Self-Supervised Representation Learning for CAD

Benjamin T. Jones[1] Michael Hu[1] Milin Kodnongbua [1] Vladimir G. Kim[2] Adriana Schulz[1]
[1]University of Washington [2]Adobe Research
{benjones, mkhu, milink, adriana}@cs.washington.edu vokim@adobe.com

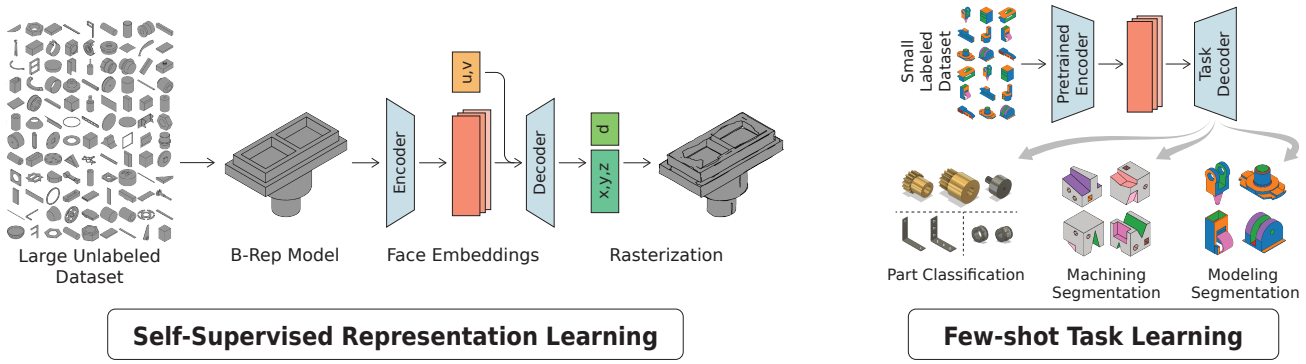


Figure 1. Overview of our technique. We train a geometric self-supervision task of a large, unlabeled dataset of CAD Boundary Representations (B-Reps) to learn geometrically relevant representations for each B-Rep face. These pre-trained representations are then used to train few-shot segmentation and classification learning tasks on labeled B-Rep datasets.

Abstract

Virtually every object in the modern world was created, modified, analyzed and optimized using computer aided design (CAD) tools. An active CAD research area is the use of data-driven machine learning methods to learn from the massive repositories of geometric and program representations. However, the lack of labeled data in CAD's native format, i.e., the parametric boundary representation (B-Rep), poses an obstacle at present difficult to overcome. Several datasets of mechanical parts in B-Rep format have recently been released for machine learning research. However, large-scale databases are mostly unlabeled, and labeled datasets are small. Additionally, task-specific label sets are rare and costly to annotate. This work proposes to leverage unlabeled CAD geometry on supervised learning tasks. We learn a novel, hybrid implicit/explicit surface representation for B-Rep geometry. Further, we show that this pre-training both significantly improves few-shot learning performance and achieves state-of-the-art performance on several current B-Rep benchmarks.

1. Introduction

Almost every human-made object that exists today started its life as a model in a CAD system. As the prevalent method of creating 3D shapes, repositories of CAD models are extensive. Further, CAD models have a robust structure, including geometric and program representations that have the potential to expose design and manufacturing intent. Learning from CAD data can therefore enable a variety of applications in design automation and design- and fabrication-aware shape reconstruction and reverse engineering.

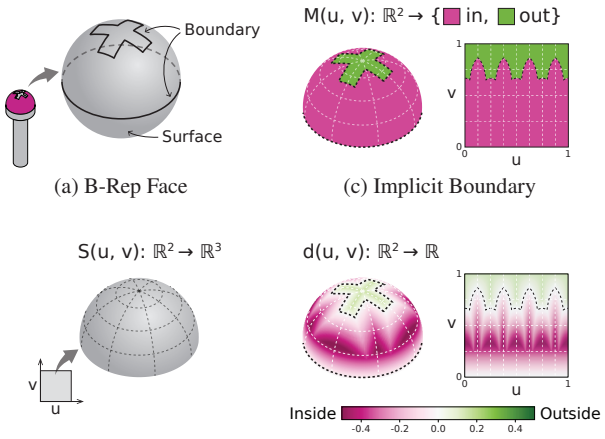
An important challenge in learning from CAD is that most of this data does not have labels that can be leveraged for inference tasks. Manually labeling B-Rep data is time consuming and expensive, and its specialized format requires CAD expertise, making it impractical for large collections.

In this work we ask: how can we leverage large databases of *unlabeled* CAD geometry for analysis and modeling tasks that typically require labels for learning?

Our work is driven by a simple, yet fundamental observation: the CAD data format was not developed to enable easy visualizing or straightforward geometric interpretation: it is a format designed to be compact, have infinite resolution, and allow easy editing. Indeed, CAD interfaces consistently run sophisticated algorithms to convert the CAD representation into geometric formats for rendering. Driven by this observation, *our key insight is to leverage large col-*

lections of unlabeled CAD data to learn to geometrically interpret the CAD data format. We then leverage the networks trained over the geometric interpretation task in supervised learning tasks where only small labeled collections are available. In other words, we use geometry as a model of *self-supervision* and apply it to *few-shot-learning*.

Specifically, we learn to rasterize local CAD geometry using an encoder-decoder structure. The standard CAD format encodes geometry as parametric boundary representations (B-Reps). B-Reps are graphs where the nodes are parametric geometry (surfaces, curves, and points) and edges denote the topological adjacency relationships between the geometry. Importantly, the parametric geometry associated with each node is *unbounded*, and bounds are computed from the topological relationships: curves bounding surfaces and points bounding curves. As shown in Figure 2, the geometry of a B-Rep face is computed by *clipping* the surface primitive to construct a surface patch, where the clipping mask is constructed from adjacent edges.



