SO(3)-Equivariant Learning on CAD Boundary Representations

Matteo Ballegeer*
Ghent University
CVAMO, FlandersMake@UGent
matteo.ballegeer@ugent.be

Dries F. Benoit
Ghent University
CVAMO, FlandersMake@UGent
dries.benoit@ugent.be

Abstract

Learning on boundary representations (B-reps) requires robustness to the pose of a part, since CAD models are not stored in a canonical orientation. Existing B-rep networks either operate on world-frame coordinates and normals, and thus degrade under rotation, or encode each face in a local reference frame. Local frames yield invariance, but reduce inter-face messages to scalars and discard the relative orientation of adjacent faces. We propose SO3brep, an $\mathrm{SO}(3)$-equivariant graph neural network on the B-rep face-adjacency graph. Node features are computed in local frames and mapped into a shared frame by Wigner D-matrices. Clebsch–Gordan tensor products couple these features with the directions between adjacent faces, so that relative orientation is preserved throughout message passing and the output remains $\mathrm{SE}(3)$-invariant. Across three face-segmentation and two classification benchmarks with randomly rotated test parts, SO3brep achieves the highest accuracy among the compared B-rep networks and equivariant point cloud networks, with the fewest parameters. Its advantage is largest with few labeled parts, a setting typical of industrial CAD. Code is available at `https://github.com/UGent-CVAMO/so3brep`.

1 Introduction

Boundary representations (B-reps), the native format of CAD, describe a solid by the parametric faces that bound it (Weiler, 1986). Rather than converting the part to a mesh or point cloud, B-rep learning uses the face-adjacency graph (Ansaldi et al., 1985) to capture topology, while features on the nodes and edges encode the geometry of the surfaces and their shared curves (Jayaraman et al., 2021). With the increasing availability of CAD datasets (Koch et al., 2019; Lambourne et al., 2021), graph-based B-rep learning has been used for shape classification, face segmentation, manufacturing effort analysis, and B-rep generation (Jayaraman et al., 2021; Cao et al., 2020; Lambourne et al., 2021; Colligan et al., 2022; Ballegeer et al., 2026; Wu et al., 2024; Xu et al., 2024, 2025).

Most B-rep networks, however, read coordinates and surface normals in the world frame (Jayaraman et al., 2021; Wu et al., 2024), so their inputs change when the part is rotated. Yet the pose of a CAD part is a convention, not a property of the part. CAD systems leave the choice of up axis to the user, with both SOLIDWORKS and Fusion 360 allowing Y or Z (Dassault Systèmes, 2020; Autodesk, 2026), and a part saved inside an assembly carries the assembly's coordinates (Willis et al., 2022). Benchmarks conceal this sensitivity because their parts share a canonical pose. FoV-Net (Ballegeer and Benoit, 2026b) showed that world-frame networks collapse once the test parts are rotated, and achieves rotation invariance by expressing every face in its own local reference frame (LRF). This invariance has a price: each face is described in its own frame, so the scalar messages it sends discard the orientation relative to the receiving face (Lippmann et al., 2025).

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

An equivariant network instead rotates its features with the part and, for an invariant prediction, discards their directional components only at the readout. When every feature transforms as an irreducible representation (irrep) of $\mathrm{SO}(3)$, messages can carry vectors and tensors in one shared frame, so a face can compare the directions in which its neighbors lie (Thomas et al., 2018; Weiler et al., 2018). In graph networks, such steerable messages have outperformed invariant ones (Brandstetter et al., 2022; Batzner et al., 2022). Because the symmetry is built into the layers, the network does not have to learn rotations from augmented copies (Cohen and Welling, 2016; Thomas et al., 2018), and equivariant networks have been shown to be more data-efficient than both non-equivariant (Brehmer et al., 2025) and invariant ones (Batzner et al., 2022). Data efficiency matters for CAD, where labeled training sets are small, since labels require expert time and many part libraries are proprietary due to intellectual property concerns (Jones et al., 2023; Stjepandić et al., 2015).

We introduce **SO3brep**, an $\mathrm{SO}(3)$-equivariant graph neural network on the B-rep face-adjacency graph. Features computed in the local frame of each face's parameterization are rotated into a shared frame and coupled with inter-face directions through Clebsch–Gordan tensor products, making SO3brep, to our knowledge, the first B-rep network with equivariant rather than invariant message passing. On five CAD benchmarks with randomly rotated test parts, it achieves the highest accuracy with the fewest parameters, with the largest advantage when training data is scarce.

2 Related work

Learning on B-reps. Learning methods on B-reps take as input the part's face-adjacency graph, in which nodes are faces and edges are the curves they share (Cao et al., 2020). UV-Net (Jayaraman et al., 2021) encodes each face by a CNN on points and normals sampled over its UV domain and each edge by a 1D CNN on its curve; BRepNet (Lambourne et al., 2021) passes messages along oriented coedges. Later networks add face and edge attributes (Colligan et al., 2022; Wu et al., 2024; Zhang et al., 2024), graph attention (Lee et al., 2023) or transformers (Dai et al., 2025; Zou and Zhu, 2025), or pretrain on unlabeled parts (Jones et al., 2023; Lou et al., 2023). The same UV encodings are used for B-rep generation (Xu et al., 2024, 2025). Some of the added attributes, such as surface type and area, are rotation-invariant, but the UV grids that carry most of the geometry are sampled in the world frame, so the predictions change when the part is rotated (Ballegeer and Benoit, 2026b). Heidari and Iosifidis (2025) survey the field.

Rotation-invariant learning. Invariant networks remove the pose from their input through distances and angles (Deng et al., 2018; Li et al., 2021b), a canonical pose from PCA (Li et al., 2021a), averaging over a set of frames (Puny et al., 2022), or local reference frames (Chen et al., 2024). Fei and Deng (2024) survey these approaches for point clouds. For B-reps, FoV-Net (Ballegeer and Benoit, 2026b) takes the frame of each face from its parameterization and describes the surrounding solid by rays cast in that frame. Concurrently with this work, EquiCAD (Su et al., 2026) encodes face and edge samples with $\mathrm{O}(3)$-equivariant layers and pools them to invariant scalars before message passing. In both, faces exchange invariant scalars. Such scalars limit what local message passing can tell apart: descriptors built from distances and angles alone can coincide for distinct structures (Pozdnyakov and Ceriotti, 2022), and invariant message passing fails to distinguish some geometric graphs that equivariant message passing with higher-degree features separates (Joshi et al., 2023).

Equivariant geometric learning. Equivariant networks transform their features together with the input in every layer, so directional information is kept throughout. Group-equivariant and steerable CNNs build the symmetry into convolution (Cohen and Welling, 2016, 2017; Cohen et al., 2018; Weiler et al., 2018, 2021); tensor field networks and the $\mathrm{SE}(3)$-Transformer (Thomas et al., 2018; Fuchs et al., 2020) carry it to message passing on irreps coupled by Clebsch–Gordan tensor products (Geiger and Smidt, 2022). This design underlies interatomic potentials such as NequIP, MACE, and Equiformer (Batzner et al., 2022; Batatia et al., 2022; Liao and Smidt, 2023) and equivariant point cloud networks (Fuchs et al., 2020; Kim et al., 2024). Cheaper variants replace the tensor product by scalar-weighted vectors (Satorras et al., 2021; Deng et al., 2021) or reduce its cost (Passaro and Zitnick, 2023; Liao et al., 2024). To our knowledge, no B-rep network passes equivariant messages between faces.

