backbone' networks [14].

Within CAD, boundary representation (b-rep) models are the *de facto* digital encoding for geometry. However, due to the data structure's complexity, it was not until recently that this 3D data representation has been fully utilised by DL approaches [8–10]. Some methods still use *hand-crafted* feature descriptors to convert the raw b-reps to useable input to the network — this arguably does not fully leverage the high ceiling of deep representation learning [15]. Conversely, other methods apply more geometric approaches to CAD model understanding [1,11,16] - giving hints that the neural networks can learn sufficiently informative representations from the primitive 3D data, and make useful predictions.

An advantage of using a more generic 3D shape representation is that it allows for easier application outside b-rep models. It is much easier to convert a b-rep model to a mesh, or a voxel, or a point cloud representation than it is to reverse engineer a parameterised b-rep model from these simpler representations [17]. While the current work focuses on automated feature recognition in applications that use b-rep models, other domains could benefit from this capability. For instance, automated feature recognition on meshes or point clouds is required for digital product life-cycle management [18,19].

This work investigates whether a simple, geometry-first approach to CAD model understanding can achieve competitive performance compared to approaches which are specifically designed for b-rep processing. A point cloud representation is chosen as the data structure for its simplicity and flexibility — this is further elaborated on in Section 2. Instead of presenting a brand-new neural network architecture for semantic feature recognition of CAD models, an additive study is performed. A baseline approach is revisited and design decisions, both

* Corresponding author.
E-mail address: g.vidanes@soton.ac.uk (G. Vidanes).

https://doi.org/10.1016/j.cad.2023.103629
Received 31 March 2023; Received in revised form 27 August 2023; Accepted 4 October 2023
Available online 10 October 2023

novel and from recent literature, are critically analysed for potential improvements before integration. This is inspired by work from Liu et al. [20] and Qian et al. [21].

The paper is structured as follows. Section 2 reviews related work in the field of automated feature recognition, introduces the framework of graph neural networks, and discusses the purpose of the current work in relation to the field. Section 3 describes the additive study — in terms of the baselines, benchmarks, and training and evaluation methods used. Section 4 forms the bulk of the paper where a single public dataset (*MFCAD++* [10]) is used to build an 'optimal' model configuration. Because of the additive nature of the study, each subsection within Section 4 contains both method descriptions and immediate results before continuing with further system modifications. Finally, Section 5 presents the ultimate results when applying the proposed method extensions on a number of public CAD semantic segmentation datasets [8,10,11] - this is discussed and compared with the results of approaches presented in those works.

2. Literature review

2.1. Background

There has been much work over the past decades around automated feature recognition. This commonly centred around the automated transition between CAD and computer-aided manufacturing (CAM) or in the area of computer-aided process planning (CAPP) more broadly. For a full historical review, see Shah et al. [22]. There also exists the application for automated transition between CAD and simulation. Where the identification and location of features are used to aid in meshing, boundary condition specification, and post-processing [13, 23]. The majority of methods would now be classed as 'algorithmic' or '*expert systems*' — in contrast to modern machine learning methods. Arguably the most prolific of these algorithmic methods are topological or graph-based approaches [24]. A graph is constructed using b-rep faces as nodes and connected by their topological adjacency; these edges are also given attributes based on the convexity of the b-rep edge connecting the faces. Geometric features can then be recognised by matching known sub-graphs.

On the other hand, basic learning-based approaches have also been attempted since the 1990s [25,26]; these have mainly focused on the use of face adjacency graphs together with *hand-crafted* features (shape descriptors) as inputs to fully-connected neural networks. More recently, as large convolutional neural networks (CNN) have dominated computer vision, these ideas have increasingly been applied to feature recognition on 3D shapes. A pioneering work by Zhang et al. [1] introduced a dataset of CAD models each containing a single machining feature to be classified; a voxel-based 3D CNN was also presented for learning on this dataset. This neural network is a classifier – one semantic label per input shape – and therefore a rule-based segmentation algorithm was used to recognise multiple machining features in a part. Subsequent works also employed the rule-based pre-segmentation and neural network classifier combination: Shi et al. [27] used a multi-view approach with 2D CNNs while Yao et al. [16] used a *PointNet++* model as the classifier network.

The aforementioned approaches tend to be limited by the rule-based pre-segmentation step. To build a model which is able to perform semantic segmentation directly, one needs a dataset with these labels for training. A number of such datasets have been publicly released in recent years. The *MFCAD* dataset from Cao et al. [28] was algorithmically generated to create various CAD models containing multiple machining features — each face is labelled with the machining operation which created it. Colligan et al. [10] build on this with *MFCAD++*, which has more intersecting features and attempts to ensure that each generated shape is physically manufacturable. Zhang et al. [11] also make available a separate machining features dataset with similar semantic labels, algorithmically generated with *CATIA*. Apart from machining features, there is also the *Fusion 360 Gallery* segmentation dataset from Lambourne et al. [8] - consisting of a variety of human designed shapes. The b-rep faces are labelled with the CAD modelling operation which was used to create them.

2.2. Geometric deep learning

To better understand the seemingly disparate geometry segmentation approaches in the growing literature, this subsection gives a brief overview of the geometric deep learning framework of Bronstein et al. [5]. In this framework, most real world data can be encoded as a graph. Where a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of nodes $\mathcal{V}$, with some D-dimensional attributes $\mathbf{x} \in \mathbb{R}^D$, connected by a set of edges $\mathcal{E} = \{\mathbf{e} \mid \mathbf{e} \in \mathcal{V} \times \mathcal{V}\}$. In this context, the task of geometry segmentation then becomes node classification — where the nodes are geometric entities like b-rep faces, mesh triangles, or points.

Hierarchical representations which are rich in semantic meaning can be extracted from the graph by the learning system. This is done by repeatedly aggregating the attributes (or latent vectors) of connected nodes. The generalised operator for this, first formalised by Gilmer et al. [29], can be defined as:

$$\mathbf{x}'_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} \psi(\mathbf{x}_u, \mathbf{x}_v)\right) \tag{1}$$

where $\mathbf{x}_u$ is the latent vector of node u and $\mathcal{N}_u$ is the neighbouring nodes of node u, for some definition of neighbourhood. ϕ and ψ are typically differentiable, parameterised functions – like multi-layer perceptrons (MLPs) – and $\bigoplus$ is some symmetric (permutation invariant) function. $\mathbf{x}'_u$ is then the new latent vector of node u after the aggregation operation. This is referred to as the *message-passing* formulation.

Eq. (1) can then be made more specific and less expressive depending on the application. Stacking these aggregations in layers results in a graph neural network (GNN). From this, Bronstein et al. [5] show that one can derive most other DL architectures, even 2D grid CNNs and attentional-based *Transformers*.

2.3. CAD semantic segmentation

With appropriate datasets, models for end-to-end semantic segmentation can be developed. In the current literature, there is no clear consensus on the best 3D representation to pair with DL approaches. As the prevalent representation for CAD, it is reasonable to try and directly use the b-rep data. However, the data structure's complexity also necessitates complexity within the neural network structure. Jayaraman et al. [9] proposed a graph-based approach to leverage topology paired with 2D convolution in the parameter domain to encode the b-rep geometry. Lambourne et al. [8] introduced a unique convolution technique using a topological neighbourhood of b-rep entities. Both Colligan et al. [10] and Cao et al. [28] adopted a graph-based approach inspired by the attributed adjacency graphs popular in b-rep processing. *Hand-crafted* shape descriptors were used for graph nodes — for instance, face area or b-rep surface type. While these methods are able to easily extract information from the explicitly represented topology, geometric encoding is non-trivial in comparison. The implicit representation of the surface geometries is difficult to process — evidenced by the sophisticated and specialised kernels needed to handle them in CAD software. In addition, the trend in deep representation learning favours allowing systems to learn from raw data rather than "engineering by hand" [15]. This paper argues that the mentioned b-rep methods lean towards the latter approach. Instead, this work proposes to extend more general methods to fit the specific application rather than constructing specific architectures from scratch. Thus allowing the exploitation of advances in the wider field which will be discussed in the following.

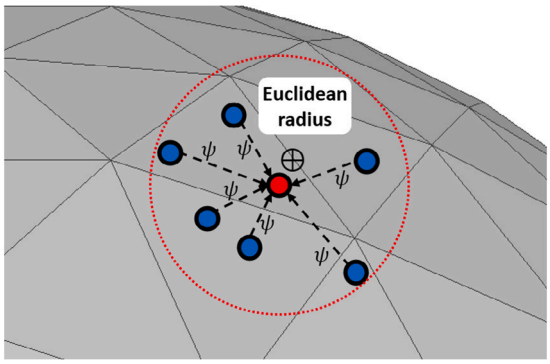

Fig. 1. Illustration of point convolution. Aggregating points within a Euclidean neighbourhood — an implicit local graph.

Another frequently used representation in CAE is the polygonal mesh. Mesh DL approaches in the literature [3,30,31] are often focused on computer graphics applications with slightly less precision requirements to CAE. Those found in the CAE literature claim good results but often struggle with large, high resolution meshes [23]. With the GNN formulation, mesh approaches and point approaches, discussed subsequently, can be seen as variants of each other. However, this work is steered away from mesh encodings for its relative inflexibility — without remeshing, it is tied to the existing vertices. In addition, vertices can often be on b-rep edges which results in labelling ambiguity at the intersection of geometric features.

Point clouds are a popular 3D encoding in DL literature, in large part due to autonomous navigation applications [32,33]. As sets, they could be processed with purely invariant functions [34,35]. However, because of the underlying geometry, the inherent metric space imparts local relationships. Thus, one can utilise a GNN to process this implicit graph. The majority of approaches simply use geometric Euclidean neighbourhoods as connectivity [2,36–39], illustrated in Fig. 1. Alternatively, Wang et al. [40] uses Euclidean neighbourhoods computed in the latent vector space. Viewing them as GNNs, point-based approaches in the literature are mostly variants of each other. The authors of this work advocate for this holistic view which allows more focus on improving specific aspects rather than proposing whole new architectures that introduce confounding variables to any claimed result.