(a) B-Rep Face (c) Implicit Boundary

(b) Explicit Surface (d) Implicit Boundary (SDF)

Figure 2. A B-Rep face (a) is a surface patch cut from a geometric primitive surface (b). The adjacent edges define a clipping mask (c), which we learn an SDF (d).

Thus, B-reps are constructed piecewise by *explicitly* defined surfaces with *implicitly* defined boundaries. This observation drives our proposed learning architecture, which reconstructs faces by jointly decoding the explicit surface parameterization as well as the implicit surface boundary. Our proposed encoder uses message passing on the topological graph to capture the boundary information to encode B-Rep faces. To handle graph heterogeneity (nodes comprised of faces, edges, and, vertices), we use a hierarchical message passing architecture inspired by the Structured B-Rep GCN [16]. Our decoder uses the learned embeddings as latent codes for two per-face neural function evaluators: one mapping from $\mathbb{R}^2 \rightarrow \mathbb{R}^3$ that encodes the face's parametric surface (Figure 2 (b)), and one mapping $\mathbb{R}^2 \rightarrow \mathbb{R}$ that encodes the face's boundary as a signed distance field (SDF) *within* the parametric surface (Figure 2 (c,d)).

We apply our proposed model of B-Rep self-supervision to learn specialized B-Rep tasks from very small sets of labeled data—10s to 100s of examples vs 10k to 100k. To do this, we use the embeddings learned on self-supervision as input features to supervised tasks. We evaluate our approach on three tasks and datasets from prior work [3, 6, 21] and validate our findings across varying training set sizes. We show that our model consistently outperforms prior supervised approaches, significantly improving performance on smaller training sets. By using less data, our approach also proves substantially faster to train, making possible applications that depend on training speed. We believe that our differentiable CAD rasterizer paves the way to many exciting future applications, and show one possibility by prototyping a reverse engineering example.

2. Related Work

Learning from CAD Collections. Recent interest in learning from CAD data has grown due to advances in machine learning (ML) and the release of large CAD datasets, including collections of B-reps and program structure [20, 37], CAD sketches [31], and CAD assemblies [16]. Prior work leveraged such collections for diverse applications, including segmentation [6], classification [3], assembly suggestions [16, 36], and generative design of CAD sketches [10, 28, 31, 32], B-Reps [13, 14], and CAD programs [37, 38].

However, a fundamental gap separates the capabilities shown in past work and real-world applications that can transform CAD design, such as auto-complete modeling interfaces or reconstruction of complex geometries: the lack of task-specific labels in large datasets. For example, most prior work leveraged the Onshape Public dataset [20], which generally contains designs created by novice CAD users and so does not capture the design process of CAD experts. The Fusion 360 dataset [37] is significantly smaller than Onshape's; other public resources, such as GrabCAD [1], contain designs from multiple CAD systems that are mostly unlabeled. In this work, we advance the development of new CAD learning applications by proposing a unique direction for leveraging unlabeled data in the supervised learning of CAD geometry.

Learning on B-Reps. A CAD B-Rep is a specialized data structure that encodes solid geometry as a graph of parametric shapes and their topological relationships; it has the advantages of arbitrary spatial resolution and easy programmatic editability. B-Reps can be exported to other common geometric representations, such as polygonal meshes, using geometric CAD kernels [2]. Several techniques for learning on B-Reps use message passing networks. BRepNet [21]

and UV-Net [15] create a reduced graph of B-Rep faces, while the Structured B-Rep GCN (SB-GCN) [16] proposes a hierarchical structure over the four classes of topological entities (faces, loops, edges, and vertices). However, these methods require a CAD kernel in the loop to generate the features from the B-Rep format and large labeled sets for learning. While we use a CAD kernel to train our encoder-decoder on unlabeled data, our task-specific networks can be trained on small sets of labeled data without requiring a CAD kernel at inference time. More recent applications of machine learning to B-Rep structures have focused on generation [13, 14, 39].

Neural Shape Representations. Neural shape generation is an active research area with a large number of representations used by prior techniques. Some methods operate over a fixed discretization of the domain into points [4, 9], voxel grids [5, 24], or vertex coordinates of a mesh template [33]. Due to irregularity of geometric data, functional representation is often used to learn to represent shapes as continuous functions, such as surface atlases [12, 40] or signed distance fields defined over a volume [7, 25, 29]. In this work we chose to use functional representation of the output shape as a neural occupancy and a 3D mapping function over UV domains. Functional representation is well-suited for heterogeneous geometry with varied levels of detail since it does not require a fixed sampling rate to be chosen. Furthermore, our representation, which combines implicit fields and atlas-like embeddings, directly produces a surface and can be easily supervised with the ground truth B-Rep data.

Few-Shot Learning. Performance of strongly supervised methods is commonly hampered by the lack of training data. Few-shot learning techniques often rely on learning rich features in a self-supervised fashion and then using a few examples to adapt these features to a new task [35]. One common self-supervision strategy is to withhold some data from the original input and train a network to predict it. For example, one can remove color from images and train a network to colorize [22, 41], remove part of an image and train a network to complete it [30], randomly perturb orientation and predict the upright position [11], or randomly shuffle patches and predict their true ordering [8, 26]. Another commonly used tool is auto-encoders, which encode input to a lower-dimensional space and then attempt to reconstruct it with a decoder [18, 19]. Our approach is closest to that of auto-encoders, except our input and output are in two different representations: CAD B-Reps and surface rasterizations. This lets us learn features related to the actual 3D geometry.