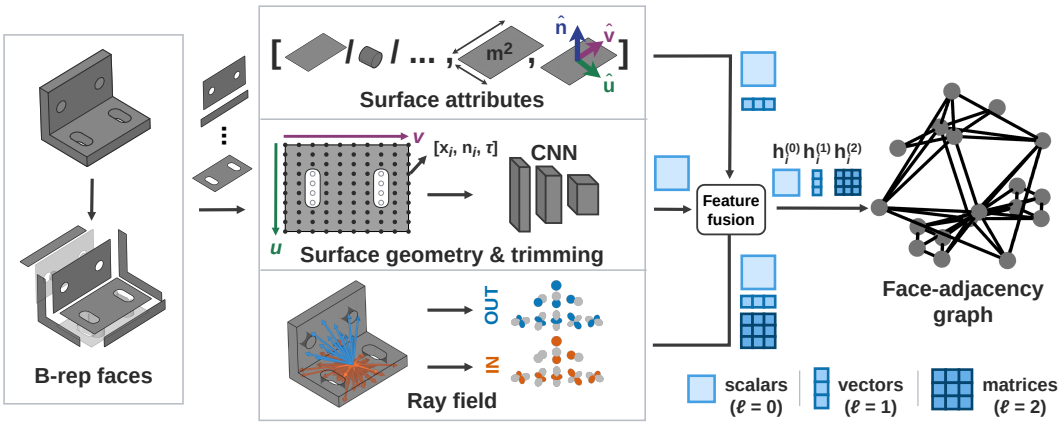

Figure 1: **Face feature construction.** Each B-rep face is initialized with features from three encoding paths before message passing. Surface attributes (ℓ=0, 1): surface-type and log-area scalars, plus the three LRF axes as ℓ=1 channels. Surface geometry and trimming (ℓ=0): a CNN on the UV grid, expressed in the LRF. Ray field (ℓ=0, 1, 2): a spherical-harmonic (SH) encoding of rays cast over two hemispheres, with ray directions fixed in the LRF. Feature fusion combines the scalars and projects all paths equivariantly to the GNN input features $\mathbf{h}_i^{(\ell)}$, with C_ℓ channels for each degree $\ell \in \{0, 1, 2\}$.

3 SO3brep

SO3brep is an $\mathrm{SO}(3)$-equivariant graph neural network on the B-rep face-adjacency graph. Each face is first described in its own local reference frame (Sec. 3.1). Node and edge features built from these frames (Secs. 3.2 and 3.3) are then expressed in one shared world frame, where equivariant message passing combines them (Sec. 3.4). An invariant readout makes the prediction independent of the pose. Positions enter only as differences, making the network translation-invariant as well.

Irreps and spherical harmonics. Every feature is an irrep of $\mathrm{SO}(3)$ of degree $\ell \le \ell_{\max} = 2$, a scalar, vector, or traceless symmetric matrix with $2\ell + 1$ components, on which a rotation $\mathbf{R}$ acts by the Wigner D-matrix $\mathbf{D}^{(\ell)}(\mathbf{R})$. A feature with C channels is an array $\mathbf{h}^{(\ell)} \in \mathbb{R}^{(2\ell+1)\times C}$ whose columns are the channels, rotated as $\mathbf{D}^{(\ell)}(\mathbf{R})\,\mathbf{h}^{(\ell)}$; linear maps across channels act from the right, $\mathbf{h}^{(\ell)}\mathbf{W}$, so they commute with this rotation and preserve equivariance. Directions are encoded by real spherical harmonics (SH) $Y^{(\ell)}(\hat{\mathbf{d}}) \in \mathbb{R}^{2\ell+1}$, which rotate as irreps: $Y^{(\ell)}(\mathbf{R}\hat{\mathbf{d}}) = \mathbf{D}^{(\ell)}(\mathbf{R})\,Y^{(\ell)}(\hat{\mathbf{d}})$.

3.1 Local reference frames

SO3brep builds its node features in a local reference frame (LRF) per face, a coordinate system anchored at the face that does not depend on the pose of the part. Each B-rep face is the image of a trimmed domain $\Omega_i \subset \mathbb{R}^2$ under a parametric surface $\mathbf{x}_i$ with parameters (u, v). We follow Ballegeer and Benoit (2026b) and anchor a local frame at the center $(\bar{u}_i, \bar{v}_i)$ of the bounding box of Ω_i, mapped to 3D as $\mathbf{c}_i = \mathbf{x}_i(\bar{u}_i, \bar{v}_i)$. The CAD kernel provides canonical directions at this point: tangents aligned with (u, v) and a unit normal. These give an orthonormal basis:

$$\hat{\mathbf{u}}_i = \frac{\partial_u \mathbf{x}_i}{\|\partial_u \mathbf{x}_i\|}\bigg|_{(\bar{u}_i,\bar{v}_i)}, \quad \hat{\mathbf{n}}_i = \frac{\partial_u \mathbf{x}_i \times \partial_v \mathbf{x}_i}{\|\partial_u \mathbf{x}_i \times \partial_v \mathbf{x}_i\|}\bigg|_{(\bar{u}_i,\bar{v}_i)}, \quad \hat{\mathbf{v}}_i = \hat{\mathbf{n}}_i \times \hat{\mathbf{u}}_i. \tag{1}$$

These axes define a rotation matrix $\mathbf{R}_i = [\hat{\mathbf{u}}_i \mid \hat{\mathbf{v}}_i \mid \hat{\mathbf{n}}_i] \in \mathrm{SO}(3)$ whose columns are the frame axes in world coordinates. Rotating the part by $\mathbf{R}$ maps $\mathbf{R}_i \mapsto \mathbf{R}\mathbf{R}_i$, so the LRF coordinates of any world-frame vector $\mathbf{y}$ do not depend on the pose: $(\mathbf{R}\mathbf{R}_i)^\top \mathbf{R}\mathbf{y} = \mathbf{R}_i^\top \mathbf{y}$. Thus, $\mathbf{D}^{(\ell)}(\mathbf{R}_i)$ moves features from the LRF into the shared world frame (Sec. 3.2).

3.2 Node features

Each face's initial node state $\mathbf{h}_i$ combines three encoding paths (Fig. 1), computed from its LRF.

Surface attributes. We extract analytic scalar attributes directly from the CAD kernel: five surface-type flags (plane, cylinder, cone, sphere, torus), a flag for rational free-form surfaces, and the log-scaled face area, giving $\mathbf{s}_i^{\mathrm{CAD}} \in \mathbb{R}^7$. The three LRF axes $\hat{\mathbf{u}}_i, \hat{\mathbf{v}}_i, \hat{\mathbf{n}}_i$, expressed in the world frame, also enter the node state as $\ell{=}1$ irreps, so the GNN sees how each face is oriented.

Surface geometry and trimming structure. Following UV-Net (Jayaraman et al., 2021), we sample each face on a regular $10{\times}10$ UV grid, recording at each grid point (u_p, v_p) the 3D position $\mathbf{x}_i(u_p, v_p)$, the surface normal $\mathbf{n}_{ip}$, and a binary trim indicator τ_{ip}, giving a fixed-size representation of the surface geometry regardless of surface type. All quantities are expressed in the LRF by centering at $\mathbf{c}_i$ and rotating by $\mathbf{R}_i^\top$ (Ballegeer and Benoit, 2026b):

$$\mathbf{x}'_{ip} = \mathbf{R}_i^\top \big(\mathbf{x}_i(u_p, v_p) - \mathbf{c}_i\big), \qquad \mathbf{n}'_{ip} = \mathbf{R}_i^\top \mathbf{n}_{ip}, \tag{2}$$

so identical surfaces in different global orientations yield identical representations. The resulting $10{\times}10{\times}7$ array is processed by a small CNN and pooled to an embedding $\mathbf{s}_i^{\mathrm{UV}}$ of $\ell{=}0$ features.

Ray field. To describe the geometry around each face, we follow Ballegeer and Benoit (2026b) and cast $M{=}72$ rays over each of two Fibonacci hemisphere grids defined in the LRF. The outward hemisphere, centered on $\hat{\mathbf{n}}_i$, probes the surrounding geometry; the inward one, centered on $-\hat{\mathbf{n}}_i$, probes the solid interior. Each ray returns whether it hits the part, the hit distance, and the cosine between the ray and the surface normal at the hit point. Inside a closed solid every inward ray hits, so for inward rays the hit indicator is replaced by the proximity $1/(1 + \text{distance})$, which is large behind thin walls. A ray that misses returns zeros. Each face thus carries six signals, three per hemisphere; we write g_{isk} for the value of signal s on ray k, $\hat{\mathbf{d}}_{sk}$ for the direction of that ray in the LRF (shared by the three signals of a hemisphere), and $\bar{g}_{is}$ for the mean of the signal over its hemisphere. Because the directions are fixed, their SH encoding is a fixed linear map:

$$\mathbf{f}_{is}^{(0)} = \bar{g}_{is}, \qquad \mathbf{f}_{is}^{(\ell)} = \frac{1}{M} \sum_{k=1}^{M} \big(g_{isk} - \bar{g}_{is}\big)\, Y^{(\ell)}(\hat{\mathbf{d}}_{sk}), \quad \ell \in \{1, 2\}. \tag{3}$$

On a hemisphere $Y^{(\ell)}$ does not average to zero, so centering is needed for the $\ell{>}0$ coefficients to describe only how the signal varies with direction. This gives six degree-ℓ irreps $\mathbf{f}_i^{(\ell)} \in \mathbb{R}^{(2\ell+1)\times 6}$: $\ell{=}0$ holds the mean of each signal, $\ell{=}1$ its dominant direction, and $\ell{=}2$ its quadrupolar contrast. The $\ell{=}0$ coefficients and the norms of the $\ell{>}0$ coefficients feed a scalar MLP.

