A. Supplementary Material

A.1. Dataset statistics

The *Fusion 360 Gallery* segmentation dataset contains a total of 35,858 B-rep bodies with corresponding segmentation information, high quality triangle meshes and point clouds. The train/test split used in this work is published with the dataset. It contains $5,399$ bodies in the test set, with the remaining $30,459$ bodies for use training and performing validation. Example designs from the dataset are shown in Figure 10, colored according to segmentation label.

The complexity of the models can be understood from the number of faces per B-rep body and the number of CAD modeling operations used in their construction. Histograms of these distributions are shown in Figure 7. Many of the bodies are relatively simple with half of them having fewer than 9 faces. There is a long tail of more complicated B-reps with the most complex having 421 faces. Just over half of the bodies were constructed with more than one CAD modeling operation and 31% have two or more. Only 1% of the bodies used more than 10 operations in the construction history and the maximum number of operations used to create any B-rep in the dataset is 59.

As explained in Section 4, the dataset is modeled entirely using extrusions, revolutions, fillets and chamfers. The Autodesk Fusion 360 CAD package allows the geometry/topology created by each modeling operation to be immediately combined with the body being constructed using either a boolean union, subtraction or intersection. Extrusions are the most common way geometry is created, comprising 74% of modeling operations. To avoid a very large imbalance in the dataset, the collection of faces created by extrusions are subdivided as follows. When the extrusions were used to create new bodies or unioned with to existing bodies, the faces are placed in the *Extrude* class, while faces from extrusions which are subtracted from or intersected with existing bodies are placed in the *Cut* class. The faces generated by extrusions and revolutions are then further subdivided into side and end faces. For extrusions, side faces are created by sweeping the profile geometry while end faces are parallel to the plane on which the profile was sketched. For revolutions, sides faces are created by revolving the profile. *RevolveEnd* faces are only created when the extrusion is not 360 degrees as shown in Figure 9. This results in eight possible labels: *ExtrudeSide*, *ExtrudeEnd*, *CutSide*, *CutEnd*, *Fillet*, *Chamfer*, *RevolveSide*, and *RevolveEnd*.

The fraction of faces with each label type is shown in Figure 8. We see that just over half the faces are in the *ExtrudeSide* class, making the dataset relatively imbalanced. In particular the *RevolveEnd* class is very rare, accounting for just 0.08% of faces. The *CutEnd* and *Chamfer* classes

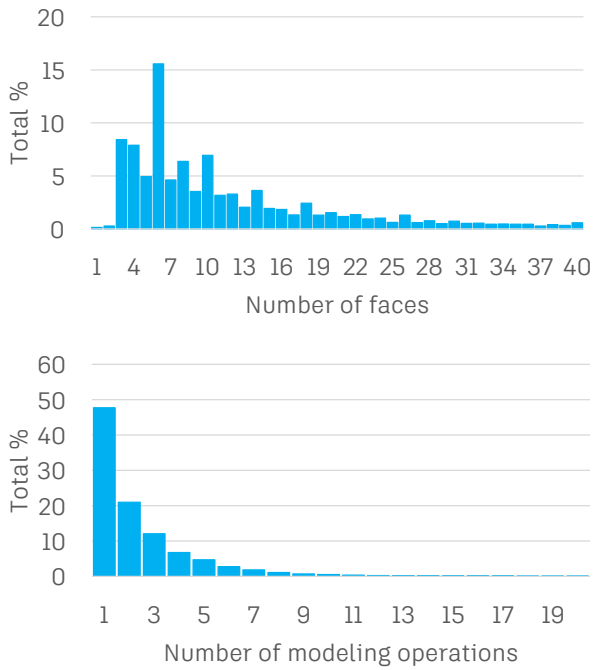

Figure 7: The distribution of faces per body (top) and of CAD modeling features used to generate each body (bottom).