3. Geometric Self-Supervision

Our goal is to learn relevant features of CAD B-Reps that can be used on many different modeling and analysis tasks. We use an encoder-decoder architecture on B-Rep surfaces to learn a latent space of relevant features. Based on the insight that the learned features should include a geometric understanding of CAD shapes, we train our encoder-decoder to learn to rasterize CAD models. We further choose to learn this embedding at the face level as opposed to learning a feature per part. This is driven by our application domain, where tasks typically require an understanding of local topological information. Figure 1 shows our approach at a high-level.

Decoder. As noted previously, the format we selected for decoder output is driven by the observation that B-Reps are compositions of explicitly represented surfaces with implicitly represented boundaries. For example, the geometry associated with faces are unbounded parametric surfaces $S : \mathbb{R}^2 \rightarrow \mathbb{R}^3$, and the boundary of such surfaces is captured by the geometry of the neighboring edges. Edges, in turn, are defined by unbounded parametric curves bounded by neighboring vertices. Vertices are represented as points. In addition to domain parameters (u,v), parametric geometry has other, fixed parameters (like radius or cone angle), which we call *shape parameters*. A bounded surface is called a *clipped surface*, and the bounding function, which is defined *implicitly* by the bounding topology, is its *clipping mask*, illustrated in Figure 2 (c).

For self-supervision, we must choose an output geometric representation for surface patches with boundary. Bounded patches lack a natural parameterization and so are not suitable to learn directly as parametric functions. They are also difficult to represent as neural implicits [27]. However, the clipping function itself defines a closed region *within* the parameterization of the supporting surface, which is a function that lends itself well to implicit representation in 2D. We therefore choose to learn an implicit function of the clipping region as a 2D signed distance function. Since this representation relies on an explicit surface parameterization, we also learn the supporting surface parameterization. Crucially, this does not require parameterizing a boundary.

We choose to use a conditional neural field as our decoder since this representation can capture both explicit and implicit geometry. Specifically, the explicit parametric surface is an $\mathbb{R}^2 \rightarrow \mathbb{R}^3$ function, mapping (u, v) coordinates of a face's supporting surface to the 3D position of that point (x, y, z). The clipping mask encoded as an SDF over the parametric domain is a function $\mathbb{R}^2 \rightarrow \mathbb{R}$, which maps the same (u, v) coordinates to a value (d) measuring the signed distance to the boundary. We combine these two function

to learn a single field that maps $\mathbb{R}^2 \rightarrow \mathbb{R}^4$. We parameterize this field as a 4-layer, fully connected ReLU network, where the input coordinates and conditioning vector are concatenated to the output at every hidden layer (similar to DeepSDF [29], see our supplemental material for a full definition). The conditioning vector is the output of our face encoder described below. We call the evaluation of this field "rasterization" because raster sampling the field and filtering by d yields a 3D surface rasterization.

To normalize the field input range and constrain the uv-space, we must sample while rasterizing. We reparameterize the uv-space of each surface prior to training so that the clipping mask fits within the unit square ($[0, 1]^2$). In this way, our encoder-decoder is learning the explicit surface, the implicit boundary, and the support range of the implicit boundary mask.

Encoder. Our encoder design is driven by the same observation, i.e., that B-Reps are compositions of explicitly represented surfaces with implicitly represented boundaries. For the encoder, this means that we must capture adjacent topological entities. We propose to encode this using message passing on the topological graph.

As Figure 3 shows, the B-Rep topological graph has a hierarchical structure; high-dimensional topological elements are adjacent only to the immediate lower dimensional entities: faces are adjacent to their bounding edges, which are adjacent to their end-point vertices. Driven by this observation, we use as our face-encoder a hierarchical message passing network inspired by SB-GCN [16], a graph convolutional network that leverages the hierarchical structure of B-Reps to learn embeddings for each topological entity.

Our encoder is structured as two graph message passing layers, first aggregating vertex information for the edges they bound, and then this bounded edge information for the faces. Crucially, we need to (1) support both node and adjacency level features in our message passing – since whether a vertex is the start or end of an edge, and whether an edge is to the left or right of a face interior is critical to correctly reproducing that edge – and (2) support highly variable node degree since faces vary widely in the number of bounding faces. To achieve these goals we use multi-headed additive graph attention [34] with graph-adjacency features.

Existing B-Rep representation learning networks [15, 16, 21] use evaluated geometry information as input features in addition to pure parametric geometry, such as surface bounding boxes and areas. These features are computed by a CAD kernel, i.e., modeling software that understands how to construct and evaluate CAD geometry. Since we want to force our network to learn the evaluation function of a CAD kernel as well as be fully differentiable back to the shape parameters, we only use the parametric geometry definitions as input features. Please see our supplemental material for the full details of our encoder architecture and input encodings.

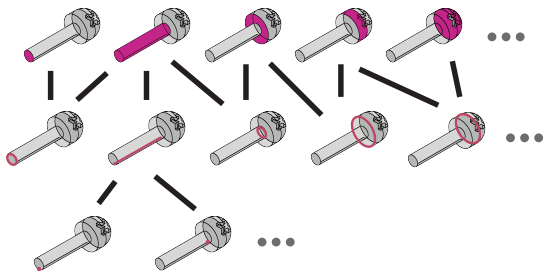


Figure 3. A boundary representation consists of parametric surface patches that are bounded by parametric edges, which in turn are bounded by vertices. The relationship between these entities forms a hierarchical graph.