This work chooses a point cloud representation as the geometry centric approach. Its simplicity and flexibility enables a consistent core neural network architecture across applications. The wide literature coverage of these methods speaks to its generality. Kashefi et al. [41] and Kashefi and Mukerji [42] use a point-based approach to perform fluid flow predictions and works by Zhang et al. [11] and Yao et al. [16] suggest that this encoding is able to represent CAD geometry sufficiently for feature identification. Many works in the literature cite drawbacks to point-based approaches as reasons to choose other representations. For instance, it is often incorrectly stated that these approaches fundamentally require a fixed number of points as input and thus sampling issues are faced — salient parts of a geometry can be missed or underrepresented. Colligan et al. [43] tackle this problem but still struggle due to the fixed size input assumption. When viewing point-based approaches in the GNN framework, one can easily see that this is an unnecessary requirement. This and other perceived drawbacks will be addressed in detail in this work to show that these approaches should not be discounted for solid geometry processing. Therefore, this work also serves to collate and develop the discussion on best practices for applying point-based approaches.

3. Method

3.1. The PointNet++ approach

The *PointNet++* architecture, from Qi et al. [2], was chosen as the core structure for the additive study. This applies the *PointNet* set operator [35] – a pioneer in learning on irregular, permutation invariant data – to a neighbourhood for building hierarchical representations. It is arguably the simplest instantiation of a point-based GNN. The opportunity is taken here to summarise the *PointNet++* approach in the lens of the geometric deep learning framework.

For *PointNet++*, Eq. (1) can be simplified to the *'convolutional flavour'* of aggregation [5]:

$$\mathbf{x}'_u = \phi\left(\mathbf{x}_u, \bigoplus_{v\in\mathcal{N}_u} c_{uv}\psi(\mathbf{x}_v)\right) \tag{2}$$

where c_{uv} is a constant weighting for each neighbour. To obtain the *PointNet++* aggregation function: max pooling is used for the symmetric function $\bigoplus$, Euclidean neighbourhood is used for $\mathcal{N}$, and $\mathbf{x}_u$ is removed as a dependence of ϕ. Therefore,

$$\mathbf{x}'_u = \phi\left(\max_{v\in\mathcal{N}_u} \psi(\mathbf{x}_v)\right) \tag{3}$$

In this approach, the latent vector, $\mathbf{x}_v$, is a concatenation of the previous latent vector, $\mathbf{z}$, and the point's coordinate in the neighbourhood frame: $\mathbf{x}_v = [\mathbf{z}_v; \mathbf{p}_v - \mathbf{p}_u]$. It is also worth noting that self-loops are added to the implicit graph; in other words, node u is within $\mathcal{N}_u$.

The parameters of Eq. (3) are shared across nodes in a layer which results in the familiar locality and positional invariance inductive biases of a convolutional-type, shape analyser. Scale separation and hierarchical representations are also useful for shape recognition [44,45]. In 2D CNNs this is achieved by pooling layers; in GNNs this is achieved by downsampling of nodes. A common approach is *farthest point sampling* [46]. The nodes u in the aggregation for Eq. (3) are taken from this downsampled subset, the 'convolution centres', while the nodes v – for the neighbourhoods – are taken from the original set. This aggregates the information from a 'higher-resolution' point cloud to a 'lower-resolution' one, analogous to CNN pooling layers.

3.2. Baseline architecture

To implement a DL system, one needs to arrange layers into an architecture which can learn from data and output relevant predictions. The work by Qi et al. [2] presented a number of *PointNet++* architectures for different benchmark tasks; the current work chooses the architecture proposed for ShapeNet part segmentation [47]. Of the two segmentation tasks addressed in [2], this is likely the most relevant to feature recognition due to the use of a global shape descriptor. Intuitively, the semantics of the shape as a whole should give an indication of what types of features are present.[1]

The baseline architecture is illustrated in Fig. 2; it follows a standard U-Net-like bottleneck architecture [48]. The layer widths and downsampling sizes are those proposed for part segmentation in the original *PointNet++* work. Radii of 0.2 and 0.4 are used for the intermediate convolution layers, with the input point cloud being normalised to a sphere of unit radius centred at zero. It is worth noting that while the *PointNet++* aggregation function allows for further transformation after the symmetric function (ϕ in Eq. (3)), in practice this is not used (i.e. the identity function is used for ϕ). Therefore, the MLPs shown in the convolution section of the architecture correspond to the per-neighbour transformation, ψ. Furthermore, every MLP layer is followed by a batch normalisation layer [49] which is itself followed by a rectified linear unit (ReLU) [50] acting as the non-linear activation function. This is except for the final layer, which has a $softmax$ function [45] instead of an activation function to produce a conditional probability distribution over the possible semantic classes. Lastly, dropout [51] with probability 0.5 is used in the penultimate layer. This baseline and subsequent modifications were implemented as graph neural networks with the *PyTorch* [52] and *PyTorch Geometric* [53] libraries.

[1] This is the case for real geometries that have been designed by engineers with intent, however this is arguably lacking from the shapes in the public machining features datasets.

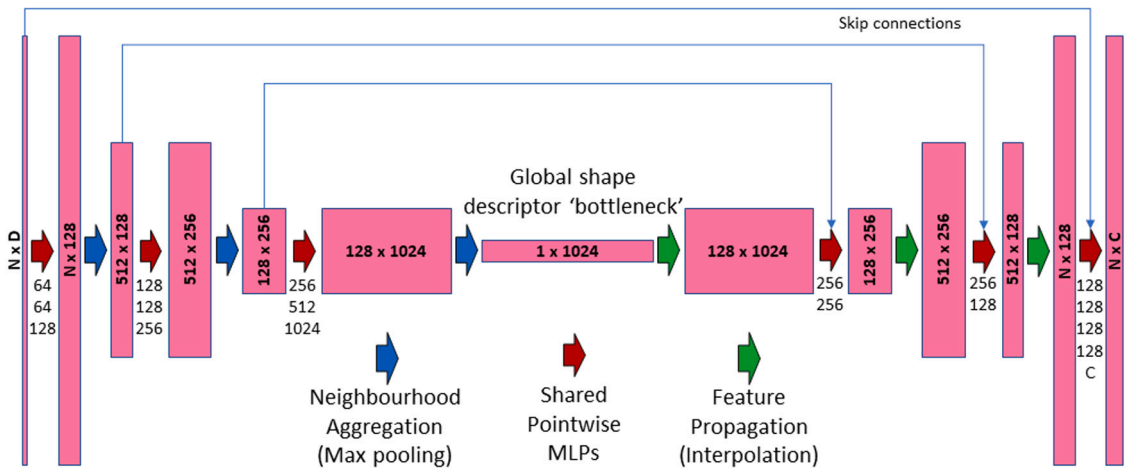

Fig. 2. Block diagram illustrating the baseline *PointNet++* architecture. Arrangement and width of layers is shown. The bottleneck structure is also shown. N is the number of input points, D is the input feature dimension, and C is the number of classes.

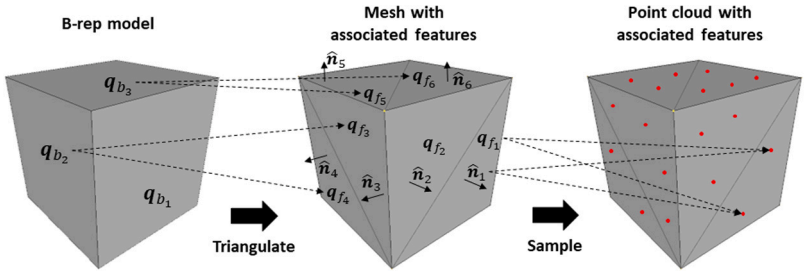

Fig. 3. Illustration of information flow from b-rep model to triangular mesh to point cloud. The chosen b-rep metadata, $\mathbf{q}_{b_x} \in \mathbb{R}^D$, is propagated to associated triangles. The inherited mesh triangle metadata, $\mathbf{q}_{f_x} \in \mathbb{R}^D$, is then also inherited by the sampled points; as well as the unit normal of the triangle, $\hat{\mathbf{n}}_x$.

3.3. Benchmark dataset(s)

As discussed in Section 2, the authors of this work are aware of four publicly available CAD datasets which have semantic segmentation annotations. Both for practicality and the opportunity to test the generality of the proposed modifications, only one of these datasets was used during the additive study and the rest were kept for final evaluation.

The *MFCAD++* dataset was chosen through a process of elimination. It was deemed that the *Fusion 360 Gallery* dataset's labels were semantically ambiguous; in the sense that the shape does not uniquely determine the labels of each face.[2] Therefore, analysis of model learning and predictions during the additive study would likely be more difficult compared to the available machining features datasets. The dataset from Zhang et al. [11] was not chosen as the primary dataset since it is only available in their chosen point format - this is not amenable to the explorations in the current study. Lastly, *MFCAD++* was chosen over *MFCAD* since it was deemed that the former is simply an overall improvement on the latter. At the time of writing, the b-rep method presented within the same work [10] remains the state-of-the-art by default — obtaining an overall face labelling accuracy of 97.37% on the testing set.

The shapes in the *MFCAD++* dataset are primarily made available in the STEP file format. Converting these to a point-based format for this work is relatively trivial with the use of *Python* libraries. To minimise information loss that can occur with b-rep translation [54], the *OpenCASCADE* kernel[3] was used to load and triangulate the shapes — the same one used in [10]. The *Trimesh* library[4] was used to sample points from these triangular meshes. Briefly, to obtain N points, N triangles are sampled (with replacement) using a probability distribution proportional to their areas. A point is then sampled from each triangle using the method described in [55]. To facilitate neural network training, and the additive study more broadly, metadata from the b-rep faces were extracted and propagated to the triangles and the points. Critical metadata to be retained are the semantic label of the b-rep face which the mesh triangle and points belong to, and some identifier in order to aggregate point predictions to b-rep face predictions — for direct comparison with the b-rep method(s). Other metadata available in the b-rep model can also be used and is explored in Section 4.4. The data flow and point cloud extraction is illustrated in Fig. 3.