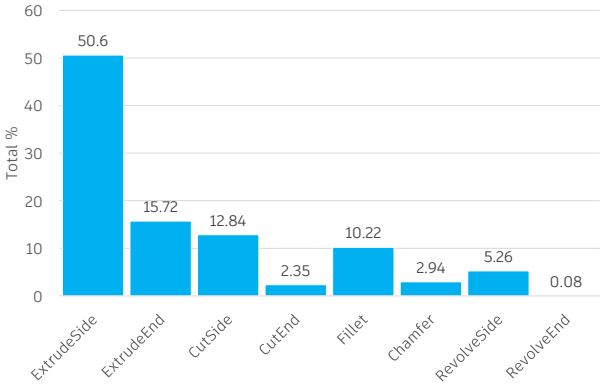

Figure 8: The percentage of faces in each class.

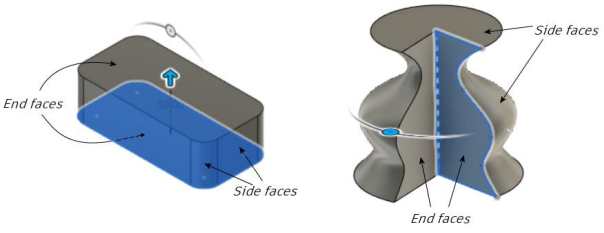

Figure 9: Side and end faces for extrusions and revolutions.