Feature fusion. The three scalar sources (analytic CAD attributes $\mathbf{s}_i^{\mathrm{CAD}}$, UV-grid embeddings $\mathbf{s}_i^{\mathrm{UV}}$, and the ray scalars from the MLP above) are each projected to width C_0 and fused by an MLP, producing the scalar node representation $\mathbf{h}_i^{(0)} \in \mathbb{R}^{C_0}$. The ray coefficients $\mathbf{f}_i^{(\ell)}$ with $\ell{>}0$ are expressed in each face's own LRF, so the same coefficient describes a different direction on every face. We rotate them into the world frame with the Wigner D-matrix of the LRF (Geiger and Smidt, 2022):

$$\tilde{\mathbf{f}}_i^{(\ell)} = \mathbf{D}^{(\ell)}(\mathbf{R}_i)\, \mathbf{f}_i^{(\ell)}. \tag{4}$$

The coefficients are fixed relative to the face and the LRF turns with the part, so $\tilde{\mathbf{f}}_i^{(\ell)}$ turns with the part as well. A linear layer per degree maps them, together with the LRF axes, to the vector and matrix node features $\mathbf{h}_i^{(1)}$ and $\mathbf{h}_i^{(2)}$.

3.3 Edge features

Each edge of the face-adjacency graph carries two descriptors of the faces it joins: directional features and rotation-invariant scalars. The directional features enter the Clebsch–Gordan (CG) tensor product, and the scalars set the strength of each message through the radial network (Sec. 3.4).

Directional edge features. Adjacent faces i and j are joined by an edge in each direction; along the edge ji, the sender j passes a message to the receiver i. We summarize the relative orientation of the two faces with four world-frame unit vectors (Fig. 2), built from the face centers $\mathbf{c}_i$, $\mathbf{c}_j$, the parametric midpoint $\mathbf{b}_{ji}$ of their shared boundary curve, and its endpoints $\mathbf{p}_{ji}$, $\mathbf{p}'_{ji}$:

$$\hat{\mathbf{r}}_{ij} = \frac{\mathbf{c}_j - \mathbf{c}_i}{d_{ji}}, \quad \hat{\mathbf{r}}_{jb} = \frac{\mathbf{b}_{ji} - \mathbf{c}_j}{\|\mathbf{b}_{ji} - \mathbf{c}_j\|}, \quad \hat{\mathbf{r}}_{bi} = \frac{\mathbf{c}_i - \mathbf{b}_{ji}}{\|\mathbf{c}_i - \mathbf{b}_{ji}\|}, \quad \hat{\mathbf{t}}_{ji} = \frac{\mathbf{p}'_{ji} - \mathbf{p}_{ji}}{\|\mathbf{p}'_{ji} - \mathbf{p}_{ji}\|}, \tag{5}$$

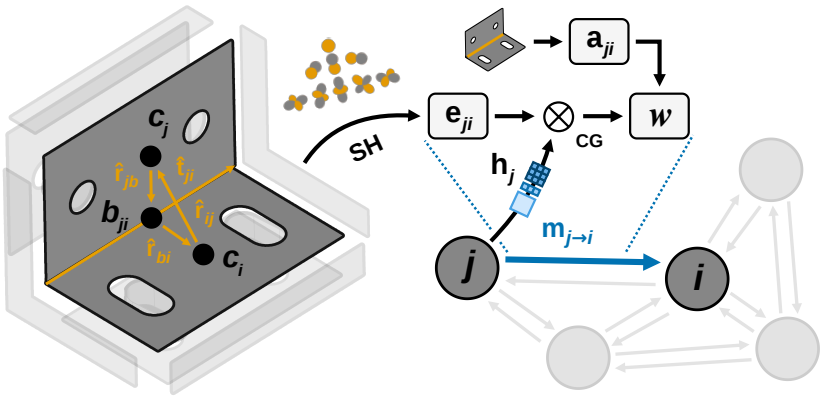

Figure 2: **Equivariant message passing.** Four unit vectors join the face centers $\mathbf{c}_i$, $\mathbf{c}_j$ and the shared edge: $\hat{\mathbf{r}}_{ij}$, $\hat{\mathbf{r}}_{jb}$, $\hat{\mathbf{r}}_{bi}$, and the edge chord $\hat{\mathbf{t}}_{ji}$. Their SH $\mathbf{e}_{ji}$ are coupled with the sender's irreps $\mathbf{h}_j$ by CG tensor products, so each message carries the relative directions of the two faces. A radial network maps the invariant edge scalars $\mathbf{a}_{ji}$ and the center distance to the gates of each coupling path.

with $d_{ji} = \|\mathbf{c}_j - \mathbf{c}_i\|$. The subscripts of $\hat{\mathbf{r}}$ give its direction: $\hat{\mathbf{r}}_{ij}$ points from the receiver center $\mathbf{c}_i$ to the sender center $\mathbf{c}_j$; $\hat{\mathbf{r}}_{jb}$ and $\hat{\mathbf{r}}_{bi}$ follow the message from $\mathbf{c}_j$ through the shared boundary to $\mathbf{c}_i$; and $\hat{\mathbf{t}}_{ji}$ is the chord of the boundary curve, signed by the curve's orientation. Where the two points of a difference coincide, as at the endpoints of a closed curve, the vector is set to zero, with $Y^{(\ell)}(\mathbf{0}) = \mathbf{0}$ for $\ell{>}0$. Their SH form the four channels of the edge irreps,

$$\mathbf{e}_{ji}^{(\ell)} = \left[\, Y^{(\ell)}(\hat{\mathbf{r}}_{ij}),\ Y^{(\ell)}(\hat{\mathbf{r}}_{jb}),\ Y^{(\ell)}(\hat{\mathbf{r}}_{bi}),\ Y^{(\ell)}(\hat{\mathbf{t}}_{ji}) \,\right] \in \mathbb{R}^{(2\ell+1)\times 4}, \quad \ell \in \{1, 2\}, \tag{6}$$

and the constant $\mathbf{e}_{ji}^{(0)} = Y^{(0)}$ is kept once.

Invariant edge scalars. The scalar descriptor $\mathbf{a}_{ji} \in \mathbb{R}^9$ holds the cosine $\hat{\mathbf{n}}_j \cdot \hat{\mathbf{n}}_i$ between the face normals and eight attributes of the shared curve: two convexity flags (concave, convex; both zero for smooth edges), the arc length (log-compressed), whether the curve is closed, and four curve-type flags (circle, non-rational B-spline, rational B-spline, line). The center distance is expanded in 16 Gaussian radial basis functions, $\mathrm{rbf}(d_{ji}) \in \mathbb{R}^{16}$ (Schütt et al., 2017).

3.4 Equivariant message passing

Node features are updated by L layers, each made of two residual blocks: equivariant message passing and an equivariant feed-forward network. Both blocks start with an RMS normalization (Zhang and Sennrich, 2019), taken per irrep block so that it is equivariant (Liao and Smidt, 2023).