Training. We trained our encoder-decoder to minimize L^2 losses at randomly sampled uv-points on each face's supporting surface. We randomly sampled in the re-parameterized uv-square $(u, v) \in [-0.1, 1.1]^2$ to ensure positive and negative SDF samples were collected, biased to preferentially sample near clipping boundaries (see supplemental for details).

We trained our geometric self-supervision over 23266 shapes from the training set of the Fusion 360 Gallery Segmentation dataset [37]. Optimization was performed using the Adam optimizer [17] with a learning rate of .0005. For our experiments, we used an embedding size of 64 for all message passing layers, 2 attention heads for the vertex-to-edge layer, 16 attention heads for the edge-to-face layer, and a hidden size of 1024 for all self-supervision decoder layers.

4. Few-Shot Learning

In typical task-specific applications, it is challenging to find or construct large collections of labeled CAD data. Therefore, we propose leveraging our rich latent space to enable supervised learning over very small collections. In addition to enabling training on scant data, our approach also ensures that we do not need to use computationally expensive CAD kernel functionality to generate features at inference time.

We propose two few-shot learning setups. The first consists of B-Rep segmentation tasks, which assign a task-specific label to each face of a B-Rep, e.g., the type of machining technique that can be used to construct that surface. The second consists of B-Rep classification tasks, which assign a label to an entire B-Rep, e.g., the mechanical function of that part (gear, bracket, etc.). We frame both setups as multi-class classification problems. See our supplemental

material for full network specifications.

B-Rep Segmentation Network. Since our encodings are learned at the face level, they can be used directly as input for face-level predictions. Classification of a face often depends on its context within a part; we therefore use a small message passing network to capture this context, i.e., a 2-layer Residual MR-GCN [23]. We use pre-computed face embeddings from our geometric self-supervised learning as node features and face-face adjacency as edges. We construct this graph by removing vertex nodes from the B-Rep graph and contracting edge nodes, preserving multiple edges (if any) between faces. Output predictions for each face are then made with a fully connected network with two hidden layers.

B-Rep Classification Network. While we do not have latent codes for entire B-Rep shapes, we take a similar approach to previous B-Rep learning architectures and max-pool learned features for each face. To do this, we project each face's embedding into a new vector for pooling using a fully connected layer, followed by a second projection of the pooled part features into the prediction output space.

5. Results

We validate the application of our proposed approach by applying our method to three few-shot learning tasks. We further evaluate our method by analyzing rasterization results.

5.1. Construction-Based Segmentation

The first task we apply our method to is segmentation of B-Rep geometry by the modeling operation used to construct each face (extrusion, revolution, chamfer, etc.). This requires the model to understand how geometry is constructed in CAD software. For this task we use 27450 parts from the the Fusion 360 Gallery dataset, a collection of user-constructed parts annotated with one of 8 face construction operations on each face [21].

We first pre-trained our geometric self-supervision network over this dataset without labels, then trained our face-level prediction network using the face embeddings from the self-supervision. Figure 4 shows some classification results. Our method achieves 65% accuracy after seeing just 10 training examples, 79% after 100, and is 96% accurate when given all 23266 examples in the training set.

We compare our method to three network architectures for B-Rep learning: SB-GCN [16], BRepNet [21], and UV-Net [15]. SB-GCN is a hierarchical message passing network that incorporates information from all three dimensions of B-Rep geometry (faces, edges, and vertices), and uses an intermediate *loop* layer to aggregate closed paths

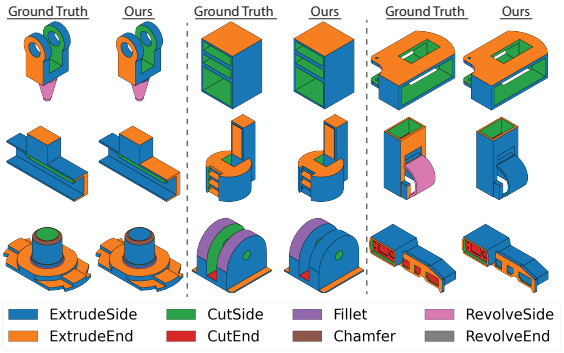


Figure 4. Segmentation example results for the Fusion 360 gallery task. The CAD operation that created each face is indicated by color. (Left) Our model's prediction. (Right) The ground truth.

of edges, and uses both the parametric geometry definitions and kernel computed features (bounding boxes and mass properties) as input features. BRepNet defines a convolution operator relative to co-edges in a B-Rep structure and uses parametric face and edge types as well as concavity and edge length features. UV-Net is a message passing network between adjacent faces; it uses grid-sampled points of faces and edges as its features. For UV-Net, we train an end-to-end version, as well as a self-supervised version using the part augmentation supervision and contrastive loss described in [14] (UV-Net-SS). For UV-Net-SS, we use the same task-specific network as our method to fairly compare the self-supervision. We trained and tested each network on random subsets of the training data that ranged from 10 to 23266 examples. We repeated this 10 times for each training set size (all three methods were given the same training sets).

Figure 5 plots the average accuracy for each training size across these runs with a bootstrapped 95% confidence interval. Our method outperforms the baselines for all dataset sizes, doing so by a significant margin in the few-shot regime. It significantly outperforms UV-Net with self-supervision, indicating that our geometric self-supervision is much more effective than a contrastive loss setup. It also has a smaller confidence interval, indicating that pre-training on our rasterization task makes classifications more robust to choice of training example.[1] In addition to these quantitative results, Figure 6 compares classifications at the 100 example level. Our method starts generalizing with 10s to 100s of examples, whereas the baselines fail to generalize and usually learn the most frequent label at this data scale.