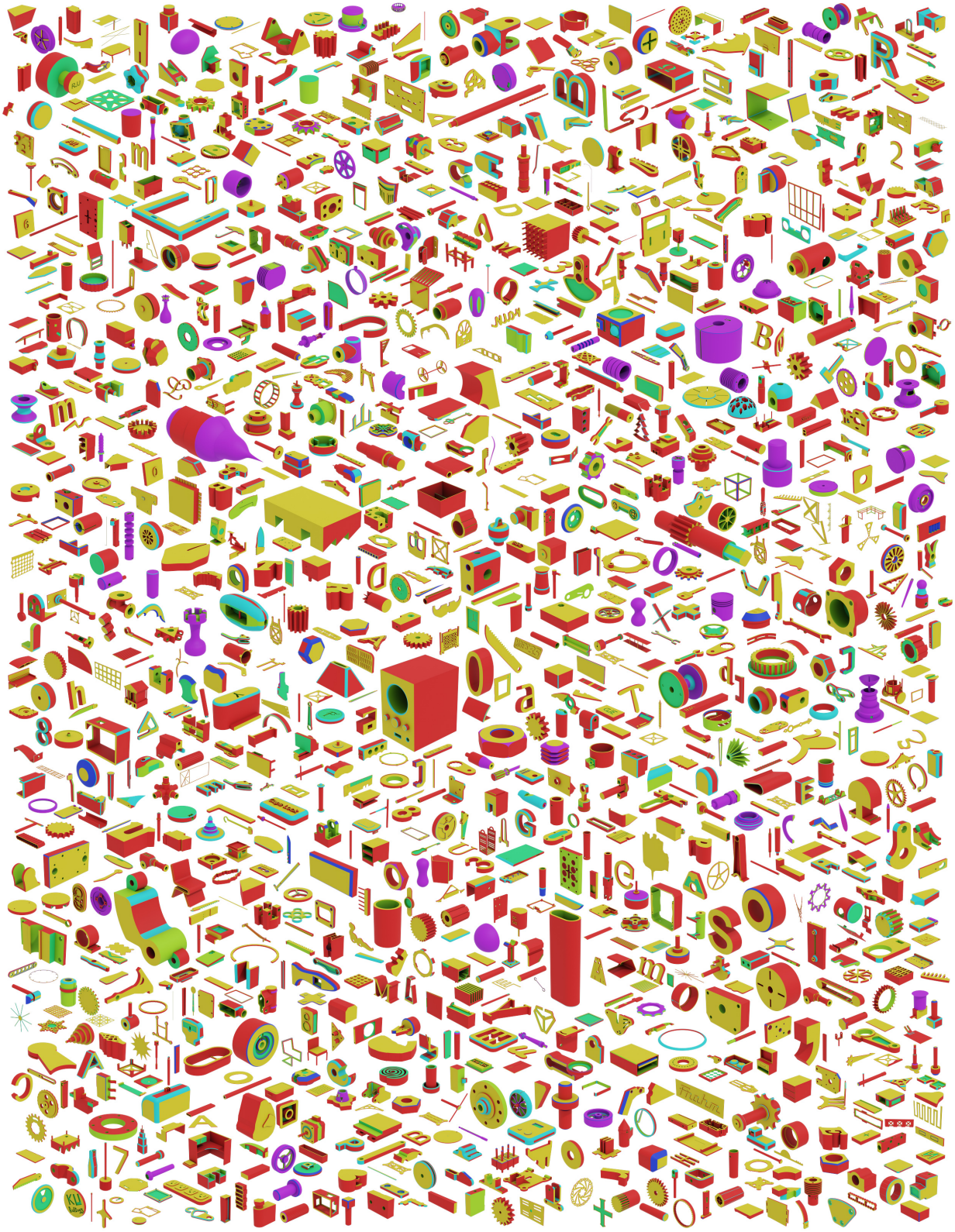

Figure 10: Example designs from the *Fusion 360 Gallery* segmentation dataset, colored by segmentation label.

Kernel	Faces	Edges	Coedges
Simple edge	**F**, **MF**	**E**	**I**, **M**
Asymmetric	**F**, **MF**	**E**	**I**, **N**
Asymmetric+	**F**, **MF**	**E**	**I**, **M**, **N**
Asymmetric++	**F**, **MF**	**E**, **NE**	**I**, **M**, **N**
Winged edge	**F**, **MF**	**E**, **NE**, **PE**, **MNE**, **MPE**	**I**, **M**, **N**, **P**, **MN**,**MP**
Winged edge+	**F**, **MF**	**E**, **NE**, **PE**, **MNE**, **MPE**	**I**, **M**, **N**, **NM**, **P**, **PM**, **MN**, **MNM**, **MP**, **MPM**
Winged edge++	**F**, **MF**	**E**, **NE**, **PE**, **MNE**, **MPE**, **NMNE**, **PMPE**, **MPMPE**, **MNMNE**	**I**, **M**, **N**, **NM**, **P**, **PM**, **MN**, **MNM**, **MP**, **MPM**, **NMN**, **PMP**, **MPMP**, **MNMN**

Table 2: The topological walks making up the kernels shown in Figure 5.

are also rare at 2.35% and 2.94% of the faces respectively. The faces in these classes are often planar, so the correct segment type can only be identified by considering the surrounding shape. This makes the identification of these rare classes an extremely challenging task.

A.1.1 Data preparation

In this section we describe the processing performed on the designs from the Autodesk Online Gallery[1] to generate the *Fusion 360 Gallery* segmentation dataset. Scripts were used to drive the Application Programming Interface (API) of the *Fusion 360* CAD package to load each Fusion document and suppress all features except extrusions, revolutions, fillets and chamfers. While this procedure modified the shapes of some models, it greatly increased the number of B-rep bodies available for the dataset.

The number of faces and edges in the resulting bodies, along with the surface area and volume of each body, was then recorded. To remove duplicate models, we first search for bodies with the same number of faces and edges. The area and volume of these candidates are then checked and bodies for which the area and volume matches to within 1% were discarded. The de-duplication procedure was verified using thumbnails of the resulting duplicate free dataset. These were ordered by surface area and images were manually inspected to verify that the area and volume tolerance was appropriate for detecting duplicate bodies, even when rigid body transforms had been applied. Models with small topological differences were not considered to be duplicates as BRepNet is designed to be sensitive to differences in the model topology.

All the B-reps in the dataset were transformed to place the center of their bounding boxes at the origin. A uniform scaling was then applied so that the largest length of the bounding box is 2 model units across. Meshes and point clouds were then extracted from these scaled models.

A.1.2 Input feature standardization

The input features for BRepNet are standardized using the following procedure. The mean, μ and standard deviation, σ, are computed for each input feature for the entire training set. The standardized values x' are then computed as

$$x' = \frac{x - \mu}{\sigma} \tag{1}$$

A.1.3 Support for operation grouping and ordering problems

In addition to the segmentation data, we also provide more detailed information about the modeling operations used to construct the solid. For each B-rep face we provide a unique identifier for the operation which created it, allowing groups of faces created by different operations of the same type to be separated. For extrusions and revolutions we also provide a classification of the face as *Start*, *End* or *Side*. We provide the type of each operation and the order in which the operations were applied in the parametric model history. For the point cloud and mesh representations we provide the mapping from each point and triangle to the face from which it was sampled, allowing this extra information to be used for all representations. More details on how this data is organized is in the dataset documentation[2].