Message construction. The CG tensor product $\otimes_{\mathrm{CG}}$ (Thomas et al., 2018; Geiger and Smidt, 2022) couples a degree-ℓ_1 sender feature $\mathbf{h}_j^{(\ell_1)}$ with a degree-ℓ_2 edge irrep $\mathbf{e}_{ji}^{(\ell_2)}$ into a degree-ℓ_3 output, which is possible for every pair in $\mathcal{I}_{\ell_3} = \{(\ell_1, \ell_2) \in \{0, \dots, \ell_{\max}\}^2 : |\ell_1 - \ell_2| \le \ell_3 \le \ell_1 + \ell_2\}$. Every channel of $\mathbf{h}_j^{(\ell_1)}$ is coupled with every channel of $\mathbf{e}_{ji}^{(\ell_2)}$. For instance, a sender vector and an edge direction ($\ell_1{=}\ell_2{=}1$) couple to their dot product ($\ell_3{=}0$), their cross product ($\ell_3{=}1$), and a traceless symmetric matrix ($\ell_3{=}2$). The message of degree ℓ_3 gates these products and mixes them:

$$\mathbf{m}_{j\to i}^{(\ell_3)} = \left(\bigoplus_{(\ell_1,\ell_2)\in\mathcal{I}_{\ell_3}} \left[\left(\mathbf{h}_j^{(\ell_1)} \otimes_{\mathrm{CG}} \mathbf{e}_{ji}^{(\ell_2)} \right)^{(\ell_3)} \odot\ w^{(\ell_1,\ell_2,\ell_3)}(\mathbf{a}_{ji}, \mathrm{rbf}(d_{ji})) \right] \right) \mathbf{W}^{(\ell_3)}, \tag{7}$$

where w is the radial network, an MLP on the invariant edge scalars, $\odot$ multiplies each channel by its gate, $\bigoplus$ concatenates the coupling paths along the channel axis, and $\mathbf{W}^{(\ell_3)}$ mixes channels. All CG coefficients are fixed; the learnable parameters are $\mathbf{W}^{(\ell_3)}$ and the radial network.

Aggregation with scalar attention. Messages from the neighborhood $\mathcal{N}(i)$ are aggregated using learned attention weights. For each of A heads, a scalar query $q_i^{(a)}$ and key $k_j^{(a)}$ are linear functions of the $\ell{=}0$ channels of nodes i and j, giving attention weights $\alpha_{ji}^{(a)} = \mathrm{softmax}_{j\in\mathcal{N}(i)}\big(q_i^{(a)} k_j^{(a)}/\sqrt{C_0}\big)$. Each head weights a disjoint, equal-width slice of the message channels within each irrep block. The aggregated features are then linearly mixed and combined with a residual connection:

$$\mathbf{h}_i^{(\ell)\prime} = \mathbf{h}_i^{(\ell)} + \left(\sum_{j\in\mathcal{N}(i)} \bigoplus_{a=1}^{A} \alpha_{ji}^{(a)}\, \mathbf{m}_{j\to i}^{(a,\ell)} \right) \mathbf{V}^{(\ell)}, \tag{8}$$

where $\mathbf{m}_{j\to i}^{(a,\ell)}$ denotes the a-th head slice of $\mathbf{m}_{j\to i}^{(\ell)}$, and $\mathbf{V}^{(\ell)}$ mixes channels.

Equivariant feed-forward block. A second residual block transforms each node equivariantly:

$$\mathbf{h}_i'' = \mathbf{h}_i' + \mathrm{FFN}(\mathbf{h}_i'). \tag{9}$$

The FFN applies a linear map across channels, the gated nonlinearity of Weiler et al. (2018), and a second linear map. The nonlinearity passes $\ell{=}0$ channels through SiLU and multiplies each $\ell{>}0$ channel by the sigmoid of an extra $\ell{=}0$ output of the first map (Geiger and Smidt, 2022). Every layer is equivariant: edge SH rotate with the part, CG products are equivariant, all gates and attention weights are invariant scalars on whole channels, and linear maps act across channels only.

Invariant readout. The $\ell{=}0$ components of the final node states are projected via a two-layer MLP to per-node descriptors $\mathbf{z}_i \in \mathbb{R}^{d_{\mathrm{out}}}$; higher-degree components are discarded to ensure an invariant output. The graph descriptor $\mathbf{z}_G$ concatenates the mean and max pooling of all $\mathbf{z}_i$. For classification, $\mathbf{z}_G$ is passed to an MLP head. For segmentation, each $\mathbf{z}_i$ is concatenated with $\mathbf{z}_G$ before a classification head that outputs per-face class probabilities.

4 Experiments

We evaluate SO3brep on three face-segmentation and two classification benchmarks in Secs. 4.1 and 4.2, compare message functions on identical inputs (Sec. 4.3), measure data efficiency (Sec. 4.4), check equivariance numerically (Sec. 4.5), and close with ablations (Sec. 4.6).

Datasets. For face segmentation, **MFCAD++** (Colligan et al., 2022) (59.6K parts, 25 classes) and **MFTRCAD** (Xia et al., 2024) (28.6K parts, 27 classes) label each face with a machining feature; MFTRCAD adds rotational features and multi-layer Boolean operations. **Fusion 360** (Lambourne et al., 2021) (35.7K parts, 8 classes) labels each face with the modeling operation that created it. For classification, **SolidLetters** (Jayaraman et al., 2021) contains 96K solids of 26 letters in 2,002 fonts, and **BenDFM** (Ballegeer and Benoit, 2026a) contains 14K sheet-metal parts labeled by whether they can be bent without the part colliding with the tooling.

Baselines. We compare against four methods: (i) **UV-Net** (Jayaraman et al., 2021), which encodes coordinates and normals in the world frame; (ii) **FoV-Net** (Ballegeer and Benoit, 2026b), an $\mathrm{SE}(3)$-invariant B-rep network built on per-face frames; (iii) **VN-PointNet** (Deng et al., 2021) and (iv) **CSEConv** (Kim et al., 2024), two equivariant point cloud networks trained on 1,024 points sampled by farthest point sampling on the faces. CSEConv supports classification only.

Evaluation protocol and metrics. All models are trained on the original orientations of each dataset without rotation augmentation and tested twice: on the original test set (Original) and with every test part rotated about the three coordinate axes by random angles between 0 and 2π ($\mathrm{SO}(3)$). Segmentation is scored by per-face accuracy and mean intersection-over-union (mIoU) over classes, classification by accuracy. Point cloud baselines are scored per face by a majority vote over the points sampled on that face. We report mean $\pm$ std over five seeds.

Implementation. Node features have $C_0{=}64$, $C_1{=}16$, and $C_2{=}8$ channels of degree $\ell{=}0, 1, 2$, with $L{=}3$ equivariant GNN layers and $A{=}8$ attention heads; the readout uses $d_{\mathrm{out}}{=}128$. All models are trained with cross-entropy loss, batch size 64, and early stopping on validation loss (patience 25)

Table 1: Face segmentation on original and randomly rotated test parts. All models are trained on the original orientations. Accuracy and mIoU in %, mean ± std over five random seeds. Best in **bold**, second best underlined.

		Acc.		mIoU	
Model	Params (M)	Original	SO(3)	Original	SO(3)
MFCAD++					
UV-Net	1.37	98.62 ±0.05	41.09 ±6.38	96.02 ±0.15	20.66 ±3.04
FoV-Net	0.59	99.05 ±0.06	99.05 ±0.06	97.27 ±0.20	97.27 ±0.20
VN-PointNet	1.08	83.57 ±1.61	83.57 ±1.61	31.38 ±3.51	31.38 ±3.51
SO3brep	**0.41**	**99.45** ±0.02	**99.45** ±0.02	**98.47** ±0.12	**98.47** ±0.12
MFTRCAD					
UV-Net	1.37	89.54 ±0.18	38.29 ±2.01	68.08 ±0.57	18.24 ±0.70
FoV-Net	0.59	90.28 ±0.26	90.28 ±0.26	71.84 ±0.40	71.84 ±0.40
VN-PointNet	1.08	77.12 ±1.09	77.12 ±1.09	17.52 ±1.35	17.52 ±1.35
SO3brep	**0.41**	**92.80** ±0.18	**92.80** ±0.18	**78.54** ±0.52	**78.54** ±0.52
Fusion 360					
UV-Net	1.36	87.59 ±0.16	68.60 ±1.53	65.92 ±0.99	37.16 ±0.98
FoV-Net	0.59	91.76 ±0.34	91.76 ±0.34	73.69 ±1.44	73.69 ±1.44
VN-PointNet	1.08	74.72 ±1.62	74.72 ±1.62	28.49 ±2.23	28.49 ±2.23
SO3brep	**0.40**	**95.11** ±0.12	**95.11** ±0.12	**82.05** ±0.50	**82.05** ±0.50

Table 2: Shape classification on original and randomly rotated test parts. All models are trained on the original orientations. Accuracy in %, mean ± std over five random seeds. Best in **bold**, second best underlined.

