Supplementary Material for: Neural Face Identification in a 2D Wireframe Projection of a Manifold Object

Kehan Wang*
University of California, Berkeley
wang.kehan@berkeley.edu

Jia Zheng Zihan Zhou
Manycore Tech Inc.
{jiajia, shuer}@qunhemail.com

1. ABC Dataset

Since ABC dataset [2] is created from Onshape public repository, it contains many duplicate shapes, such as simple boxes and cylinders that were created when novice users tested the Onshape software. There also exist many over-complicated shapes that do not help our model generalize (Figure 1(a)). To make our model learn more effectively, we use the following set of heuristics to filter the ABC dataset when constructing our own dataset.

Topology. Same shapes must share the same topology. To find duplicate shapes, we first group shapes of matching topology features together, and then find duplicates within each group. When grouping, we consider the following topology features: the number of parts, surfaces, edges of a shape, and its face and edge type distribution.

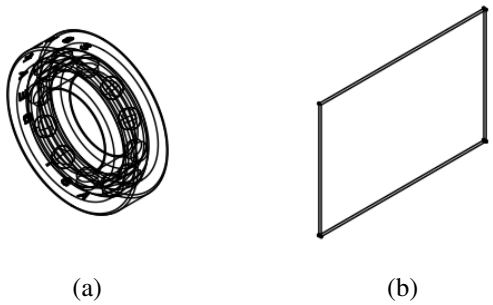
Visual similarity. We then further cluster within each group based on the visual similarity of the shapes. We use three orthogonal views of a shape as its visual feature, and run agglomerative clustering[1] based on the Jaccard distance of three views between two shapes.

Thickness. Because the ABC dataset contains a lot of primitives which only consist of a plane, we filter out shapes that are too thin (Figure 1(b)). The thickness of a shape is defined as the minimum distance of all pair-wise distances between two edges within a shape. We remove shapes that have thickness smaller than 0.05.

Complexity. We also filter out shapes with more than 42 faces or 37 edges in a face to eliminate over-complicated shapes that result in multiple small, overlapping faces in the same area of the line drawing.

2. AtlasNet

In order to train AtlasNet, we convert the vectorized line drawings to bitmap images, as shown in Figure 2. Following SPARE3D [1], the hidden lines are drawn in red. We adopt the original implementation[2] of AtlasNet and use its original optimization settings.



(a) (b)

Figure 1. Examples of over-complicated shapes and thin shapes in ABC Dataset.

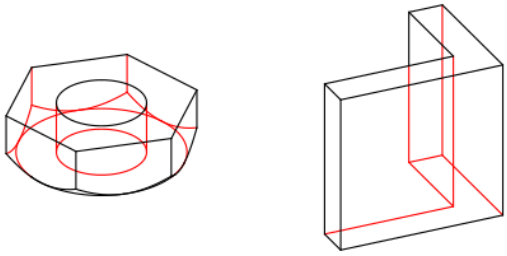


Figure 2. Examples of image inputs to AtlasNet.

3. Network Architecture

Figures 3 and 4 compares the network architectures of a naive *seq2seq* model and our proposed model *Faceformer*. In the *seq2seq* model, the encoder takes all co-edges and special tokens as input. Then, the decoder takes a start token `[SOS]` as the first input, and outputs all face predictions sequentially. In our model, every co-edge is used as the first input to the decoder, and all faces are generated simultaneously.

*Work done during an internship at Manycore Tech Inc.
[1]We use `cluster.AgglomerativeClustering` from SciPy.
[2]https://github.com/ThibaultGROUEIX/AtlasNet

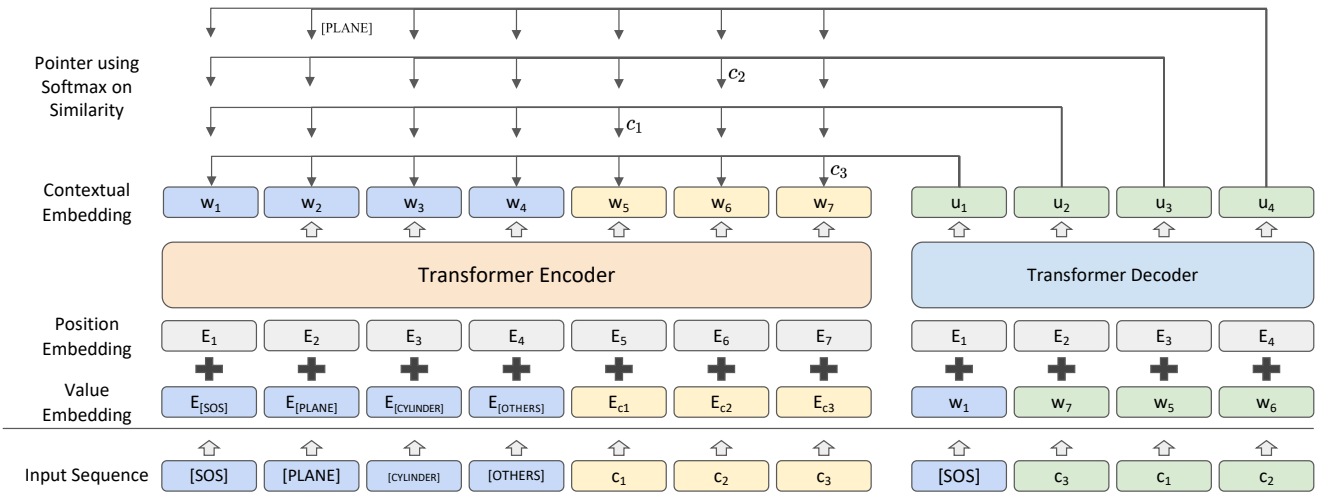


Figure 3. The network architecture of a naive *seq2Seq* model.