This additional data is intended to support research into reverse engineering tasks. Grouping the points or triangles according to the operation which created them is a first step towards rebuilding the parametric history. Further subdividing each extrusion based on the *Start*, *End* and *Side* information allows the extrusion direction to be predicted. Slicing the mesh perpendicular to this extrusion direction can then allow the profile curves to be extracted and the

[1] https://gallery.autodesk.com/fusion360

[2] https://github.com/AutodeskAILab/Fusion360GalleryDataset

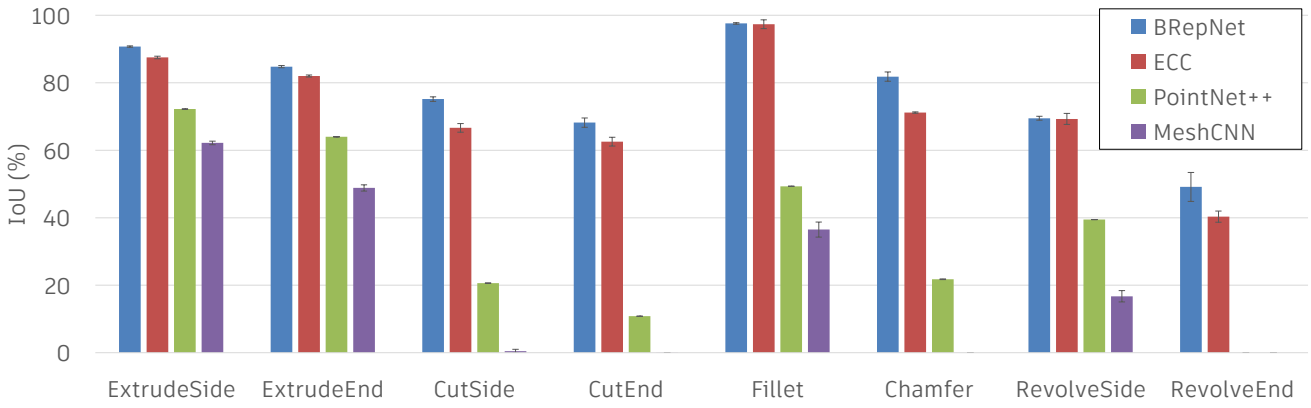

Figure 11: The per-face IoU values achieved by each network for each segment class.

extruded volumes regenerated. Finally, by predicting the ground truth operation type and order, sensible sequences for the modeling operations can be learned.

A.2. Kernels

In Section 5.2 the performance of BRepNet was compared with a number of different kernels. Diagrams showing the entities taking part in these kernels are shown in Figure 5. In these diagrams each topological walk in the kernel terminates on a distinct entity. It should be noted that this will not always be the case for arbitrary B-reps data. Some local topology will result in two or more topological walks terminating on the same entity. No special case handling is required when this happens. The procedure described in Equation 2 simply concatenates feature vectors from the same entity into the same row of Ψ multiple times. The network learns to recognizer these repeated patterns in the feature vectors in the same way as in the case where the entities are distinct. Table 2 gives the full lists of topological walks which make up the kernels.

A.3. Experiments

A.3.1 Training details

For all the experiments described in Section 5, the BRepNet network was trained using the Adam optimizer, with default parameters (learning rate of 0.001 and betas 0.9 and 0.999). The B-reps in the dataset were divided into mini-batches, each containing approximately 1000 faces. Multiple B-Reps can be combined into a same batch by row-wise concatenation of the input feature matrices $\mathbf{X}^f$, $\mathbf{X}^e$ and $\mathbf{X}^c$, and diagonal concatenation of the matrices $\mathbf{N}$, $\mathbf{P}$, $\mathbf{M}$, $\mathbf{E}$ and $\mathbf{F}$. Training and evaluate was performed using NVIDIA Tesla V100 GPUs. Training took an average of 45s per-epoch with the network typically achieving a minimum loss value by around the 15th epoch. Hence the BRepNet network could be trained on the *Fusion 360 Gallery* segmentation dataset in under 12 minutes from random seed.