One could also directly sample points from the continuous b-rep surfaces, avoiding the approximation of curved surfaces by the mesh faces. However, the use of the discretised mesh allows the use of efficient, vectorised implementations. Sampling from the b-rep surface directly is reliant on the specific CAD kernel.

3.4. Training and evaluation details

A standard neural network training approach was employed. The ADAM optimiser was used [56] with an initial learning rate of 0.001 and a weight decay of 0.0001. An exponential learning rate scheduler was also used with a factor of 0.7 and a step size of 20 epochs. These defaults were adapted from the original *PointNet++* work [2]. By default, the cross-entropy [45] loss across point predictions is used for optimisation; modifications to this are explored in Section 4.3. Overall face accuracy was used as the primary performance metric following

[2] This is addressed in their work and is by design — the dataset is meant to implicitly encode *how* humans model 3D shapes in CAD.

[3] Specifically, using the *Python* bindings: https://github.com/tpaviot/pythonocc-core.

[4] https://trimsh.org/index.html

the literature. By default, the b-rep face predictions for calculating this metric were obtained by the modal prediction of the points associated to the face. Later in the additive study, a modification to the architecture allowed the model to make direct face predictions and these were used instead when available.

The official training, validation, and testing split provided with the *MFCAD++* dataset was used. After each training epoch, the validation set was used to track the out-of-sample performance. The neural networks are trained with 300 epochs, this was found to be sufficient for convergence, and the parameters which correspond to the maximum face accuracy across the validation set were extracted, following the lead of Colligan et al. [10].[5] Due to the stochastic nature of neural network training, five training runs were done with different seeds for each configuration to attempt to assess the statistical significance of any performance improvements. When predictions or other results are scrutinised at an individual level, the model with the median accuracy for that configuration was used.

Finally, following best practice [57,58], the validation set is used during the additive study when assessing the potential improvement of model configuration and the testing set is reserved for final evaluation after an 'optimal' configuration is chosen. This ensures that no data leakage occurs; to minimise progressive overfitting and maximise the generalisation of the model, information about the test data must not 'leak' into either model training or configuration.

4. Additive study

4.1. Baseline

To validate the experimental setup, a baseline was obtained using the defaults described in Sections 3.2 and 3.4. It is also worth noting that the unit surface normal at each point is used as per-node input attributes following the literature. An average overall face accuracy of 86.28% was obtained across five training runs with a standard deviation of 0.11%. This is slightly higher than the *PointNet++* result reported by Colligan et al. [10] on their *MFCAD++* dataset. They report the use of the example *PointNet++* implementation within the *PyTorch Geometric* code base which has a different configuration to the reference architecture described in Section 3.2. This and other training hyperparameter differences likely account for the difference. The authors of this work accept this baseline going forward.

4.2. Point coverage

When applying a point-based neural network to CAD models, the literature often randomly samples a fixed number of points from the surface of each shape with 2048 points nominally used. However, this is almost arbitrary when the full continuous surface of the CAD model is available and is merely a choice left over from the 3D benchmarks that the point-based methods were originally designed and tested against.[6] This configuration is particularly problematic when working with CAD models since they often have small faces which can be missed by simple random sampling. For instance, across the *MFCAD++* validation set, it was found that 4.8% of b-rep faces on average are not sampled. Since the performance metric being used is the labelling accuracy of b-rep faces, this imposes an upper-limit to the neural network's measured performance — these faces are not processed and therefore no predictions are made for them. To illustrate this, the baseline configuration achieves an average overall point labelling accuracy of 97.67% with a standard deviation of 0.03% — significantly better than the b-rep face labelling accuracy. Additional analysis to illustrate the consequences of this simple random sampling is given in Appendix A. One way to combat under-sampling of b-rep faces is to simply use more points. It was found that the proportion of b-rep faces which are not sampled decreases linearly with increasing point cloud size — an R^2 value of 0.993.

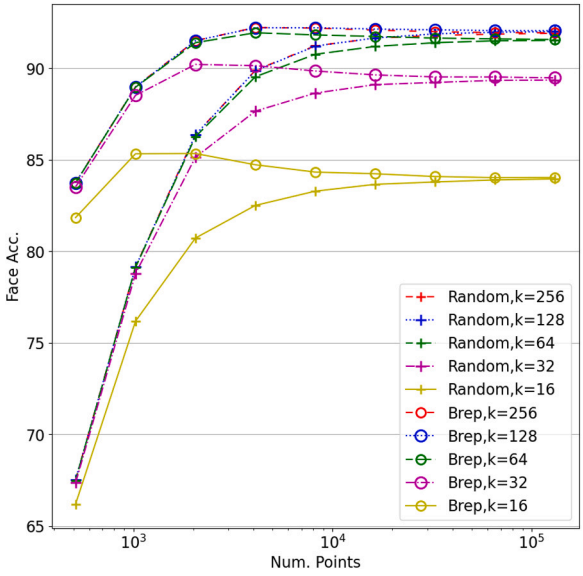

Fig. 4. Validation set performance of a single trained model when faced with different point sampling configurations. Simple random sampling and b-rep stratified sampling is shown across a range of point cloud sizes. Different maximum point neighbourhood sizes are also shown. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

To assess the effect of different sampling strategies on the b-rep face labelling accuracy, a single, trained, baseline model was used. Training models from scratch for each sampling configuration on top of evaluating performance was computationally impractical because of the amount of configurations and, critically, the increasing size of the inputs. Using a model trained on 2048 points randomly sampled from the surface and evaluating on different sampling configurations acts as a surrogate for the true performance. This is possible because the shared weights and the symmetric function aggregation of the convolution make the point-based GNN fundamentally agnostic to point cloud size. To control the experiment, the downsampling sizes of the original architecture are kept constant — i.e. regardless of input point cloud size, 512 points are sampled to act as the convolution centres at the first layer. A limited number of models were trained and evaluated on their 'native' sampling configuration to validate this surrogate approach — an R^2 value of 0.989 was found. Details of which configurations were used to validate are given in Appendix B.

Fig. 4 shows that simply increasing the number of points sampled from the surface makes a significant positive impact on face labelling accuracy up to a point. Once 'full coverage' is reached, extra points do not add information — especially when considering the max pooling aggregation. However, this is not a computationally efficient way to encode the surface.

An alternative is to perform stratified sampling with the b-rep faces being used as the strata. Specifically, the desired number of points is spread out across the b-rep faces based on area, with a minimum of one point per face being enforced. The process is illustrated in algorithm 1. For each stratum, the standard process described in Section 3.3 can then be used to obtain the desired amount of points per b-rep face. This retains the same local point densities of simple random sampling but ensures that all b-rep faces are sampled at least once.

[5] This is not mentioned in their paper but was found to be the case in their public code release: https://gitlab.com/qub_femg/machine-learning/hierarchical-cadnet.

[6] The ShapeNet part segmentation dataset is distributed as point clouds with on average around 2000 points per object.

It is worth noting that because of the one point minimum constraint (and general rounding up), the point clouds are no longer a fixed size — the 'desired' amount of points is merely an estimate. As discussed in Section 2, it is common in literature to assert a uniform size requirement. This is merely an implementation constrain for efficient mini-batching. With modern techniques developed for graph learning [53], one is able to perform mini-batching training with non-uniform sized examples.

Algorithm 1: B-rep stratified sampling

Data: N points desired (minimum)
Mesh, $\mathcal{M} = (\mathcal{V}, \mathcal{F})$, annotated with associated b-rep faces and other features
Result: Point cloud, $\mathcal{P} = \{\mathbf{p} \mid \mathbf{p} \in \mathbb{R}^D\}$
$\mathcal{P} = \{\}$
$n \leftarrow N \div surfaceArea(\mathcal{M})$
foreach *b-rep face in shape* **do**
  $\mathcal{M}' \leftarrow$ mesh faces associated with this b-rep face
  $N' \leftarrow surfaceArea(\mathcal{M}') \times n$
  $N' \leftarrow ceiling(N')$
  $\mathcal{P}' \leftarrow extractPointCloud(\mathcal{M}', N')$
  append $\mathcal{P}'$ to $\mathcal{P}$
end

Fig. 4 also compares the accuracy resulting from using the two sampling strategies, with the green dashed lines being the nominal model configuration. At smaller point cloud sizes, it is observed that b-rep stratified sampling significantly outperforms simple random sampling at a given approximate number of points. At larger point cloud sizes, both sampling strategies converge towards the same asymptotic value. This is because the stratified sampling particularly makes a difference among the smaller b-rep faces when using a smaller point cloud encoding. When using larger point clouds, missing faces with simple random sampling becomes less likely and the advantage of stratified sampling diminishes. Further details of this analysis and empirical results are given in Appendix C.

As previously noted, the stratified sampling approach results in slightly larger point clouds which could be contributing to the accuracy improvement. However, it was found that for the smallest point budgets investigated (approximately 512 and 1024 points) there was only an average increase in point cloud size from simple random sampling of 3% and 1% respectively. This difference further diminishes as point budget increases.[7] The results in Fig. 4 suggest that such a small increase in input point cloud size would not account for the significant accuracy improvement observed.

With the aim of collating best practices for applying point-based GNNs, it is worth addressing a common implementation detail found in the code literature. Because the *PointNet++* aggregation uses a radius neighbourhood, there is no limit to the size of the set $\mathcal{N}_u$. Traditional implementations adapted 2D matrix convolutions with a 1×1 matrix size. An upper limit was set on the number of points each convolution could use, and radii which had less than this were padded to form the dense matrix. Neighbourhoods which exceeded the limit were sampled to a random subset. In theory, this affects the *receptive field* of the neural network since salient features could be missed by the convolution; albeit not as severe as the previously discussed sampling since predictions are still generated via interpolation and *feature propagation* for all input points. To balance computational and memory restrictions, previous implementations treated the maximum size of the set $\mathcal{N}_u$ as a dataset specific hyperparameter. With the availability of optimised 'gather and scatter' operations for GNNs [53], this is less of an issue and one has more freedom in choosing the maximum size of $\mathcal{N}_u$.

[7] Interestingly, the absolute increase stays at around 15 points on average.