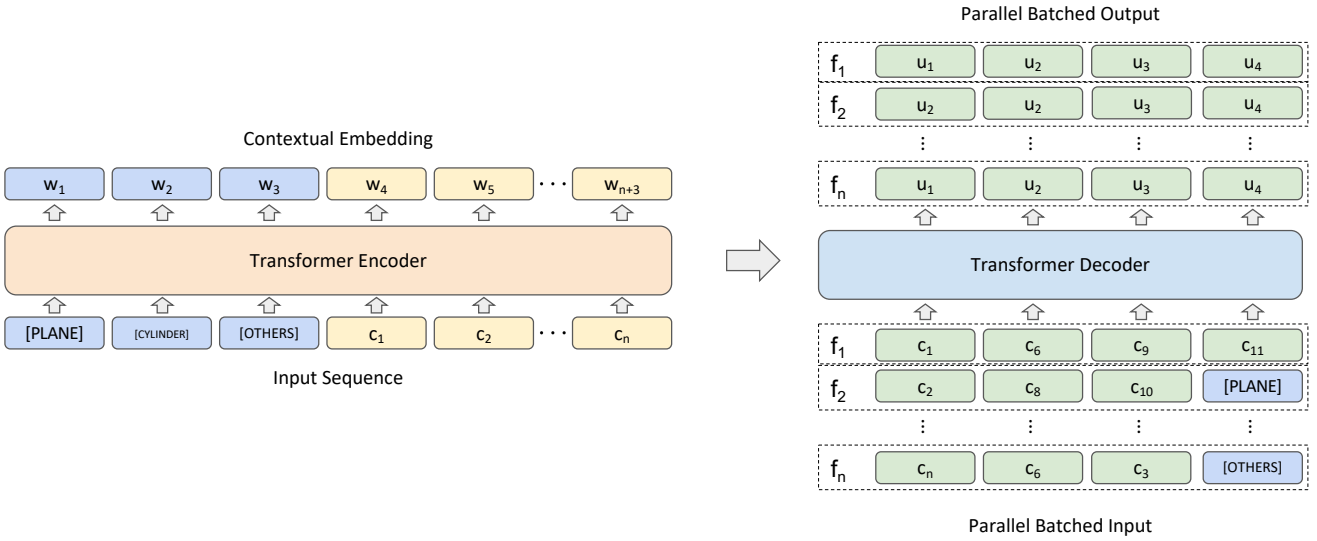


Figure 4. The network architecture of our proposed model *Faceformer*.

4. Experiment on Shape Complexity

We have also investigated the effect of 3D object complexity on the performance of our deep face identification model. Figure 5 shows the precision and recall versus the number of faces and edges in the test set. As one can see, our model's performance remains largely stable with increased complexity. Note that results for objects with more than 25 faces or 125 edges are less informative as such objects are rare in the test set.

5. 3D Mesh Reconstruction and Alignment

The 3D object reconstruction algorithm described in the main text (Section 5) only recovers a 3D wireframe model of the object (by computing the depth of each vertex). To further recover the mesh model, we use pythonOCC[3], a Python 3D development framework built upon the Open CASCADE Technology, to connect the 3D vertices into edges, edges into wires, and wires into faces. Specifically, we use pythonOCC's BRepBuilderAPI to make edges, wires and faces. We congregate all reconstructed faces of an object into one compound and generate its mesh with pythonOCC's ShapeTesselator. Unenclosed face predictions are omitted through this process.

Due to orthographic projection, the mesh models reconstructed from the 2D line drawings do not necessarily align with the ground truth - there can be a constant amount of

[3] https://github.com/tpaviot/pythonocc

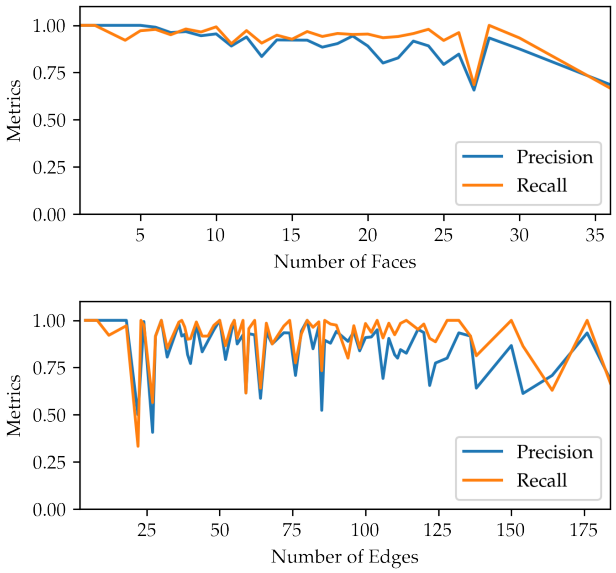


Figure 5. Precision and recall *versus* number of faces and number of edges in a given object.

shift applied depth-wise to our mesh. To counter this, we apply a depth-wise shift to align our meshes by minimizing depth-wise distances between our vertices and ground truth's vertices.

References

[1] Wenyu Han, Siyuan Xiang, Chenhui Liu, Ruoyu Wang, and Chen Feng. Spare3d: A dataset for spatial reasoning on three-view line drawings. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 14690–14699, 2020. 1

[2] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 9601–9611, 2019. 1