We further note that training our method is significantly faster to train than the baselines (see Figure 7). This raises the possibility of segmentation tasks trained on-the-fly to enable predictors to be trained and deployed during the

[1] BRepNet's very small confidence interval at small training set sizes is an artifact of it predominantly or entirely predicting one class.

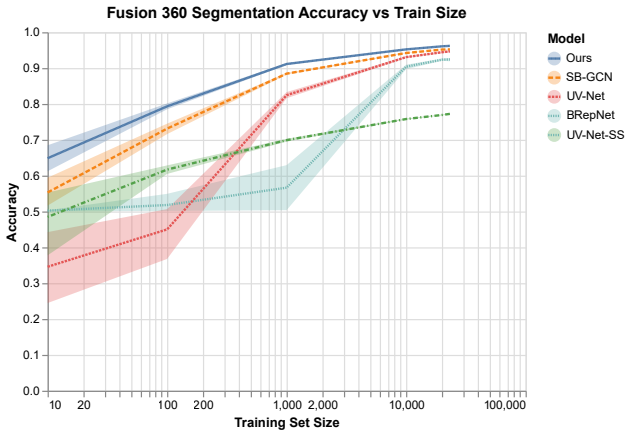


Figure 5. Comparison of our method versus SB-GCN, UV-Net, BRepNet, and Self-Supervised UV-Net (UV-Net-SS) for construction-based segmentation on the Fusion 360 Gallery Segmentation dataset. The mean accuracy across 10 training runs is plotted with a bootstrapped 95% confidence interval.

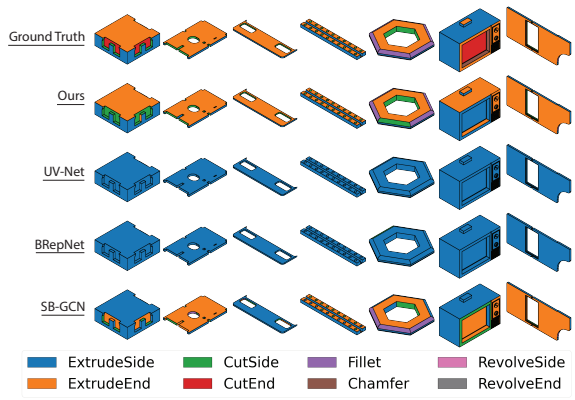


Figure 6. Few shot segmentation results from our method versus baseline models on the Fusion 360 Gallery task, trained over just 100 models. Face color indicates the modeling operation used to create a face. Our model performs significantly better in the low data regime, where it can differentiate operations. However, the baselines largely guess the most common operation, which is ExtrudeSide.

modeling process.

5.2. Manufacturing-Driven Segmentation

The second segmentation problem we considered is face segmentation, which classifies how each face is manufactured. We evaluated this on the MFCAD dataset [6], a synthetic dataset of 15488 CAD models where each face is labeled with one of 16 types of machining feature (chamfer, through hole, blind hole, etc.) The parts in this dataset are generated by applying machining operations to square stock and so consist entirely of planar faces and straight line edges. Compared to the Fusion 360 Gallery dataset,

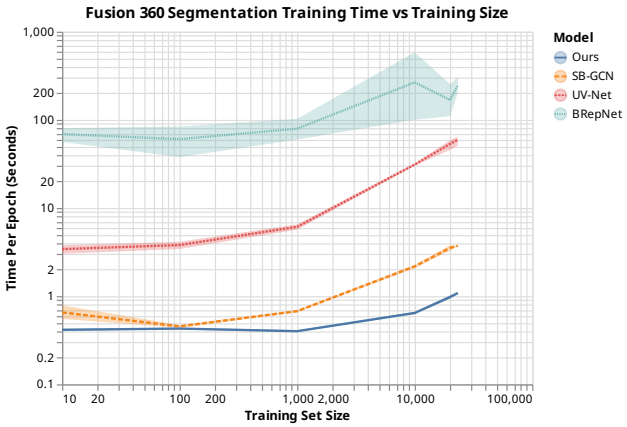


Figure 7. Training times on the Fusion 360 segmentation task. All models were trained on an NVIDIA RTX 2080 Ti, and reported time is the average time per epoch at each training set size.

this task has twice as many classes to choose from and relies more on neighborhood and boundary information since all faces are planar. It also shows the ability of our approach to generalize pre-trained features to out-of-distribution data since we use face codes pre-trained on Fusion 360 Gallery models (in this and all examples), which unlike MFCAD are all human-modeled.

We compare this task to the same four baselines, illustrated qualitatively in Figure 8 and quantitatively in Figure 9. We removed curvature features from BRepNet since they are universally zero on this dataset, and BRepNet requires all input features to have non-zero standard deviation. As before, our method outperforms the baselines at few-shot learning and achieves comparable accuracy at large data sizes (Ours, UV-Net, and BRepNet are essentially perfect given sufficient training data). UV-Net performs slightly better with more samples (1000); we hypothesize this is partially due to a domain gap between human-modeled parts of the Fusion 360 Gallery dataset we used in our self-supervised learning stage, and the synthetic parts of MFCAD, which might enable competing techniques to learn how to use features specific to the synthesis process to their advantage. The stability of our prediction accuracy as measured by confidence interval significantly exceeds both baselines.

5.3. Part Classification

In addition to segmentation, we also applied our method to part classification. For this, we used the FabWave [3] dataset, a hand-labeled subset of GrabCAD [1] that is categorized into classes of mechanical parts (gears, brackets, washers, etc.). Many parts within a category are parametric variations of each other. This task is important for understanding the function of mechanical parts in assemblies.