Fig. 4 illustrates that the upper limit set for the size of $\mathcal{N}_u$, k, affects the maximum accuracy achieved similarly to the sampling of input points — proportional at lower values and levelling off when a sufficient value is reached. In addition, peaks in accuracy are observed which are more prominent at smaller point cloud sizes. Essentially this is a balance of enough points to represent the surfaces, while not having too much to saturate the neighbourhoods. Further analysis of this is given in Appendix D. Since the effect of the maximum neighbourhood size can be evaluated after a model has been trained, the authors of the present work suggest simply increasing the value until maximum accuracy no longer increases.

To summarise this subsection, it was found that the optimal sampling strategy in terms of the face labelling accuracy metric (for this task and dataset) is to use b-rep stratified sampling with a point budget of approximately 4000 points. In addition, the upper limit set for the number of points in each neighbourhood should be at least 128. The following subsection will treat this as the baseline configuration for further optimisation.

4.3. Aligned loss function

The standard loss function used to optimise the parameters of a *PointNet++* style GNN is the cross-entropy loss computed across point predictions. Effectively, this is optimising for the labelling accuracy of individual points. Each b-rep face prediction is then obtained by the modal prediction of the points associated to it. Since the performance metric being used for comparison is the face labelling accuracy, the gradient descent algorithm is only indirectly optimising the metric of interest.

Ideally, the cross-entropy loss should be calculated across b-rep face predictions in the present application. Since the $argmax$ operation to obtain the point predictions and the modal operation are not strictly differentiable, one cannot simply use these aggregated face predictions to calculate the desired loss and backpropagate. The present work instead utilises an almost multi-task approach.[8] Similar to Zhang et al. [11], an extension to the *PointNet++* architecture is made. An additional prediction branch is added which uses the same feature extractor as the upstream input. This allows conditional probability vectors to be predicted directly for the b-rep faces, and a cross-entropy loss can be calculated on the outputs of each branch.

To be specific, the *PointNet++* feature extractor produces latent vectors for each point. The 'pointwise' prediction branch can then directly apply shared MLPs to this. For the 'facewise' prediction branch, the latent vectors of associated points first need to be aggregated so that each b-rep face has its own latent vector. An obvious choice for this aggregation is the same set abstraction being used upstream. Eq. (3) can be modified to give

$$\mathbf{x}_f = \phi\left(\max_{v \in \mathcal{F}_f} \psi(\mathbf{x}_v)\right) \tag{4}$$

where $\mathbf{x}_f$ is the obtained latent vector for face f and $\mathcal{F}_f$ are the set of points associated to face f. Subsequent MLP layers can then be applied to $\mathbf{x}_f$ to obtain differentiable class probability vectors for the b-rep faces.

Unsurprisingly, it is observed from Fig. 5 that incorporating the b-rep face cross-entropy loss improves the face labelling accuracy. It is also worth noting that the labelling accuracy per point can also be increased using this. Where available, the point predictions to obtain the point accuracy metric in Fig. 5 were obtained by propagating from the predictions of the 'facewise' branch — i.e. the predicted label of a b-rep face was also given to its associated points. This was found to give better accuracy than taking predictions directly from the 'pointwise'

[8] Strictly, both branches are performing the same *task* of semantic segmentation; however, the entities being segmented are different.

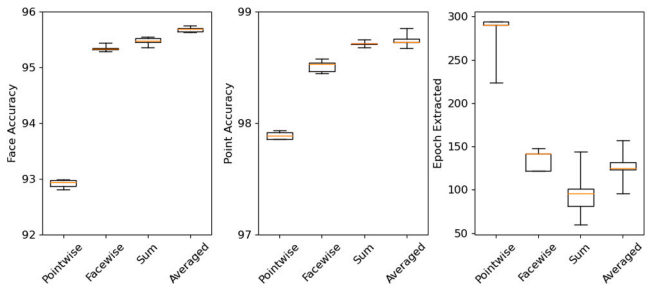

Fig. 5. Box plots showing multiple metrics on the *MFCAD++* validation set achieved by different loss function configurations across five training runs. 'Epoch extracted' refers to the training epoch which achieved the maximum validation face accuracy during training — where model parameters were saved.

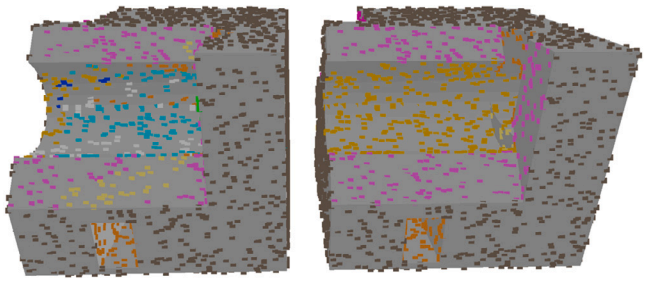

Fig. 6. Predicted point labels illustrated as colours, superimposed on the input CAD model. Both show the same set of points. The left points are coloured based on the direct predictions from the 'pointwise' branch. The right points are coloured based on predictions propagated from the 'facewise' branch. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

branch (for the configurations which jointly optimise those branches). Fig. 6 illustrates the reason for this — point predictions are of course more coherent across b-rep faces since all associated points are given the same predicted label.

In addition, it was found that combining the losses from both branches actually improves labelling accuracy overall — this agrees with other multi-task results in literature [59,60]. Simple addition and averaging of the losses were investigated; it is observed that averaging the point and face cross-entropy losses gives the best labelling accuracy. A small grid search was also performed across the relative weightings when combining the two losses — no noteworthy differences or trends were observed (more details in Appendix E). Finally, it is also worth noting from Fig. 5 that the speed of convergence during training is decreased significantly when using a 'facewise' loss in some way. Intuitively, this is not surprising since the optimiser is now directly trying to maximise the 'convergence' metric — recall from Section 3.4 that the validation face accuracy is being used as the convergence criteria.

With this, the following subsections will use an average of the cross-entropy losses from the face prediction and point prediction branches as the overall loss function to be optimised.

4.4. Input attributes

The final aspect of the data and training pipeline to investigate is the set of input attributes being used. Eq. (3) can aggregate arbitrary latent vectors, thus the input point cloud can be generalised to be $\mathcal{P} = \{\mathbf{p} \mid \mathbf{p} \in \mathbb{R}^D\}$. The contents of the vectors are application dependent, for instance, surface colour can be used when available [61]. However, this paper emphasises their availability since some works in literature disregard it when comparing against point-approaches. For instance, Poulenard and Ovsjanikov [31] only use coordinates.

The unit surface normal at each point has been used in this paper so far. A nested, mini additive study has been done to explore whether the use of any other attributes from the geometries in the *MFCAD++* dataset can be used to improve the labelling accuracy of the neural network. Colligan et al. [10] use shape descriptors extracted from the CAD kernel which summarise the geometry of each b-rep face. As illustrated in Fig. 3, this can be propagated to be used as per-point attributes as well. In addition, this work assigns an arbitrary index to each b-rep face in a geometry, which is then normalised to the interval [0, 1], for use as an attribute — with the aim of delineating points belonging to different b-rep faces.

This nested study progressed by separately adding attributes to a baseline and observing the resulting accuracy improvement, if any. In the spirit of gradient-based optimisation, the best performing configuration is taken as the baseline for the next stage, where the other attributes are added again. For completeness, at the first stage, the baseline is taken as a model which uses only point coordinates — to confirm the usefulness of surface normals. Fig. 7 shows the results of this mini additive study.

It can be clearly observed that the use of surface normals is essential for reasonable labelling accuracy. This information implicitly defines the connectivity of the points and implies the shape of the manifold surface. Oriented surface normals are extremely useful for surface reconstruction from point clouds [62], and thus it is unsurprising that they are also very useful for the neural network's geometry understanding.

The b-rep surface type information subsequently resulted in a statistically significant improvement in face labelling accuracy. This is perhaps surprising since Colligan et al. [10] reports an almost negligible effect when removing this attributes in their approach, because of the imbalance of surface type across the dataset (89.27% of b-rep faces are planar). However, it was found that the current approach was able to leverage this information within the point clouds — with an improvement of +0.89% on the mean face labelling accuracy. This is especially evident when analysing the labelling accuracy of the model variants across only non-planar faces — the improvement then becomes +1.95% when using surface type as an additional attribute.

The last attribute which makes a sufficiently significant improvement is the b-rep face index. As previously stated, this encodes information in point space of the sharp separation between discrete b-rep faces. The model then has this information available throughout the feature extractor section, rather than only at the 'facewise' prediction branch — which seems to improve semantic segmentation accuracy. As before, the baseline configuration to be used in the following will be the best performing so far. The neural networks will now use unit surface normals, b-rep surface type, and face index as per-point input attributes.

4.5. Scaling

Now that the current approach has been brought to equal footing with a specialised b-rep approach in terms of input data and training signals. It is also interesting to consider scale. The current model has 1.4M learnable parameters compared to *Hierarchical CADNet* [10] which has 9.8M.

The two main ways to scale a neural network is through depth and width. In the *PointNet++* approach, the former refers to the number of set abstraction modules in the network — a block of neighbourhood aggregation and subsequent transformation via MLPs. The latter refers to the number of parameters ('neurons') being used in the pointwise MLPs. Both of these options naturally change the number of parameters of the network; however, for most point-based approaches, changing the number of convolution centres could also be seen as scaling the model while keeping the number of parameters the same. This hyperparameter could equally be addressed in Section 4.2, especially because

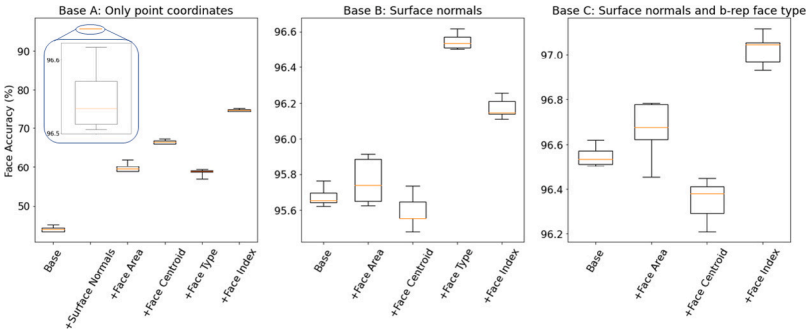

Fig. 7. Box plots showing the face labelling accuracy on the *MFCAD++* validation set of different configurations. Each plot starts with a base configuration and adds input attributes separately.