		SolidLetters		BenDFM	
Model	Params (M)	Original	SO(3)	Original	SO(3)
UV-Net	1.29	**97.36** ±0.14	9.22 ±0.70	75.50 ±0.45	56.53 ±1.12
FoV-Net	0.59	96.38 ±0.14	96.38 ±0.14	81.13 ±0.18	81.13 ±0.18
VN-PointNet	0.81	85.81 ±0.40	85.81 ±0.40	73.55 ±0.18	73.55 ±0.18
CSEConv	2.09	87.48 ±0.18	87.48 ±0.18	76.69 ±0.77	76.69 ±0.77
SO3brep	**0.36**	97.03 ±0.13	**97.03** ±0.13	**81.68** ±0.79	**81.68** ±0.79

on a single NVIDIA A5000 GPU. SO3brep uses AdamW (learning rate 10^{-3}, weight decay 0.05) with a 5-epoch linear warm-up and cosine decay to 2×10^{-5}, feature dropout p=0.15, and equivariant channel dropout p=0.10. UV-Net uses Adam, and FoV-Net and the point cloud baselines use AdamW (weight decay 0.01), all with learning rate 10^{-3}. Sec. A compares computational cost and reports graph sizes.

4.1 Face segmentation

Tab. 1 shows that SO3brep has the highest accuracy and mIoU on all three segmentation benchmarks, with and without test rotations: 99.45% accuracy on MFCAD++, 92.80% on MFTRCAD, and 95.11% on Fusion 360. Its original and rotated scores coincide because its output is invariant. UV-Net loses 19–58 accuracy points under rotation, since its input coordinates and normals are tied to the canonical pose. FoV-Net is invariant as well and is the strongest baseline; SO3brep exceeds it by 0.4, 2.5, and 3.4 accuracy points and by 1.2, 6.7, and 8.4 mIoU points on MFCAD++, MFTRCAD, and Fusion 360. VN-PointNet, which sees 1,024 sampled points and labels each face by a vote, reaches 17.5–31.4% mIoU against 71.8–98.5% for FoV-Net and SO3brep; Jayaraman et al. (2021) report the same ordering for non-equivariant networks.

4.2 Classification

Under rotation, SO3brep has the highest accuracy on both datasets (Tab. 2): 97.03% on SolidLetters and 81.68% on BenDFM. On BenDFM, its margin over FoV-Net ($+0.55$ points) is within the seed spread. On SolidLetters, UV-Net is 0.33 points more accurate on aligned parts but drops to 9.22% under rotation. Part of the SolidLetters labels depends on pose: the extruded letters b, d, p, and q are

Table 3: Controlled comparison of message functions on identical input features. FoV-Net uses its own architecture and features and is shown for reference. Accuracy and mIoU in %, mean ± std over five random seeds (for 100 training parts, over five random training subsets). Best of the four message functions in **bold**.

		Full training set				100 training parts	
		Original		$\mathrm{SO}(3)$		$\mathrm{SO}(3)$	
Message function	Params (M)	Acc.	mIoU	Acc.	mIoU	Acc.	mIoU
Fusion 360							
SO3brep (CG)	**0.40**	95.11 ±0.12	**82.05** ±0.50	**95.11** ±0.12	**82.05** ±0.50	**83.45** ±1.51	**48.53** ±4.26
Attention (non-eq.)	0.82	**95.31** ±0.10	81.84 ±0.51	87.95 ±0.27	61.89 ±0.82	79.00 ±1.52	41.92 ±2.43
Concat. (non-eq.)	0.68	95.08 ±0.25	81.09 ±1.02	87.81 ±1.11	62.14 ±1.86	76.44 ±1.36	35.86 ±2.65
Invariant (LRF)	0.68	94.78 ±0.14	80.56 ±1.06	94.78 ±0.14	80.56 ±1.06	81.51 ±1.18	44.83 ±2.42
FoV-Net (ref.)	0.59	91.76 ±0.34	73.69 ±1.44	91.76 ±0.34	73.69 ±1.44	71.82 ±1.32	33.74 ±1.45
MFTRCAD							
SO3brep (CG)	**0.41**	92.80 ±0.18	**78.54** ±0.52	**92.80** ±0.18	**78.54** ±0.52	**81.58** ±0.75	**43.01** ±1.65
Attention (non-eq.)	0.82	**92.87** ±0.10	78.10 ±0.45	75.10 ±2.54	58.67 ±2.36	69.55 ±2.32	32.56 ±2.02
Concat. (non-eq.)	0.68	92.76 ±0.24	77.81 ±0.73	75.28 ±1.61	56.76 ±1.27	65.75 ±3.91	28.92 ±3.45
Invariant (LRF)	0.68	92.69 ±0.06	77.42 ±0.40	92.69 ±0.06	77.42 ±0.40	68.11 ±0.90	29.87 ±1.19
FoV-Net (ref.)	0.59	90.28 ±0.26	71.84 ±0.40	90.28 ±0.26	71.84 ±0.40	57.05 ±2.38	18.36 ±2.48

close to rotations of one another, so an $\mathrm{SO}(3)$-invariant model can separate them only by font details, while a pose-sensitive model trained and tested on aligned parts can read their orientation. Fuchs et al. (2020) observe the same on the gravity-aligned ScanObjectNN, where giving their model the "up" direction raises its accuracy.

4.3 Message functions on identical inputs

SO3brep introduces both new input features and equivariant message passing. To separate the two, we keep SO3brep's inputs fixed and vary only the message function (Tab. 3). The *non-equivariant* variants replace the CG tensor product in Fig. 2 with attention or concatenation over the same world-frame features. The *invariant* variant skips the rotation by the Wigner D-matrices: $\ell{>}0$ node features stay in their LRFs, directional edge features are expressed in the sender's LRF, and, with no shared frame left for CG coupling, features are concatenated. It is the closest LRF-based counterpart of SO3brep and therefore the direct test of equivariant against invariant message passing.

Results. With the full training set and aligned parts, all four message functions are within 1.5 mIoU points of one another. The non-equivariant variants lose 7–18 accuracy and 19–21 mIoU points under rotation, since combining world-frame spherical-harmonic features without CG tensor products breaks equivariance; they also use 1.7–$2\times$ the parameters. Against the invariant message function, SO3brep is 0.1–0.3 accuracy points and 1.1–1.5 mIoU points higher at full data with 40% fewer parameters. With 100 training parts the difference grows to 1.9 accuracy and 3.7 mIoU points on Fusion 360 and to 13.5 and 13.1 on MFTRCAD. Under rotation, SO3brep's lead over the non-equivariant variants with 100 training parts is 4.5–7.0 accuracy and 6.6–12.7 mIoU points on Fusion 360 and 12.0–15.8 and 10.5–14.1 on MFTRCAD.

4.4 Data efficiency

Following Jones et al. (2023); Ballegeer and Benoit (2026b), we evaluate data efficiency on the aligned MFCAD++ segmentation dataset by training on smaller data subsets, using five random splits per subset size (Fig. 3). SO3brep is the most accurate model at every training-set size, and the gap is largest with few samples: with 10 training parts it is about 20 points more accurate than FoV-Net and 45 points more accurate than UV-Net, and with 50 parts it reaches 91.8%. Part of this gap comes from the directional input features SO3brep enables, as in Tab. 3. Industrial CAD training sets are often small, since labels require expert time and many part libraries are proprietary (Jones et al., 2023; Stjepandić et al., 2015), so the accuracy reached with few labeled parts decides whether a B-rep network can be used in practice.