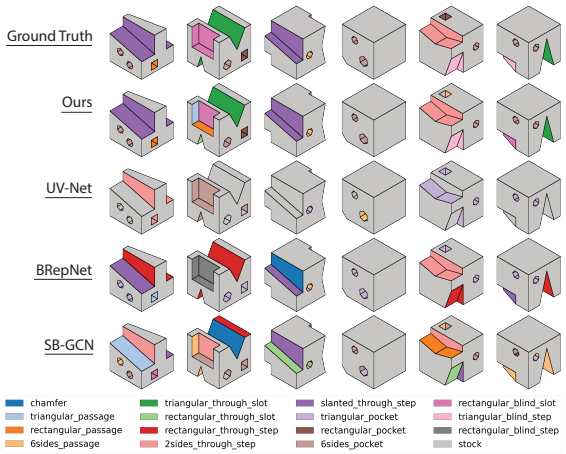


Figure 8. MFCAD few-shot segmentation comparison at 100 train samples. Top row shows ground truth labeling, followed by our, UV-Net, and BRepNet predictions.

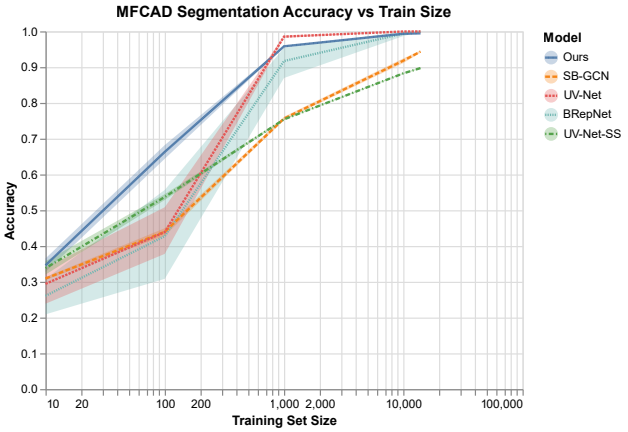


Figure 9. Comparison of our method to SB-GCN, UV-Net, BRepNet, and Self-Supervised UV-Net (UV-Net-SS) for manufacturing-based segmentation on the MFCAD dataset. The mean accuracy across 10 training runs is plotted with a bootstrapped 95% confidence interval.

As a baseline comparison, we compare only against SB-GCN and UV-Net since BRepNet does not support classification. Again using face codes pre-trained on Fusion 360 Gallery B-Reps, we train our B-Rep Classification Network over the 26 classes that have at least 3 examples compatible with our network (for train, test, and validation); we use stratified sampling to create training subsets that contain at least one part from each category in each split. Since UV-Net cannot be run on B-Reps that lack edges, we restricted our test set only to models with edges; we also removed 2 classes that UV-Net could not distinguish since they differ only in part orientation (our technique can differentiate these classes, so keeping them in the evaluation would both

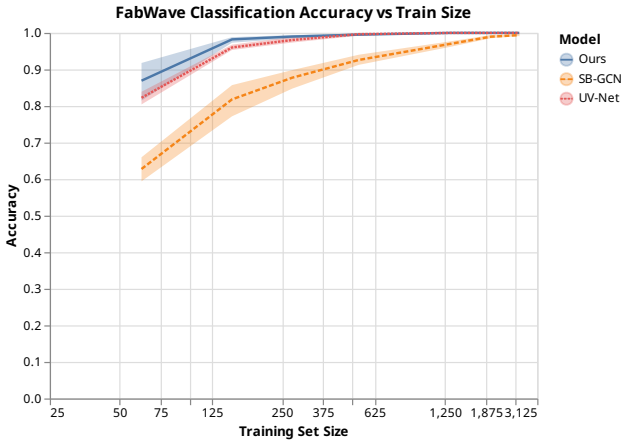


Figure 10. Comparison of our method to UV-Net for part classification on the FabWave dataset. The mean accuracy across 10 training runs is plotted with a bootstrapped 95% confidence interval.

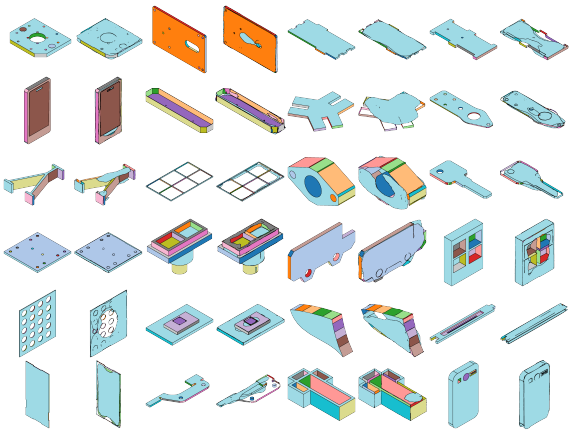


Figure 11. Shape reconstructions (right) from face embeddings on the Fusion 360 segmentation test set compared to ground truth (left). Each B-Rep face is given a unique color, which is consistent between ground truth and reconstruction.

improve our accuracy and decrease UV-Net's). Figure 10 shows the results. Our method slightly outperforms UV-Net across training sizes and significantly outperforms SB-GCN, and even achieves 100% accuracy at higher training set sizes.