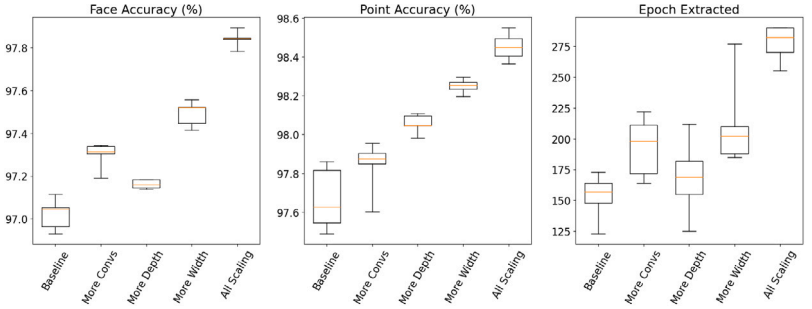

Fig. 8. Box plots showing multiple metrics on the *MFCAD++* validation achieved by scaled configurations across five training runs.

it can be changed at evaluation of a trained model. However, modifying the number of convolution centres of the network significantly increases memory usage — a similarly extensive input coverage study that includes this would have been computationally impractical. It was decided that this hyperparameter would be tackled at this stage, after a sampling configuration had already been chosen. This may not result in an optimal model configuration – a larger number of convolution centres may have revealed a different peak in Fig. 4 – but the main purpose of this work is to explore best practices for applying point-based methods to 'real' applications, rather than maximising performance on artificial benchmarks.

A simple implementation of model scaling was used in this work. When 'widening' the network, only the number of parameters of the first of the three MLPs in the first layer is specified; with the three MLPs getting sizes of $M, M, 2M$, where M is the number of parameters of the first MLP. M is then doubled for subsequent set abstraction modules. This geometric sequence of parameter numbers is extended when 'deepening' the model by adding layers.

It is worth stressing that this subsection does not aim to find the optimal architecture configuration for the *MFCAD++* dataset. One could perform a full grid search or some other meta-optimisation method of the network structure to get an optimal and efficient configuration; however, this is computationally impractical and likely dataset specific. Instead, this subsection simply serves to illustrate the potential improvements of scaling this relatively simple network — with the conjecture that a moderate portion of performance gain of newly proposed architectures can be attributed to scale rather than fundamental changes, an idea shared by [21].

Fig. 8 shows the performance gain when individually scaling up the hyperparameters described above. The performance of a model with all three hyperparameters scaled is also shown. It was decided that a nested additive study was not necessary here since all three hyperparameters more or less just increase the model's representation power. The 'All Scaling' configuration now has 10.9M learnable parameters and so is not surprising that it also consistently takes longer to train. Noting the y-axis range of the face accuracy and point accuracy plots, the significant network scaling has not made a very significant performance improvement. On the other hand, the ultimate scaled model (as well as the widened model with 5.1M learnable parameters) has now surpassed the reported accuracy of *Hierarchical CADNet* [10] as measured by the validation set. This is an important result which supports the hypothesis of the current work — extending generic methods to fit applications can not only be competitive but surpass specifically designed approaches. This result is confirmed using the official testing set in Section 5.1.

5. Results

This section applies the final model configuration resulting from the additive study, to give evidence that the proposed modifications result in a general improvement of the approach rather than being specific to the dataset they were optimised against. An instance of the final model configuration was trained on each dataset's training split and evaluated on their testing splits. Table 1 shows that the modifications presented in this work significantly improve on the result obtained by a standard *PointNet++* implementation. In addition, the model configuration presented here obtained state-of-the-art performance on one dataset and competitive results on another. The potential reasons for the relatively poor performance on the *Fusion 360 Gallery* dataset are discussed in Section 5.3.

5.1. *MFCAD++ dataset*

Fig. 9 summarises the predictions across the *MFCAD++* testing split of a version of the final model configuration which achieved the median overall accuracy among the five training runs. The values of the confusion matrix are normalised across the rows to account for the imbalance of classes. No significant failure cases are observed across the semantic classes. The lowest accuracy class, the 'rectangular through slot', is correctly identified 91% of the time — being confused with the 'rectangular passage' class 8% of the time. This and the two other semantic classes with less than 95% accuracy tend to be confused with semantic classes which are geometrically similar to them.

Table 1
Summary table of predictive accuracy of final model configuration against different segmentation datasets. Accuracy of a standard *PointNet++* implementation is also shown. Results for representative state-of-the-art models are taken from literature.

Dataset	Method	Face Accuracy (%)
MFCAD++	Standard *PointNet++*	86.28 ± 0.11
	Hierarchical CADNet [10]	97.37
	Modified *PointNet++*	**97.79 ± 0.05**
Zhang Machining	'Standard' *PointNet++*	97.89 ± 0.10
	Zhang Hybrid Model [11]	99.84
	Modified *PointNet++*	**99.12 ± 0.08**
Fusion 360 Gallery	Standard *PointNet++*	70.78 ± 0.12
	BRepNet [8]	92.52 ± 0.15
	Modified *PointNet++*	**84.52 ± 0.13**

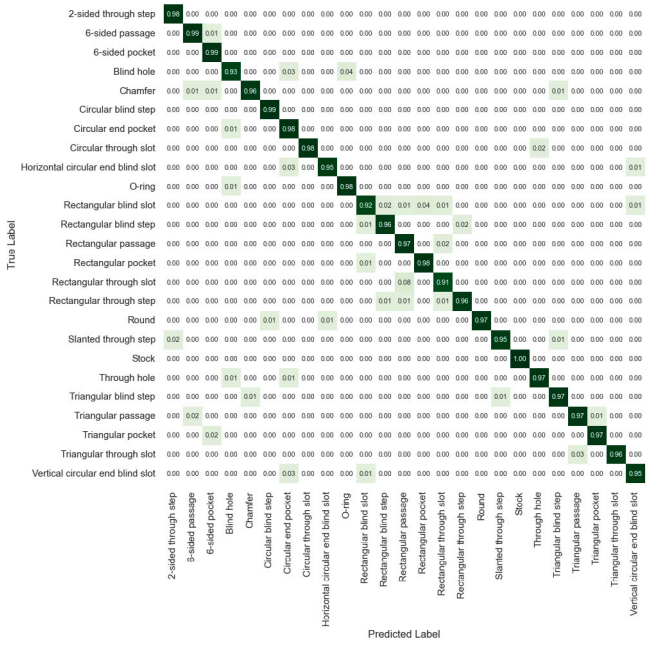

Fig. 9. Confusion matrix summarising the prediction accuracy of the final model configuration on the *MFCAD++* dataset testing split.

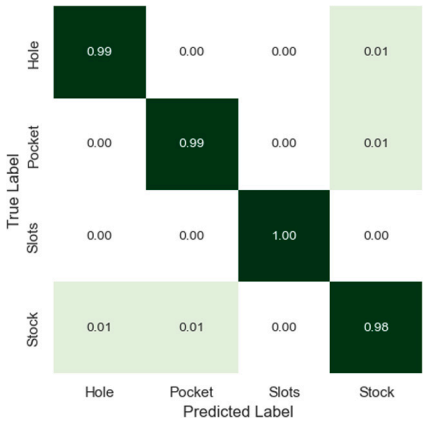

Fig. 10. Confusion matrix summarising the prediction accuracy of the final model configuration on the Zhang et al. machining features dataset testing split.

Fig. 11. An example of a geometry from the Zhang machining features dataset represented as a point cloud.

5.2. Other machining dataset

Fig. 10 summarises the predictions of a median model on the testing split of the machining features dataset from Zhang et al. [11]. It is worth noting that the dataset which is made publicly available is only a subset of the one used in their work[9]; only containing four semantic labels — stock, pocket, holes, and slots. A version of their model that was pre-trained on this subset is made available and is used for the comparison in Table 1 instead of the result stated in the paper.

In addition, the dataset is only distributed as pre-sampled point clouds. The full surface is lacking, therefore the sampling scheme presented in Section 4.2 could not be applied. 32 points have been sampled from each b-rep face regardless of size, resulting in a non-uniform coverage across the surface geometry as shown in Fig. 11. Note that this also ensures that no b-rep faces are left unsampled and likely explains the relatively high result of the 'standard' *PointNet++* implementation shown in Table 1. Because the b-rep data is not available for this dataset, only the surface normals and arbitrary b-rep indices could be used as per-point input features. Potentially the modified *PointNet++* method presented in the current work could achieve higher prediction accuracy given the full b-rep data. However, this result gives evidence to the generality of the method; it is able to process point cloud inputs without the need to reconstruct a surface mesh or reverse engineer a b-rep model.

5.3. Fusion 360 Gallery segmentation dataset

Fig. 12 summarises the predictions of a median model on the *Fusion 360 Gallery* segmentation dataset [8] official testing split. The most confused classes reflect the ambiguity within the dataset discussed in Section 3.3. Specifically, faces which were created with revolve or cut modelling operations are confused as being the result of extrude operations.

The model's performance as illustrated by the overall face accuracy metric or the confusion matrix could be seen as misleading for this dataset. Fig. 13 shows examples of model predictions which are technically incorrect when comparing to the dataset's ground truth labels. However, these labels are actually valid — the shape could be equivalently created with the modelling operations shown. Fundamentally, the ambiguity is present because there is no information about the 2D sketches used to create the geometries within the dataset.