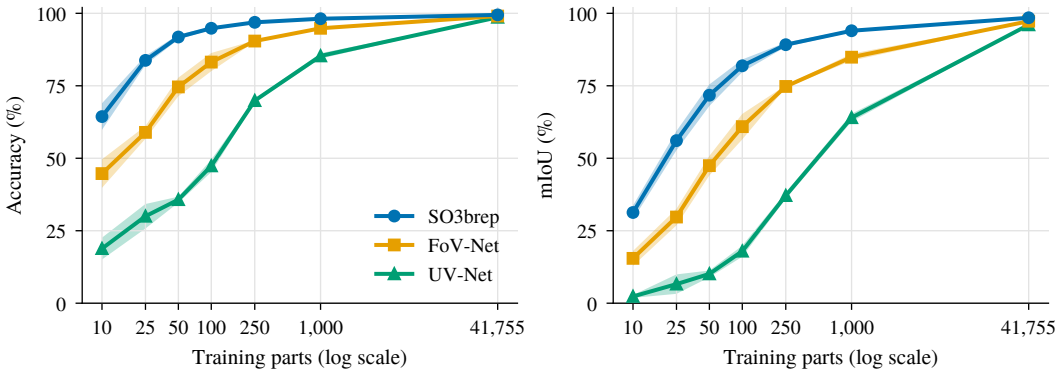

Figure 3: **Data efficiency.** Learning curves for MFCAD++ segmentation versus training size (log scale). SO3brep is above UV-Net and FoV-Net at every training-set size, with the largest gap at the smallest sizes. Shaded areas give the standard deviation over five random splits.

4.5 Equivariance validation

We measure the relative equivariance error (Fuchs et al., 2020) on the vector (ℓ=1) features of the last equivariant layer of each equivariant model,

$$\Delta_{\mathrm{EQ}} = \frac{\|\Phi(\mathbf{R}\cdot\mathcal{X}) - \rho(\mathbf{R})\,\Phi(\mathcal{X})\|_2}{\|\Phi(\mathbf{R}\cdot\mathcal{X})\|_2}, \qquad \rho(\mathbf{R}) = \mathbf{D}^{(1)}(\mathbf{R}), \tag{10}$$

where Φ is the feature extractor, $\mathcal{X}$ the input, and $\mathbf{R}$ a random rotation; we average over 1,000 SolidLetters inputs and 100 rotations per input. All three equivariant models are equivariant up to floating-point precision: $\Delta_{\mathrm{EQ}} = (3.46 \pm 4.91) \times 10^{-6}$ for SO3brep, $(9.83 \pm 9.80) \times 10^{-6}$ for VN-PointNet, and $(1.00 \pm 0.36) \times 10^{-7}$ for CSEConv.

4.6 Ablation studies

Fig. 4 ablates SO3brep on Fusion 360 segmentation. The architecture variants replace the node encodings with learned constants (*w/o node features*), replace the edge scalars and directional features with constants so that messages carry only topology (*w/o edge features*), or sum over neighbors instead of attending (*w/o attention*). The node-encoder variants remove one of the three face feature streams each, and the irrep degree ablations lower $\ell_{\max}$ to 1 and 0. Node features matter most: without them, accuracy drops by 17.4 points and mIoU by 29.3. Edge features follow (3.1 and 9.6 points), then attention (0.8 and 2.1). Among the node encoders, the UV-grid CNN contributes most (3.9 and 7.2) and the ray field least (0.5 and 1.5). Lowering $\ell_{\max}$ costs 0.3 and 1.0 points at $\ell_{\max}$=1 and 1.7 and 3.2 at $\ell_{\max}$=0, where only scalar features remain.

5 Conclusion

We introduced SO3brep, an $\mathrm{SO}(3)$-equivariant network for CAD B-reps. Node features are built in per-face frames and rotated into a shared frame with Wigner D-matrices, and messages couple them with the directions between faces through CG tensor products, so the network keeps the relative orientations of faces while its output is $\mathrm{SE}(3)$-invariant. On five benchmarks with randomly rotated test parts, it has the highest accuracy among the compared B-rep and equivariant point cloud methods, with the fewest parameters and the largest margin when training data is scarce. We hope SO3brep encourages further work on equivariant geometric deep learning for CAD.

Limitations. The edge features are limited to four unit vectors and nine scalars; richer descriptors between surfaces are possible. In-plane tangent directions are defined only up to rotation about $\hat{\mathbf{n}}_i$, making the u/v axes and edge orientations dependent on parameterization conventions and potentially inconsistent across cross-system B-rep exports. Operations that keep the solid but change its B-rep, such as splitting faces by axis-aligned planes or re-parameterizing surfaces as NURBS, also

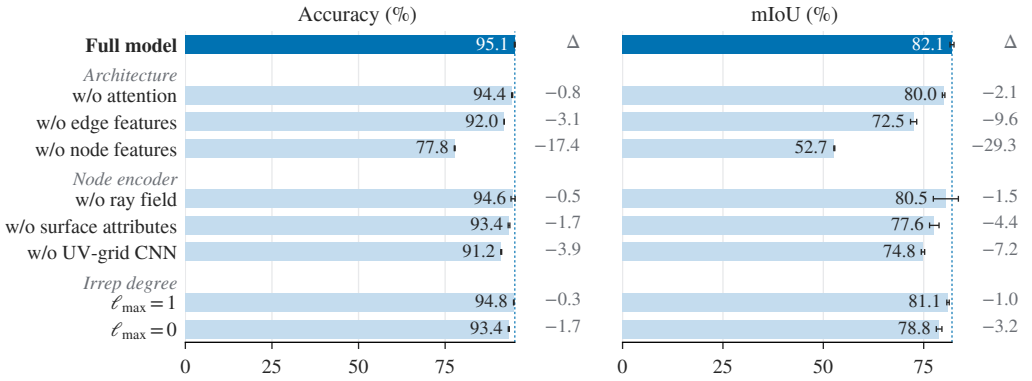

Figure 4: **Ablations on Fusion 360 face segmentation.** Each row removes or reduces one component of SO3brep; bars give accuracy and mIoU (%, mean ± std over five random seeds), and Δ the change from the full model in percentage points.

lower the accuracy of B-rep networks (Jiang et al., 2026). CG tensor products scale as $O(\ell_{\max}^6)$ in the maximum degree, which limits $\ell_{\max}$ in practice (Passaro and Zitnick, 2023; Liao et al., 2024). Predictions are per face, so continuous surface signals such as pressure fields are out of scope for the current architecture. Finally, with full training data the current benchmarks are close to saturated: all message functions in Tab. 3 are within 1.5 mIoU points on aligned parts, so these benchmarks show little of the expressivity advantage of equivariance over invariance.

Future work. Equivariant architectures enable tasks whose outputs must rotate with the input, such as predicting a machining approach direction, a drilling axis, or a part-to-part mating transform (Willis et al., 2022; Wu et al., 2023); invariant architectures cannot express these, and public CAD benchmarks for them are still missing. On the architecture side, we aim to reduce CAD-convention dependence via gauge-equivariant local reference frames (Wiersma et al., 2020; De Haan et al., 2021) and to improve inter-face geometric encoding with richer equivariant descriptors. Another direction is replacing approximate UV-grid encodings with exact NURBS-based representations (Fan et al., 2024) and better capturing trim geometry via boundary loops. Exploring alternatives to standard face-adjacency graphs, such as heterogeneous face–edge–vertex graphs (Li et al., 2026), could capture richer topological structure. Further directions are generative B-rep modeling and extension to assemblies.

Acknowledgments and Disclosure of Funding

This research was supported by Flanders Make, the strategic research centre for the Flemish manufacturing industry.

References

Ansaldi, S., De Floriani, L., and Falcidieno, B. (1985). Geometric modeling of solid objects by using a face adjacency graph representation. *ACM SIGGRAPH Computer Graphics*, 19(3):131–139.

Autodesk (2026). DefaultModelingOrientations enumerator. `https://help.autodesk.com/cloudhelp/ENU/Fusion-360-API/files/core_DefaultModelingOrientations.htm`. Fusion API Reference. Accessed November 3, 2026.

Ballegeer, M. and Benoit, D. F. (2026a). BenDFM: A taxonomy and synthetic CAD dataset for manufacturability assessment in sheet metal bending. *Journal of Intelligent Manufacturing*, pages 1–19.

Ballegeer, M. and Benoit, D. F. (2026b). FoV-Net: Rotation-invariant CAD B-rep learning via field-of-view ray casting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3024–3034.

Ballegeer, M., Van Camp, T., Jaspers, W., Bayar, A., Soe, A. N., Roelfs, M., Benoit, D. F., Decraemer, B., and Duflou, J. R. (2026). CAD-feature enhanced machine learning for manufacturing effort estimation on sheet metal bending parts. *arXiv preprint arXiv:2605.12266*.