5.4. Rasterization Evaluation

Figure 11 shows a collection of part renderings to illustrate qualitatively our reconstruction and classification results. It shows a gallery of rasterization results on unseen test examples from the Fusion 360 Gallery segmentation dataset. Reconstructions were created by sampling a 100x100 (u,v) grid in the range [-0.1,1.1] for each face to create a mesh for the supporting surface, then remov-

Task / Model	Training Set Size / Accuracy					
Fusion 360 Seg.	10	100	1000	10000	20000	23266
SVM	**0.66**	0.79	0.85	0.87	0.87	0.87
MLP	0.65	**0.80**	0.90	0.94	0.94	0.94
MP (Ours)	0.65	0.79	**0.91**	**0.95**	**0.96**	**0.96**
MFCAD	10	100	1000	10000	13940	–
SVM	**0.40**	0.51	0.56	0.57	0.57	
MLP	0.36	0.60	0.86	0.93	0.93	
MP (Ours)	0.35	**0.66**	**0.96**	**0.99**	**0.99**	

Table 1. Segmentation ablations. Reported face segmentation accuracy shows the mean of 10 runs at each dataset size with the train set subset at different random seeds (each model sees the same 10 random subsets). Models were selected by best validation loss on a random 20% validation split. Bold indicates the best accuracy at each train size for each task.

ing mesh vertices outside the predicted clipping plane SDF. Supporting surface reconstructions are highly accurate, as are the clipping masks for typical shapes with a single boundary. Complex interior and exterior boundaries of clipping planes are sometimes predicted inaccurately.

5.5. Ablations

Self-Supervision Ablations. In addition to the network described in Section 3, we also tried using a truncated SB-GCN (only its upwards pass) as the encoder network, using the same shape parameter input features. We evaluated both explicit surface and SDF accuracy; our encoder outperforms SB-GCN by 27% in the former metric and 31% in the latter.

Segmentation Ablations. We tried three types of face-level prediction networks using our self-supervised face embeddings: directly classifying faces from the embedding using a linear support vector machine (SVM), using a multi-layer perceptron (MLP), or using the message passing scheme described in Section 4 (MP) to test if neighborhood context is necessary. Table 1 shows the results of these experiments. All methods performed similar with little training data, but with more data we observed up to 36% improvement from SVM to MLP, and up to additional 10% from MLP to MP. At very low training sizes, using an SVM did outperform other methods, indicating that for certain very-low data tasks it may be a better way to apply geometrically self-supervised features. We chose to use message passing in our comparisons because it performed the best across a range of training set sizes.

Classification Ablations. We also tested adding message passing layers prior to pooling for the classification network, but found that this additional complexity did not yield any improvement.

Further details about our ablations experiments and full result tables are in our supplemental material.

5.6. Limitations and Future Work

Our work has three main limitations. The first is that we currently only support geometry with a fixed number of shape parameters (e.g. no construction geometry or B-splines), allowing us to support 77% of the Fusion 360 Segmentation dataset. The limitation is due to the choice to use fixed-size vectors as our input feature encoding for simplicity. It could be alleviated by using a sequence or tree encoder to compute fixed-size embeddings for the generic functional geometric expressions of each B-Rep topology.

The second limitation is that we self-supervise only on local information. This means that we rely on additional message passing layers in our task-specific decoders to gather neighborhood features. Finally, we create encodings only for faces, which limits the kinds of tasks we can learn. Adding a second decoder and loss term to rasterize edges could extend this work to edge-based tasks. We did not add this complexity since there are currently no edge-specific tasks in the literature to compare against.

Improving self-supervision accuracy will be a fruitful direction for future work. The accuracy of downstream learning tasks is correlated with the accuracy of our model's rasterization (see supplemental). Thus, improvements in rasterization performance should yield improvements in classification performance in both few-shot and large data regimes.

Improving rasterization performance could also unlock future applications in reverse engineering. Operating our self-supervision network as a rasterizer creates, in effect, a differentiable CAD renderer, which could be used for gradient-based optimization of B-Rep *shape parameters*. A prototype shape fitting application (see supplemental) showed promise on simple shapes, but struggles on more complex shapes, which may overcome by improved rasterization performance.

6. Conclusion

In this work we learn a spatial embedding of B-Reps and apply it to few-short learning on supervised tasks. We validate that this approach is effective compared to prior work on supervised learning. Our results show comparable results on large training sets and significantly better performance on smaller sets. By experimenting over three different tasks and datasets, we posit that this method will be widely applicable to a plethora of CAD applications. Being faster to train can enable many new applications in this domain, particularly user-guided annotation for customized predictions. Finally, our method enables a fully differential embedding of B-Rep geometry compared to prior work that required non-differentiable CAD kernels, paving the way for exciting future work on CAD optimization and reverse engineering.

References

[1] Grabcad. https://grabcad.com/. Accessed: 2022-05-19. 2, 6

[2] Siemens. parasolid cad kernel. https://www.plm.automation.siemens.com/global/en/products/plm-components/parasolid.html. Accessed: 2022-05-19. 2

[3] *Development of a Pilot Manufacturing Cyberinfrastructure With an Information Rich Mechanical CAD 3D Model Repository*, volume Volume 1: Additive Manufacturing; Manufacturing Equipment and Systems; Bio and Sustainable Manufacturing of *International Manufacturing Science and Engineering Conference*, 06 2019. V001T02A035. 2, 6

[4] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J Guibas. Learning representations and generative models for 3d point clouds. *arXiv preprint arXiv:1707.02392*, 2017. 3

[5] André Brock, Theodore Lim, James M. Ritchie, and Nick Weston. Generative and discriminative voxel modeling with convolutional neural networks. *CoRR*, abs/1608.04236, 2016. 3