This is not to say that all the model's confusion is to do with the ambiguity — Fig. 14 shows an example of incorrect and non-coherent

[9] https://github.com/HARRIXJANG/ASIN-master

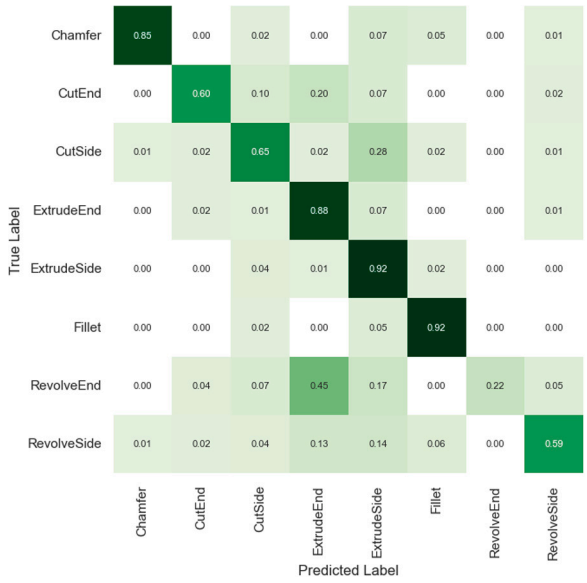

Fig. 12. Confusion matrix summarising the prediction accuracy of the final model configuration on the *Fusion 360 Gallery* segmentation dataset testing split.

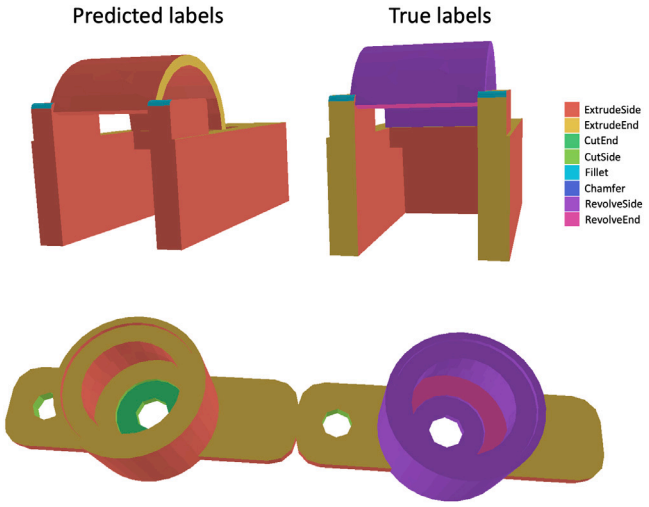

Fig. 13. Example of geometries where the model's predictions are technically incorrect but are equally valid for creating the shape. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Fig. 14. Example of geometries where the model's predictions are actually incorrect. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

predictions from the model. In this case, the 3D shape has been created by revolving a sketch in the shape of the number two with a large radius — as evidenced by the ground truth labels and the change in thickness across the numeral. A straight extrusion, as predicted by the model, would not produce this exact shape with the curve in the depth-wise direction. Interestingly, the model predicts the side faces towards the thicker side as fillets instead of the sides of extrusions; perhaps noticing the curvature.

Table 2
B-rep face labelling accuracy of different variations of the *Point Transformer* approach. The improvement from the baseline model is shown in green.

Variation	Testing split face accuracy (%)	
	MFCAD++	*Fusion360 Gallery*
Baseline	84.74	68.15
with stratified sampling	89.76 (+5.02)	72.60 (+4.45)
with b-rep face loss	94.98 (+10.24)	75.57 (+7.42)
with extra point attributes	95.85 (+11.11)	80.46 (+12.31)

The b-rep neural network presented by Lambourne et al. [8] is meant to learn *how* users design shapes from this dataset. At this time, it is unclear to the authors of the current work how their model is able to learn this, but it is speculated that explicit b-rep face topology may be very important. This is currently lacking from the face prediction branch of the approach presented here and is left as future work.

5.4. Alternative base neural network: Point Transformer

The majority of the proposed extensions are largely agnostic to the underlying point-based neural network used. This section provides some evidence for this by applying the modifications to another state-of-the-art point-based neural network from literature — *Point Transformer* from Zhao et al. [39]. A key characteristic of this approach is the use of a *self-attention* mechanism in the ϕ function of Eq. (1). For details of this approach, the interested reader is referred to the original work.

The implementation found within *PyTorch Geometric* is used here which constructs a model with 4.6M learnable parameters, larger than the default *PointNet++* architecture. This was then extended by using b-rep stratified sampling, b-rep face loss, and extra point attributes without changing the core architecture or code. Table 2 suggests the generality of the proposed extensions to other point-based approaches. It is observed that the labelling accuracy of the *Point Transformer* model is significantly improved when properly taking advantage of the available b-rep data.

6. Conclusions

The current work shows that point-based approaches should not be discounted when it comes to 3D CAD applications. The point cloud representation is flexible enough to accommodate the rich information available within b-rep models. In addition, the graph neural network formulation seems to be able to learn internal representations that are sufficiently descriptive as to make useful predictions using simple Euclidean locality. The additive study explores modifications to a baseline *PointNet++* neural network to give better performance in CAD applications. The most important can be summarised as three key concepts: full surface coverage, a loss function aligned with the task, and taking advantage of b-rep features.

With these modifications, the approach was able to achieve a new state-of-the-art result on the *MFCAD++* machining features dataset. Outperforming a b-rep neural network which uses 'hand-crafted' features. It provides competitive results on the machining features dataset presented by Zhang et al. [11] as compared to a hybrid b-rep/point approach. And it significantly improves on the performance of the 'standard' *PointNet++* model on the *Fusion360 Gallery* segmentation dataset. Results were also presented which suggest that the improvements afforded by the proposed extensions are not specific to the *PointNet++* network and can be used for other point-based approaches.

The key contributions of this work are as follows:

- Extensions are presented for properly applying point-based neural networks to CAD tasks involving surface geometry. Particularly how to take advantage of available b-rep information within the relatively simple data representation.
- These extensions significantly improve the performance of a point-based approach. State-of-the-art accuracy is illustrated on the *MFCAD++* machining features dataset when applying proposed extensions.
- Experimental results are provided through an additive study to document the construction of the system and to explore the behaviour and performance effect of individual modifications. This also serves as best-practices for those seeking to use point-based networks for their application.
- Significant critique of the literature is presented and evidence is given to counter some incorrect assumptions about point-based methods for solid geometry applications.

7. Future work

The additive study explores best practices and presents work towards optimal model configurations; however, it is acknowledged that there are complex interactions between the various hyperparameters being studied. An attempt was made to order the improvements in such a way as to minimise dependencies. Nevertheless, the optimal model configuration for a given dataset is likely to require a combinatorial search over all hyperparameters and this is suggested as future work.

Some evidence has been given for the application of the proposed extensions to other point-based approaches. With the flexibility of the geometric deep learning framework, other approaches could be studied. It is left as future work to investigate whether changing other aspects of the learning systems would be useful for CAD applications. For instance, changing the neighbourhood selection [40].

A motivation for utilising a geometry-first approach in this work is to allow for applications outside b-rep models. This has only been hinted at in the current paper and would be interesting to address in a future work - specifically, can the same architecture, if not the same trained model, perform well across 3D domains.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Gerico Vidanes reports financial support was provided by Rolls-Royce plc.

Data availability

Datasets used to train the models are already publicly available. Code note currently publicly available due to industry sponsor request.

Acknowledgements

The authors gratefully acknowledge support from the UK Defence Science and Technology Labs via contract DSTLX-1000152302, and special thanks to Dr. Fred Witham for his support in this research.

Appendix A. Baseline model predictions with random sampling

A confusion matrix of face predictions is shown in Fig. A.15. On top of the 25 machining feature labels, a 'NULL' class was also added to denote the b-rep faces for which the neural network produced no label — because they were not sampled. Of course the true label row for this class is empty since this is not an actual ground truth label. However, it is observed that its column for 'predicted' label makes up a noticeable

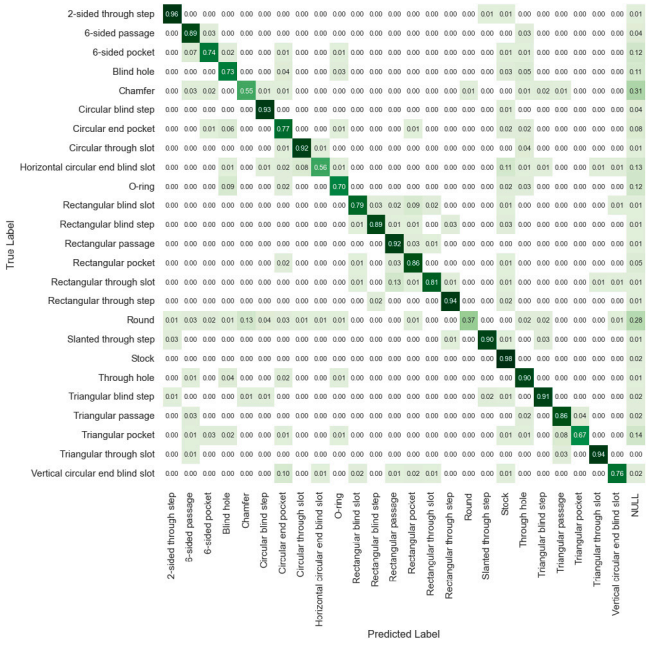

Fig. A.15. Confusion matrix for b-rep face predictions of a trained baseline model configuration.

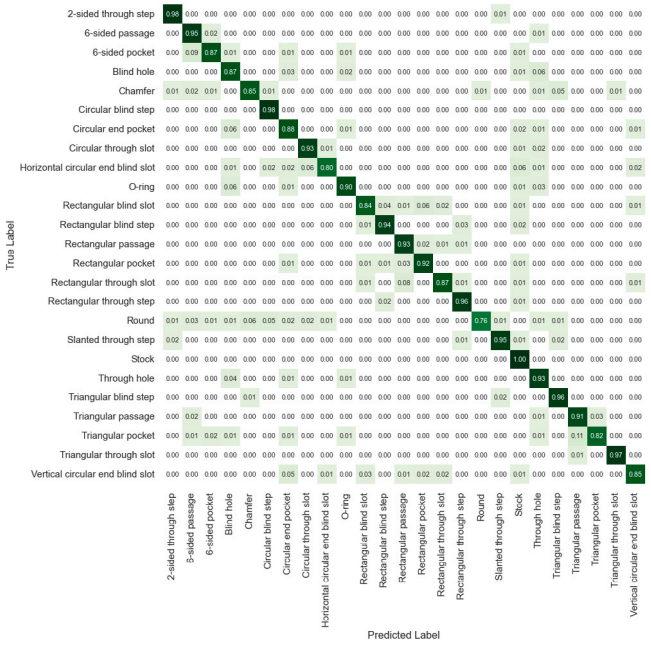

Fig. A.16. Confusion matrix for point predictions of a trained baseline model configuration.

mass of the matrix — the *pocket* classes seem to consistently be missed by the sampling and thus obtain a 'NULL' label. These classes seem to also commonly be misclassified as passages, likely linked to the fact that they are under-sampled — the neural network is shown these classes less in training. Section 4.2 in the main text seeks to alleviate this.