Batatia, I., Kovacs, D. P., Simm, G., Ortner, C., and Csányi, G. (2022). MACE: Higher order equivariant message passing neural networks for fast and accurate force fields. *Advances in Neural Information Processing Systems*, 35:11423–11436.

Batzner, S., Musaelian, A., Sun, L., Geiger, M., Mailoa, J. P., Kornbluth, M., Molinari, N., Smidt, T. E., and Kozinsky, B. (2022). E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature Communications*, 13(1):2453.

Brandstetter, J., Hesselink, R., van der Pol, E., Bekkers, E. J., and Welling, M. (2022). Geometric and physical quantities improve E(3) equivariant message passing. In *International Conference on Learning Representations*.

Brehmer, J., Behrends, S., De Haan, P., and Cohen, T. (2025). Does equivariance matter at scale? *Transactions on Machine Learning Research*.

Cao, W., Robinson, T., Hua, Y., Boussuge, F., Colligan, A. R., and Pan, W. (2020). 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*, volume 11A: 46th Design Automation Conference (DAC), page V11AT11A003. American Society of Mechanical Engineers.

Chen, Y., Duan, L., Zhao, S., Ding, C., and Tao, D. (2024). Local-consistent transformation learning for rotation-invariant point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5418–5427.

Cohen, T. and Welling, M. (2016). Group equivariant convolutional networks. In *International Conference on Machine Learning*, pages 2990–2999.

Cohen, T. S., Geiger, M., Köhler, J., and Welling, M. (2018). Spherical CNNs. In *International Conference on Learning Representations*.

Cohen, T. S. and Welling, M. (2017). Steerable CNNs. In *International Conference on Learning Representations*.

Colligan, A. R., Robinson, T. T., Nolan, D. C., Hua, Y., and Cao, W. (2022). Hierarchical CADNet: Learning from B-reps for machining feature recognition. *Computer-Aided Design*, 147:103226.

Dai, Y., Huang, X., Bai, Y., Guo, H., Gan, H., Yang, L., and Shi, Y. (2025). BRepFormer: transformer-based B-rep geometric feature recognition. In *Proceedings of the 2025 International Conference on Multimedia Retrieval*, pages 155–163.

Dassault Systèmes (2020). Specifying the up axis for view orientation. `https://help.solidworks.com/2020/English/WhatsNew/t_view_up_axis.htm`. Accessed November 3, 2026.

De Haan, P., Weiler, M., Cohen, T., and Welling, M. (2021). Gauge equivariant mesh CNNs: Anisotropic convolutions on geometric graphs. In *International Conference on Learning Representations*.

Deng, C., Litany, O., Duan, Y., Poulenard, A., Tagliasacchi, A., and Guibas, L. J. (2021). Vector neurons: A general framework for SO(3)-equivariant networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12200–12209.

Deng, H., Birdal, T., and Ilic, S. (2018). PPF-FoldNet: Unsupervised learning of rotation invariant 3D local descriptors. In *European Conference on Computer Vision*, pages 620–638. Springer.

Fan, J., Gholami, B., Bäck, T., and Wang, H. (2024). NeuroNURBS: Learning efficient surface representations for 3D solids. *arXiv preprint arXiv:2411.10848*.

Fei, J. and Deng, Z. (2024). Rotation invariance and equivariance in 3D deep learning: a survey. *Artificial Intelligence Review*, 57(7):168.

Fuchs, F., Worrall, D., Fischer, V., and Welling, M. (2020). SE(3)-transformers: 3D roto-translation equivariant attention networks. *Advances in Neural Information Processing Systems*, 33:1970–1981.

Geiger, M. and Smidt, T. (2022). e3nn: Euclidean neural networks. *arXiv preprint arXiv:2207.09453*.

Heidari, N. and Iosifidis, A. (2025). Geometric deep learning for computer-aided design: A survey. *IEEE Access*, 13:119305–119334.

Jayaraman, P. K., Sanghi, A., Lambourne, J. G., Willis, K. D., Davies, T., Shayani, H., and Morris, N. (2021). UV-Net: Learning from boundary representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11703–11712.

Jiang, H., Mohamed, H. Y., Hitt, A., Lomakina, V., Jiang, H., and Jang, J. (2026). Learn the solid, not the file: Canonical inputs for neural networks on CAD boundary representations. *arXiv preprint arXiv:2609.11573*.

Jones, B. T., Hu, M., Kodnongbua, M., Kim, V. G., and Schulz, A. (2023). Self-supervised representation learning for CAD. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21327–21336.

Joshi, C. K., Bodnar, C., Mathis, S. V., Cohen, T., and Lio, P. (2023). On the expressive power of geometric graph neural networks. In *International Conference on Machine Learning*, pages 15330–15355.

Kim, J., Yoo, H. B., Han, D.-S., Song, Y.-J., and Zhang, B.-T. (2024). Continuous SO(3) equivariant convolution for 3D point cloud analysis. In *European Conference on Computer Vision*, pages 59–75.

Koch, S., Matveev, A., Jiang, Z., Williams, F., Artemov, A., Burnaev, E., Alexa, M., Zorin, D., and Panozzo, D. (2019). ABC: A big CAD model dataset for geometric deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9593–9603.

Lambourne, J. G., Willis, K. D., Jayaraman, P. K., Sanghi, A., Meltzer, P., and Shayani, H. (2021). BRepNet: A topological message passing system for solid models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12773–12782.

Lee, J., Yeo, C., Cheon, S.-U., Park, J. H., and Mun, D. (2023). BRepGAT: Graph neural network to segment machining feature faces in a B-rep model. *Journal of Computational Design and Engineering*, 10(6):2384–2400.

Li, F., Fujiwara, K., Okura, F., and Matsushita, Y. (2021a). A closer look at rotation-invariant deep point cloud analysis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16198–16207. IEEE.

Li, W., Zhang, J., Pang, J., and Zheng, D. (2026). BrepHGNet: A face-vertex interaction heterogeneous graph neural network for feature recognition. *Robotics and Computer-Integrated Manufacturing*, 100:103251.

Li, X., Li, R., Chen, G., Fu, C.-W., Cohen-Or, D., and Heng, P.-A. (2021b). A rotation-invariant framework for deep point cloud analysis. *IEEE Transactions on Visualization and Computer Graphics*, 28(12):4503–4514.

Liao, Y.-L. and Smidt, T. (2023). Equiformer: Equivariant graph attention transformer for 3D atomistic graphs. In *International Conference on Learning Representations*.

Liao, Y.-L., Wood, B., Das, A., and Smidt, T. (2024). EquiformerV2: Improved equivariant transformer for scaling to higher-degree representations. In *International Conference on Learning Representations*.

Lippmann, P., Gerhartz, G., Remme, R., and Hamprecht, F. A. (2025). Beyond canonicalization: How tensorial messages improve equivariant message passing. In *International Conference on Learning Representations*.

Lou, Y., Li, X., Chen, H., and Zhou, X. (2023). BRep-BERT: Pre-training boundary representation BERT with sub-graph node contrastive learning. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 1657–1666.

Passaro, S. and Zitnick, C. L. (2023). Reducing SO(3) convolutions to SO(2) for efficient equivariant GNNs. In *International Conference on Machine Learning*, pages 27420–27438.

Pozdnyakov, S. N. and Ceriotti, M. (2022). Incompleteness of graph neural networks for points clouds in three dimensions. *Machine Learning: Science and Technology*, 3(4):045020.

Puny, O., Atzmon, M., Smith, E. J., Misra, I., Grover, A., Ben-Hamu, H., and Lipman, Y. (2022). Frame averaging for invariant and equivariant network design. In *International Conference on Learning Representations*.

Satorras, V. G., Hoogeboom, E., and Welling, M. (2021). E(n) equivariant graph neural networks. In *International Conference on Machine Learning*, pages 9323–9332.

Schütt, K., Kindermans, P.-J., Sauceda Felix, H. E., Chmiela, S., Tkatchenko, A., and Müller, K.-R. (2017). SchNet: A continuous-filter convolutional neural network for modeling quantum interactions. *Advances in Neural Information Processing Systems*, 30.

Stjepandić, J., Wognum, N., and Verhagen, W. J. C., editors (2015). *Concurrent Engineering in the 21st Century: Foundations, Developments and Challenges*. Springer, Cham.