[6] Weijuan Cao, Trevor Robinson, Yang Hua, Flavien Boussuge, Andrew R. Colligan, and Wanbin Pan. Graph representation of 3d cad models for machining feature recognition with deep learning. volume Volume 11A: 46th Design Automation Conference (DAC) of *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 08 2020. 2, 6

[7] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[8] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015. 3

[9] Haoqiang Fan, Hao Su, and Leonidas Guibas. A point set generation network for 3d object reconstruction from a single image. *CVPR*, 2017. 3

[10] Yaroslav Ganin, Sergey Bartunov, Yujia Li, Ethan Keller, and Stefano Saliceti. Computer-aided design as language. *Advances in Neural Information Processing Systems*, 34, 2021. 2

[11] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR 2018*, 2018. 3

[12] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. Atlasnet: A papier-mache approach to learning 3d surface generation. *arXiv preprint arXiv:1802.05384*, 2018. 3

[13] Hao-Xiang Guo, Shilin Liu, Hao Pan, Liu Yang, Xin Tong, and Baining Guo. Complexgen: Cad reconstruction by b-rep chain complex generation. *ACM Transactions on Graphics (TOG)*, 39(4):106:1–106:14, 2022. 2, 3

[14] Pradeep Kumar Jayaraman, Joseph G Lambourne, Nishkrit Desai, Karl DD Willis, Aditya Sanghi, and Nigel JW Morris. Solidgen: An autoregressive model for direct b-rep synthesis. *arXiv preprint arXiv:2203.13944*, 2022. 2, 3, 5

[15] Pradeep Kumar Jayaraman, Aditya Sanghi, Joseph G Lambourne, Karl DD Willis, Thomas Davies, Hooman Shayani, and Nigel Morris. Uv-net: Learning from boundary representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11703–11712, 2021. 3, 4, 5

[16] Benjamin Jones, Dalton Hildreth, Duowen Chen, Ilya Baran, Vladimir G. Kim, and Adriana Schulz. Automate: A dataset and learning approach for automatic mating of cad assemblies. *ACM Transactions on Graphics*, 40(6), dec 2021. 2, 3, 4, 5

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[18] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pages 3581–3589, 2014. 3

[19] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 3

[20] 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 *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9593–9603, Long Beach, CA, USA, June 2019. IEEE. 2

[21] Joseph G. Lambourne, Karl D. D. Willis, Pradeep Kumar Jayaraman, Aditya Sanghi, Peter Meltzer, and Hooman Shayani. BRepNet: A topological message passing system for solid models. *arXiv:2104.00706 [cs]*, Apr. 2021. arXiv: 2104.00706. 2, 4, 5

[22] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In *European Conference on Computer Vision*, pages 577–593. Springer, 2016. 3

[23] Changjian Li, Hao Pan, Adrien Bousseau, and Niloy J Mitra. Sketch2cad: Sequential cad modeling by sketching in context. *ACM Transactions on Graphics (TOG)*, 39(6):1–14, 2020. 5

[24] Jerry Liu, Fisher Yu, and Thomas Funkhouser. Interactive 3d modeling with a generative adversarial network. *International Conference on 3D Vision (3DV)*, 2017. 3

[25] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019. 3

[26] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016. 3

[27] David Palmer, Dmitriy Smirnov, Stephanie Wang, Albert Chern, and Justin Solomon. DeepCurrents: Learning implicit representations of shapes with boundaries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3

[28] Wamiq Para, Shariq Bhat, Paul Guerrero, Tom Kelly, Niloy Mitra, Leonidas J Guibas, and Peter Wonka. Sketchgen: Generating constrained cad sketches. *Advances in Neural Information Processing Systems*, 34, 2021. 2

[29] Jeong Joon Park, Peter Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. *CVPR*, 2019. 3, 4

[30] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016. 3

[31] Ari Seff, Yaniv Ovadia, Wenda Zhou, and Ryan P. Adams. SketchGraphs: A Large-Scale Dataset for Modeling Relational Geometry in Computer-Aided Design. *arXiv:2007.08506 [cs, stat]*, July 2020. arXiv: 2007.08506. 2

[32] Ari Seff, Wenda Zhou, Nick Richardson, and Ryan P Adams. Vitruvion: A generative model of parametric cad sketches. *arXiv preprint arXiv:2109.14124*, 2021. 2

[33] Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shihong Xia. Variational autoencoders for deforming 3d mesh models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 3

[34] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017. 4

[35] Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Comput. Surv.*, 53(3), jun 2020. 3

[36] Karl DD Willis, Pradeep Kumar Jayaraman, Hang Chu, Yunsheng Tian, Yifei Li, Daniele Grandi, Aditya Sanghi, Linh Tran, Joseph G Lambourne, Armando Solar-Lezama, et al. Joinable: Learning bottom-up assembly of parametric cad joints. *arXiv preprint arXiv:2111.12772*, 2021. 2

[37] Karl D. D. Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G. Lambourne, Armando Solar-Lezama, and Wojciech Matusik. Fusion 360 Gallery: A Dataset and Environment for Programmatic CAD Reconstruction. *arXiv:2010.02392 [cs]*, Oct. 2020. arXiv: 2010.02392. 2, 4

[38] Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6772–6782, 2021. 2

[39] Xiang Xu, Karl D.D. Willis, Joseph G Lambourne, Chin-Yi Cheng, Pradeep Kumar Jayaraman, and Yasutaka Furukawa. SkexGen: Autoregressive generation of CAD construction sequences with disentangled codebooks. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 24698–24724. PMLR, 17–23 Jul 2022. 3

[40] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 3, 2018. 3

[41] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016. 3