On top of under-sampling, the simple random sampling is actually causing the dataset to become more imbalanced overall, from the neural network's point of view; illustrated in Fig. A.16 by the significant proportion of the confusion matrix mass concentrated within the *stock* class. While the *MFCAD++* dataset has a reasonable class balance when

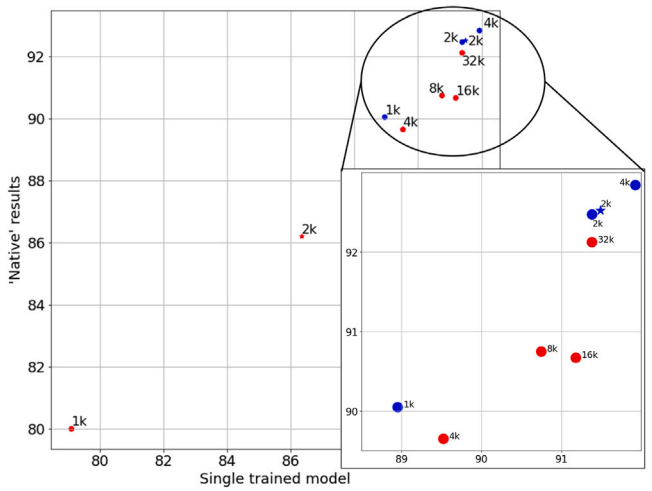

Fig. B.17. Evaluated b-rep face labelling accuracy of a single trained model with different sampling strategies against the accuracy of models trained and evaluated on the 'native' resolution. Data in red are simple random sampling and in blue are using b-rep stratified sampling. Solid circles use a maximum of 64 point neighbours during point convolution while starred data use 128. Data points are annotated with point cloud size. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

measured in terms of b-rep face labels, as reported by Colligan et al. [10], because the number of points sampled from a face is proportional to its size, machining features which tend to be larger will be over-represented in the dataset (and vice-versa). This is further illustrated by the fact that the *pocket* and *slot* classes have a disappearing amount of true labels in Fig. A.16. Section 4.3 in the main text seeks to alleviate the effect of this imbalance on the training.

Appendix B. Sampling study surrogate validation

Fig. B.17 shows the correlation of the single trained model surrogate approach with a limited number of results corresponding to neural networks trained and evaluated on a given sampling configuration.

Appendix C. Advantages of b-rep stratified sampling for small point budgets

The stratified sampling particularly makes a difference among the smaller b-rep faces when using a smaller input point cloud. Whereas when using larger point clouds, the distribution of points across the faces created by the sampling methods become very similar. This is shown for one geometry in Fig. C.18.

Quantitatively, one can define a dissimilarity metric between the two discrete distributions and compute summary statistics across the entire validation set for each point cloud size. The χ^2 histogram distance [63] is used here:

$$D(X,Y) = \sum_i \frac{(X_i - Y_i)^2}{(X_i + Y_i)} \tag{C.1}$$

The elementwise natural log of the corresponding distributions are used for X_i and Y_i, this further biases the metric towards the differences in the lower values and therefore smaller b-rep faces - a necessity suggested by Fig. C.18. Fig. C.19 confirms the aforementioned trend across the *MFCAD++* validation set. Showing that the distribution of points created by b-rep stratified sampling and simple random sampling is similar when using large point budgets and very dissimilar when using small point cloud encodings.

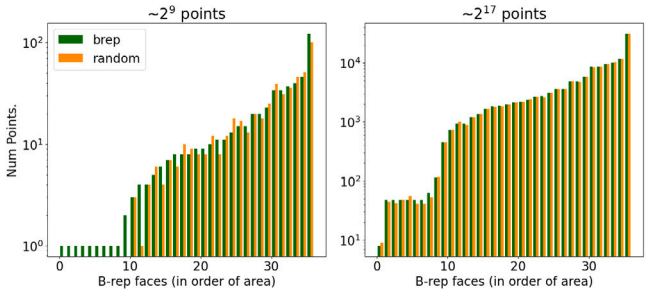

Fig. C.18. Histograms showing the number of points sampled from each b-rep face in a single *MFCAD++* example. B-rep faces are sorted in order of surface areas. The distribution resulting in using b-rep stratified sampling and simple random sampling are shown side-by-side for comparison. Two input point cloud sizes are also shown for comparison. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

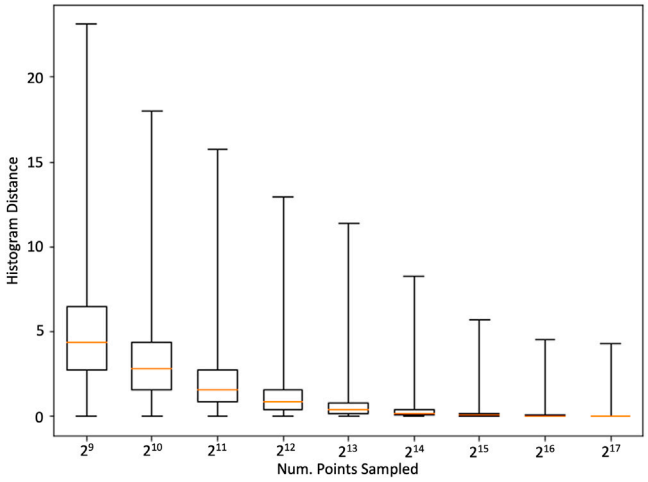

Fig. C.19. Box plots summarising the χ^2 histogram distance of point distributions obtained by b-rep stratified sampling and simple random sampling, across the *MFCAD++* validation set. For each point budget, each geometry is sampled once with each sampling method. The two point distributions obtained are then used to calculate a χ^2 metric for each geometry. These values are then used to create the box plot for each point budget.

Appendix D. Effect of insufficient downsampling neighbourhoods

One can plot the distribution of points being used in the convolutions across the b-rep faces to see the effect of insufficient maximum size of the set $\mathcal{N}$ relative to the total number of input points. Fig. D.20 shows that, for this specific shape and sampling, an upper limit of 64 aggregated points per convolution is sufficient even up to around eight thousand input points. However, at extreme input point cloud sizes, the neighbourhood saturation results in multiple faces not being represented in any of the convolutions. In Fig. D.21, it is observed that even at medium point cloud sizes, an upper limit of 16 is potentially not sufficient — one of the small faces is not represented in the convolutions.

Appendix E. Multi-branch loss weighted combinations

The effect of different weightings for the combination of the cross-entropy losses from the point prediction and face prediction branches was investigated using a small grid search. Fig. E.22 shows that there is no advantage to unbalanced combinations. Interestingly, accuracy decreases when summing the losses with unequal weightings. It is also observed that weighted averages do not make a significant difference compared to standard averaging of the two losses from the prediction branches.

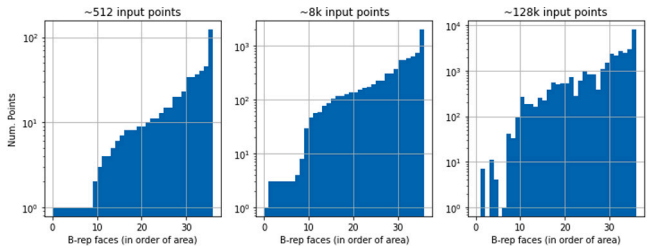

Fig. D.20. Histograms showing the number of points from each b-rep face which were used in at least one convolution, in a single *MFCAD++* example. The maximum size of set $\mathcal{N}$ has been set to 64. B-rep faces are sorted in order of surface areas. The input points were obtained with b-rep stratified sampling. Three input point cloud sizes are shown.

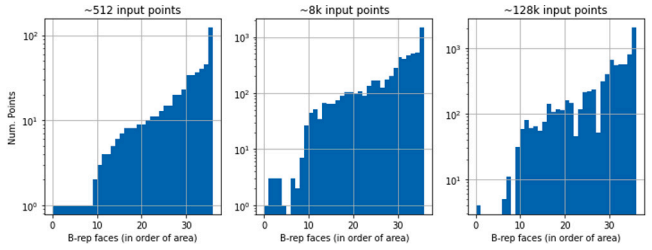

Fig. D.21. Histograms showing the number of points from each b-rep face which were used in at least one convolution, in a single *MFCAD++* example. The maximum size of set $\mathcal{N}$ has been set to 16. B-rep faces are sorted in order of surface areas. The input points were obtained with b-rep stratified sampling. Three input point cloud sizes are shown.

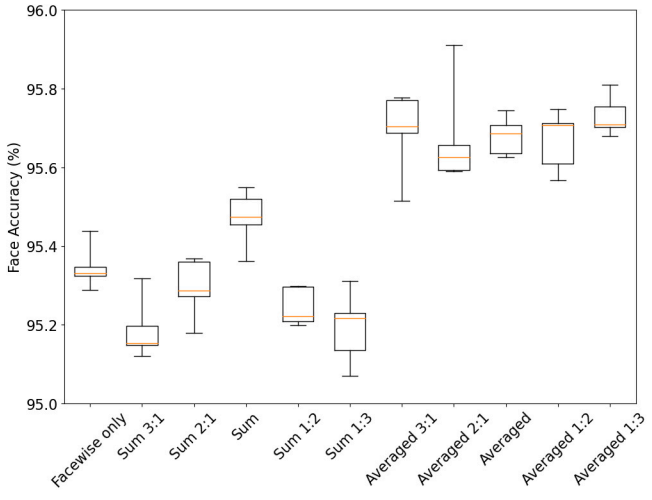

Fig. E.22. Box plot showing the different face labelling accuracies achieved by different total loss function configurations. The configuration with only pointwise loss has been omitted for clarity — it is significantly lower than the rest as shown in 5.