Su, Y., Gan, Y., Long, J., Yu, C., Zheng, W., Lv, J., and Liu, X. (2026). EquiCAD: A geometric equivariant neural network for 3D shape classification. In *International Conference on Machine Learning*, volume 306.

Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., and Riley, P. (2018). Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds. *arXiv preprint arXiv:1802.08219*.

Weiler, K. J. (1986). *Topological structures for geometric modeling (Boundary representation, manifold, radial edge structure)*. Rensselaer Polytechnic Institute.

Weiler, M., Forré, P., Verlinde, E., and Welling, M. (2021). Coordinate independent convolutional networks – isometry and gauge equivariant convolutions on Riemannian manifolds. *arXiv preprint arXiv:2106.06020*.

Weiler, M., Geiger, M., Welling, M., Boomsma, W., and Cohen, T. S. (2018). 3D steerable CNNs: Learning rotationally equivariant features in volumetric data. *Advances in Neural Information Processing Systems*, 31.

Wiersma, R., Eisemann, E., and Hildebrandt, K. (2020). CNNs on surfaces using rotation-equivariant features. *ACM Transactions on Graphics*, 39(4).

Willis, K. D., Jayaraman, P. K., Chu, H., Tian, Y., Li, Y., Grandi, D., Sanghi, A., Tran, L., Lambourne, J. G., Solar-Lezama, A., et al. (2022). Joinable: Learning bottom-up assembly of parametric CAD joints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15828–15839. IEEE.

Wu, H., Lei, R., Peng, Y., and Gao, L. (2024). AAGNet: A graph neural network towards multi-task machining feature recognition. *Robotics and Computer-Integrated Manufacturing*, 86:102661.

Wu, R., Tie, C., Du, Y., Zhao, Y., and Dong, H. (2023). Leveraging SE(3) equivariance for learning 3D geometric shape assembly. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14265–14274. IEEE.

Xia, M., Zhao, X., and Hu, X. (2024). Machining feature and topological relationship recognition based on a multi-task graph neural network. *Advanced Engineering Informatics*, 62:102721.

Xu, X., Jayaraman, P., Lambourne, J., Liu, Y., Malpure, D., and Meltzer, P. (2025). AutoBrep: Autoregressive B-Rep generation with unified topology and geometry. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pages 1–12.

Xu, X., Lambourne, J., Jayaraman, P., Wang, Z., Willis, K., and Furukawa, Y. (2024). BrepGen: A B-rep generative diffusion model with structured latent geometry. *ACM Transactions on Graphics*, 43(4):1–14.

Zhang, B. and Sennrich, R. (2019). Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32.

Zhang, S., Guan, Z., Jiang, H., Wang, X., and Tan, P. (2024). BrepMFR: Enhancing machining feature recognition in B-rep models through deep learning and domain adaptation. *Computer Aided Geometric Design*, 111:102318.

Zou, Q. and Zhu, L. (2025). Bringing attention to CAD: Boundary representation learning via transformer. *Computer-Aided Design*, page 103940.

A Computational cost and graph sizes

Tab. 4 compares preprocessing and runtime cost. SO3brep reuses the ray features of FoV-Net and adds per-edge attributes, so preprocessing takes 174–243 ms per part, against 116–195 ms for the other B-rep models and 255–283 ms for VN-PointNet. It has the fewest parameters; the CG tensor products make inference and training steps 2–3$\times$ slower than for FoV-Net and UV-Net, and about 6$\times$ faster at inference, with an order of magnitude less training memory, than for VN-PointNet.

Table 4: Computational cost. Preprocessing is averaged over 1,000 parts per dataset. Runtime is measured on Fusion 360 with batch size 64 on an NVIDIA A5000, mean $\pm$ std over 1,000 forward passes after warm-up.

	Preprocessing (ms/part)			Runtime (Fusion 360)		
Model	Fusion 360	MFTRCAD	Params (M)	Inference (ms)	Train step (ms)	Train mem. (MB)
UV-Net	115.7	195.3	1.36	10.2 $\pm$0.1	30.9 $\pm$1.4	854.6
FoV-Net	125.8	162.2	0.59	8.3 $\pm$0.4	32.0 $\pm$2.8	414.7
VN-PointNet	282.7	255.2	1.08	128.9 $\pm$1.4	419.9 $\pm$1.2	8701.9
SO3brep	173.8	243.1	0.40	22.0 $\pm$0.2	85.4 $\pm$4.4	669.2

Graph sizes and scaling. Tab. 5 lists the sizes of the face-adjacency graphs. Parts have 15–54 faces on average and at most 523. Each message-passing layer evaluates one CG tensor product, one radial network, and one attention weight per edge, each at a fixed cost for fixed channel widths, followed by node-wise updates. A forward pass through L layers therefore costs $O(L\,(|\mathcal{E}| + |\mathcal{V}|))$ and grows linearly with the number of edges $|\mathcal{E}|$, as in any message-passing GNN. The cost per edge depends on the channel widths and on $\ell_{\max}$ (Sec. 5), not on the size of the graph.

Table 5: Face-adjacency graph sizes. Edges are directed: each pair of adjacent faces contributes one edge in each direction.

		Faces		Edges	
Dataset	Parts	Mean	Max	Mean	Max
MFCAD++	59,649	30.0	86	306.5	1,096
MFTRCAD	28,637	25.5	149	236.3	1,680
Fusion 360	35,680	14.7	421	130.1	4,428
SolidLetters	96,430	33.8	523	360.7	6,224
BenDFM	14,000	53.9	126	463.1	1,232

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction claim that SO3brep is the first B-rep network with equivariant message passing between faces, that it has the highest accuracy under random test rotations on five benchmarks (Sections 4.1 and 4.2, with the exceptions stated there), that it is more data-efficient than UV-Net and FoV-Net (Section 4.4), and that equivariant message passing outperforms non-equivariant and invariant message functions on identical inputs (Section 4.3).

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The Limitations paragraph in Section 5 covers the restricted inter-face descriptors, the dependence of the in-plane axes on the kernel's parameterization, sensitivity to re-parameterization and face splitting, the cost of CG tensor products at higher degree, and the saturation of current benchmarks.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not include formal theorems or proofs. Equivariance of the architecture follows from standard results in the e3nn library and established properties of Wigner-D matrices and Clebsch–Gordan products, which are cited rather than re-proved.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 3 describes the architecture in sufficient detail to re-implement it, including the input and output representations, the sequence of operations, and the key design choices. Section 4 provides full architectural hyperparameters (channel widths, number of layers, attention heads, UV grid size, ray count. . .), optimizer settings (AdamW, learning rate schedule, weight decay, batch size, early stopping), and dataset references with sizes.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All datasets used (MFCAD++, MFTRCAD, Fusion 360, SolidLetters, BenDFM) are publicly available through their original publications. Code is available at `https://github.com/UGent-CVAMO/so3brep`, with instructions for reproducing the main results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The Implementation paragraph in Section 4 specifies all key hyperparameters, the optimizer (AdamW, learning rate 10^{-3} with a 5-epoch linear warm-up and cosine decay to 2×10^{-5}, weight decay 0.05), batch size (64), early stopping patience (25), dropout rates (0.15 on features, 0.10 on equivariant channels), the optimizers of the baselines, and the evaluation protocol (aligned vs. SO(3)-rotated test sets, five random seeds).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All results in Tables 1, 2, 3 and Figs. 3 and 4 are reported as mean $\pm$ std. over five runs, with the source of variability (random seed) clearly stated in the captions.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All models are trained on a single NVIDIA A5000 GPU (Section 4). Appendix A reports the parameter count, preprocessing time per part, inference and training-step time, and training memory of each model.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work involves geometric deep learning on CAD geometry and raises no ethical concerns.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The paper presents a method for learning on CAD geometry with applications in manufacturing and design automation. There are no foreseeable negative societal impacts beyond the general impacts of automation in manufacturing, which are not specific to this work.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The method is a discriminative model for CAD geometry classification and segmentation; it poses no meaningful risk for misuse and does not require special safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets and the e3nn python library are cited, and their terms of use are respected.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new datasets or pre-trained model checkpoints. The architecture is described in sufficient detail to re-implement it.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper involves no crowdsourcing or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper involves no human subjects research; IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not used as part of the core methodology. Any LLM assistance was limited to writing and editing and does not affect scientific content or originality.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.