For experiments comparing against PointNet++ and MeshCNN we use the official implementation and retain the default hyper-parameters where possible. PointNet++ models are trained with a batch size of 32, a learning rate of 0.001 and momentum of 0.9 using the TensorFlow implementation[3] from [2]. The PyTorch implementation[4] of MeshCNN was used with a batch size of 12, a learning rate of 0.0002 and momentum of 0.9. The maximum number of input edges for any mesh was 3500 and the pooling resolutions were set to 2500, 1750 and 1000.

The ECC used the pytorch-geometric NNConv implementation[5]. The Adam optimizer was used for the training with default parameters (learning rate of 0.001 and betas 0.9 and 0.999).

A.3.2 Comparison of IoU for different classes

In this section we show the IoU values achieved by BRepNet, the Edge-Conditioned Convolution (ECC) graph network [3], PointNet++ [2] and MeshCNN [1] for the different classes individually. Figure 11 shows the per-face IoU values each network achieved on each class and Figure 12 shows images of the face segmentation on some example models. We see that a key reason why BRepNet and the ECC network perform better than PointNet++ and MeshCNN is their ability to correctly classify faces in the rare classes.

The *RevolveEnd* class always consists of planar faces and accounts for just 0.08% of the dataset. While BRepNet finds this class challenging, achieving only 49% IoU, both PointNet++ and MeshCNN fail to identify any *RevolveEnd* faces. We believe this is because the *RevolveEnd* class can only be identified by considering a face in the context of

[3] https://github.com/charlesq34/pointnet2
[4] https://github.com/ranahanocka/MeshCNN
[5] https://pytorch-geometric.readthedocs.io/en/latest/modules/nn.html

its neighbouring faces and edges. This is something which the ECC and BRepNet approaches can do easily, as both networks are designed to leverage information from adjacent faces. PointNet++ and MeshCNN have a hierarchical design which is intended to improve the flow of information from neighbouring geometry. PointNet++ achieves an IoU of 39% of the neighbouring *RevolveSide* faces, while MeshCNN achieves only 17%. Neither architecture manages the extremely challenging task of using their identification of the *RevolveSide* faces to correctly classify the adjacent groups of planar points/edges as *RevolveEnd*.

The *CutEnd* class is also rare, accounting for just 2.35% of faces in the dataset. As for *RevolveEnd*, these faces are always planar. The primary way they can be distinguished from the more common *ExtrudeEnd* faces is by observing that they are often surrounded by concave edges. PointNet++ is able to correctly identify 10.83% of *CutEnd* faces, while MeshCNN recognizes just 0.01% of them. While MeshCNN uses dihedral angle as an input feature, it does not distinguish between concave and convex edges. This would explain why MeshCNN struggles to distinguish the subtractive extrusion classes (*CutSide* and *CutEnd*) from the more common additive extrusions (*ExtrudeSide* and *ExtrudeEnd*).

The *Fillet* and *Chamfer* classes account for 10.22% and 2.94% of faces in the dataset. Fillets are very distinctive features with smooth edges and typically cylindrical or toroidal geometry. Both BRepNet and the ECC have high IoU scores for this class (97.57% and 97.32% respectively), demonstrating that these patterns in the input features are easy to spot using both architectures. Faces in the *Chamfer* class are much less distinctive. They are often planar or conical and their edges can be either concave or convex. Consequently BRepNet and the ECC achieve lower IoU scores of 81.80% and 71.16%. PointNet++ and MeshCNN also have higher IoU values for fillets than for chamfers. For fillets these networks achieve IoUs of 49.31% and 36.49% respectively while for chamfers PointNet++ achieves only an IoU of 21.79% and MeshCNN fails to detect any chamfer features.

Both *Fillet* and *Chamfer* faces tend to have relatively small areas and consequently are prone to under-sampling when fixed edge-count meshes and fixed size point clouds are generated, however the IoU values PointNet++ and MeshCNN achieve for *Fillet* suggests this was not a major limiting factor for the relatively small solids in the *Fusion 360 Gallery* segmentation dataset. Comparing the results for *Fillet* with the less distinctive *CutSide* class we see that all network are more successful at detecting fillets, despite these faces having smaller areas and accounting for a similar fraction of the dataset.

The *ExtrudeSide* and *ExtrudeEnd* classes account for 50.60% and 15.72% of the dataset respectively. All networks do well on these classes. It should be noted that for both BRepNet and the ECC the IoU achieved for the most common *ExtrudeSide* class is smaller than for the more distinctive *Fillet* class.