References

[1] Zhang Z, Jaiswal P, Rai R. Featurenet: Machining feature recognition based on 3d convolution neural network. Comput Aided Des 2018.

[2] Qi CR, Yi L, Su H, Guibas LJ. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Adv Neural Inf Process Syst 2017.

[3] Hanocka R, Hertz A, Fish N, Giryes R, Fleishman S, Cohen-Or D. Meshcnn: a network with an edge. ACM Trans Graph 2019.

[4] Su H, Maji S, Kalogerakis E, Learned-Miller E. Multi-view convolutional neural networks for 3d shape recognition. In: Proceedings of the IEEE international conference on computer vision. 2015, p. 945–53.

[5] Bronstein MM, Bruna J, Cohen T, Veličković P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. 2021, arXiv preprint arXiv:2104.13478.

[6] Chen Y, Zhang F, Zhang C, Xue T, Zekelman LR, He J, et al. White matter tracts are point clouds: Neuropsychological score prediction and critical region localization via geometric deep learning. 2022, arXiv preprint arXiv:2207.02402.

[7] Jackson JM, Liu R, Dyer EL. Building representations of different brain areas through hierarchical point cloud networks. In: Medical imaging with deep learning. 2022.

[8] Lambourne JG, Willis KD, Jayaraman PK, Sanghi A, Meltzer P, Shayani H. BRepNet: A topological message passing system for solid models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021, p. 12773–82.

[9] Jayaraman PK, Sanghi A, Lambourne JG, Willis KD, Davies T, Shayani H, et al. Uv-net: Learning from boundary representations. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021, p. 11703–12.

[10] Colligan A, Robinson T, Nolan D, Hua Y, Cao W. Hierarchical CADNet: Learning from B-reps for machining feature recognition. Comput Aided Des 2022.

[11] Zhang H, Zhang S, Zhang Y, Liang J, Wang Z. Machining feature recognition based on a novel multi-task deep learning network. Robot Comput-Integr Manuf 2022.

[12] Al-Wswasi M, Ivanov A, Makatsoris H. A survey on smart automated computer-aided process planning (ACAPP) techniques. Int J Adv Manuf Technol 2018.

[13] Zhang X, Toal DJ, Bressloff N, Keane A, Witham F, Gregory J, et al. Prometheus: a geometry-centric optimisation system for combustor design. In: Turbo expo: Power for land, sea, and air. 2014.

[14] Benali Amjoud A, Amrouch M. Convolutional neural networks backbones for object detection. In: International conference on image and signal processing. Springer; 2020, p. 282–9.

[15] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature 2015.

[16] Yao X, Wang D, Yu T, Luan C, Fu J. A machining feature recognition approach based on hierarchical neural network for multi-feature point cloud models. J Intell Manuf 2022.

[17] Várady T, Martin RR, Cox J. Reverse engineering of geometric models—an introduction. Comput Aided Des 1997.

[18] Dawes WN, Meah N, Kudryavtsev A, Evans R, Hunt M, Tiller P. Digital geometry to support a gas turbine digital twin. In: AIAA scitech 2019 forum. 2019, p. 1715.

[19] Salval H, Keane A, Toal D. Multiresolution surface blending for detail reconstruction. Graph Vis Comput 2022.

[20] Liu Z, Mao H, Wu C-Y, Feichtenhofer C, Darrell T, Xie S. A convnet for the 2020s. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022, p. 11976–86.

[21] Qian G, Li Y, Peng H, Mai J, Hammoud HAAK, Elhoseiny M, et al. PointNeXt: Revisiting PointNet++ with improved training and scaling strategies. In: Advances in neural information processing systems. 2022, p. 23192–204.

[22] Shah JJ, Anderson D, Kim YS, Joshi S. A discourse on geometric feature recognition from CAD models. J Comput Inf Sci Eng 2000.

[23] Nobari AH, Rey J, Kodali S, Jones M, Ahmed F. Conformal predictions enhanced expert-guided meshing with graph neural networks. 2023, arXiv preprint arXiv:2308.07358.

[24] Joshi S, Chang T. Graph-based heuristics for recognition of machined features from a 3D solid model. Comput Aided Des 1988.

[25] Prabhakar S, Henderson MR. Automatic form-feature recognition using neural-network-based techniques on boundary representations of solid models. Comput Aided Des 1992.

[26] Sunil V, Pande S. Automatic recognition of machining features using artificial neural networks. Int J Adv Manuf Technol 2009.

[27] 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.

[28] 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: International design engineering technical conferences and computers and information in engineering conference. 2020.

[29] Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE. Neural message passing for quantum chemistry. In: International conference on machine learning. 2017, p. 1263–72.

[30] Masci J, Boscaini D, Bronstein M, Vandergheynst P. Geodesic convolutional neural networks on riemannian manifolds. In: Proceedings of the IEEE international conference on computer vision workshops. 2015, p. 37–45.

[31] Poulenard A, Ovsjanikov M. Multi-directional geodesic neural networks via equivariant convolution. ACM Trans Graph 2018.

[32] Huang L. Review on LiDAR-based SLAM techniques. In: 2021 international conference on signal processing and machine learning. 2021, p. 163–8.

[33] Li Y, Ibanez-Guzman J. Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems. IEEE Signal Process Mag 2020.

[34] Zaheer M, Kottur S, Ravanbakhsh S, Poczos B, Salakhutdinov RR, Smola AJ. Deep sets. Adv Neural Inf Process Syst 2017.

[35] Qi CR, Su H, Mo K, Guibas LJ. Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017, p. 652–60.

[36] Xu Y, Fan T, Xu M, Zeng L, Qiao Y. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In: Proceedings of the European conference on computer vision. 2018, p. 87–102.
[37] Liu Y, Fan B, Xiang S, Pan C. Relation-shape convolutional neural network for point cloud analysis. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019, p. 8895–904.
[38] Lan S, Yu R, Yu G, Davis LS. Modeling local geometric structure of 3d point clouds using geo-cnn. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019, p. 998–1008.
[39] Zhao H, Jiang L, Jia J, Torr PH, Koltun V. Point transformer. In: Proceedings of the IEEE/CVF international conference on computer vision. 2021, p. 16259–68.
[40] Wang Y, Sun Y, Liu Z, Sarma SE, Bronstein MM, Solomon JM. Dynamic graph cnn for learning on point clouds. ACM Trans Graph 2019.
[41] Kashefi A, Rempe D, Guibas LJ. A point-cloud deep learning framework for prediction of fluid flow fields on irregular geometries. Phys Fluids 2021.
[42] Kashefi A, Mukerji T. Physics-informed PointNet: A deep learning solver for steady-state incompressible flows and thermal fields on multiple sets of irregular geometries. J Comput Phys 2022.
[43] Colligan AR, Robinson TT, Nolan DC, Hua Y. Point cloud dataset creation for machine learning on cad models. Comput Aided Des Appl 2021.
[44] LeCun Y, Boser B, Denker J, Henderson D, Howard R, Hubbard W, et al. Handwritten digit recognition with a back-propagation network. Adv Neural Inf Process Syst 1989.
[45] Goodfellow I, Bengio Y, Courville A. Deep learning. MIT Press; 2016.
[46] Eldar Y, Lindenbaum M, Porat M, Zeevi YY. The farthest point strategy for progressive image sampling. IEEE Trans Image Process 1997.
[47] Yi L, Kim VG, Ceylan D, Shen I-C, Yan M, Su H, et al. A scalable active framework for region annotation in 3D shape collections. SIGGRAPH Asia 2016.
[48] Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation. In: International conference on medical image computing and computer-assisted intervention. 2015, p. 234–41.
[49] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning. 2015, p. 448–56.
[50] Nair V, Hinton GE. Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning. 2010, p. 807–14.
[51] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 2014.
[52] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. Pytorch: An imperative style, high-performance deep learning library. Adv Neural Inf Process Syst 2019.
[53] Fey M, Lenssen JE. Fast graph representation learning with PyTorch geometric. 2019, arXiv preprint arXiv:1903.02428.
[54] Dimitrov L, Valchkova F. Problems with 3D data exchange between CAD systems using neutral formats. Proc Manuf Syst 2011.
[55] Weisstein EW. Triangle point picking. MathWorld–A Wolfram Web Resource; 2006, URL: https://mathworld.wolfram.com/TrianglePointPicking.html. [Accessed 21 September 2022].
[56] Kingma DP, Ba J. Adam: A method for stochastic optimization. 2014, arXiv preprint arXiv:1412.6980.
[57] Kapoor S, Narayanan A. Leakage and the reproducibility crisis in ML-based science. 2022, arXiv preprint arXiv:2207.07048.
[58] Lones MA. How to avoid machine learning pitfalls: a guide for academic researchers. 2021, arXiv preprint arXiv:2108.02497v2.
[59] He K, Gkioxari G, Dollár P, Girshick R. Mask R-CNN. In: Proceedings of the IEEE international conference on computer vision. 2017, p. 2961–9.
[60] Standley T, Zamir A, Chen D, Guibas L, Malik J, Savarese S. Which tasks should be learned together in multi-task learning? In: International conference on machine learning. 2020, p. 9120–32.
[61] Thomas H, Qi CR, Deschaud J-E, Marcotegui B, Goulette F, Guibas LJ. Kpconv: Flexible and deformable convolution for point clouds. In: Proceedings of the IEEE/CVF international conference on computer vision. 2019, p. 6411–20.
[62] Berger M, Tagliasacchi A, Seversky LM, Alliez P, Guennebaud G, Levine JA, et al. A survey of surface reconstruction from point clouds. In: Computer graphics forum. 2017, p. 301–29.
[63] Pele O, Werman M. The Quadratic-Chi histogram distance family. In: Computer vision – ECCV 2010. 2010, p. 749–62.