References

[1] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: a network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019. 4

[2] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017. 4

[3] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017. 4

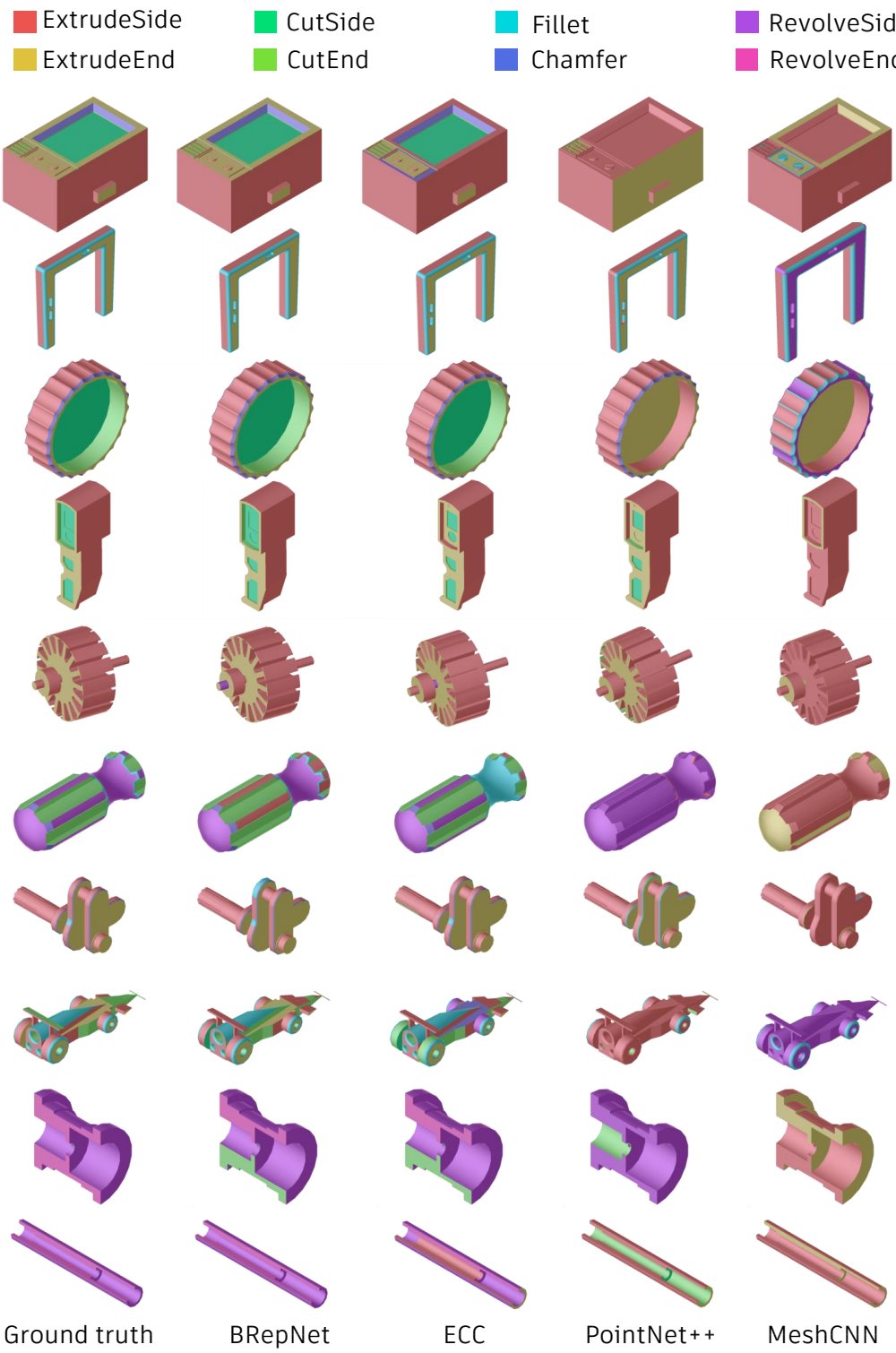

Figure 12: The per-face segmentation generated by each network.