APS360 Final Project Report:

An Investigation of Rotationally Invariant Shape Completion in 3D Space

Team 45 Christopher Agia, Polina Govorkova,
YiFei Tang, Chao Glen Xu

Project Link github.com/agiachris/rotational3DCNN (GitHub)

1. Introduction

Recent years have brought forth various technologies that depend heavily on sensing capabilities via software. Examples include, but are not limited to: autonomous driving, robotic manufacturing, and augmented reality. A commonality amongst these tasks is that they require some form of interaction with the 3D world (e.g. virtual or physical), but must operate on sensor information that is inherently an incomplete representation (e.g. 2D images, 2D depth maps, sparse 3D point clouds). In this project, we address a component of this challenge by learning to infer the complete 3D structure of objects from their partial 3D representations with convolutional neural networks (CNNs). Classical approaches to shape reconstruction that exploit object symmetries [1] or plane fitting [2] struggle to generalize to shape irregularities. Alternative methods that fit 3D mesh models to object instances [3] are dependent on the size and variation of their prior libraries - larger libraries result in higher quality reconstructions, but suffer from slow retrieval times. This motivates a deep learning powered solution that is capable of reasoning about full shape geometries under practical computational and operational constraints.

In this work, we represent the partial shapes of objects in terms of a perspective signed distance field [4] (SDF, i.e. input) and infer a perspective-invariant distance field of the completed shape (DF, i.e. target), as illustrated in Fig. 1. Through this formulation, we hypothesize that the CNNs must learn *rotationally* or *perspective* invariant representations of objects belonging to the set of eight class categories contained in our dataset.

2. Illustration / Figure

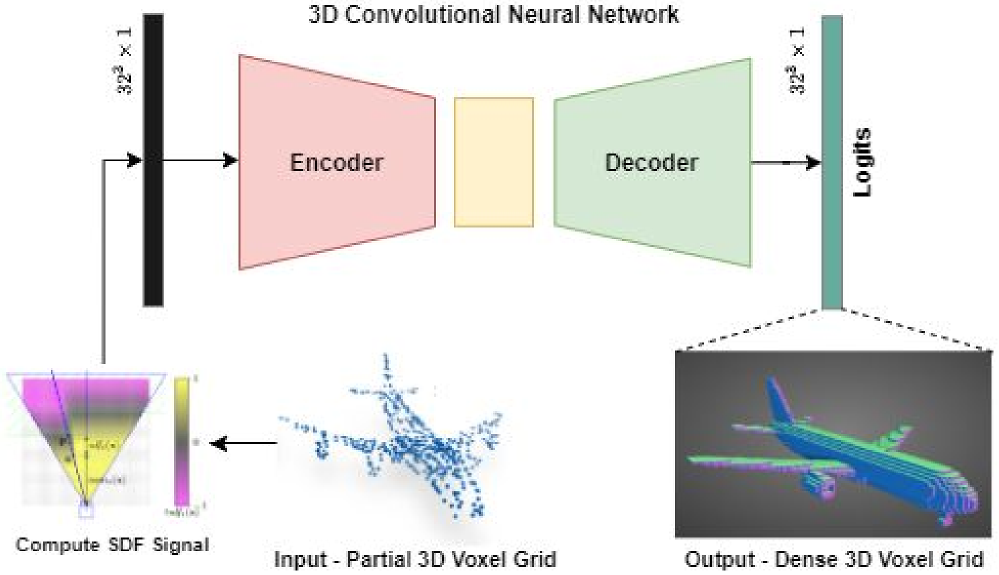

3D Convolutional Neural Network
$32^3 \times 1$
Encoder
Decoder
$32^3 \times 1$
Logits
Compute SDF Signal
Input - Partial 3D Voxel Grid
Output - Dense 3D Voxel Grid

Figure 1: System Overview - 3D Convolutional Neural Network with an encoder / decoder architecture to reconstruct an object from it's partial SDF voxel representation[1] [4] [5].

3. Background and Related Work

Shape completion and volume reconstruction are well explored problems in the research community. Proposed methods can be categorized into model-based algorithms and learning-based

[1] The system overview was taken from the project proposal as our objective remains unchanged.

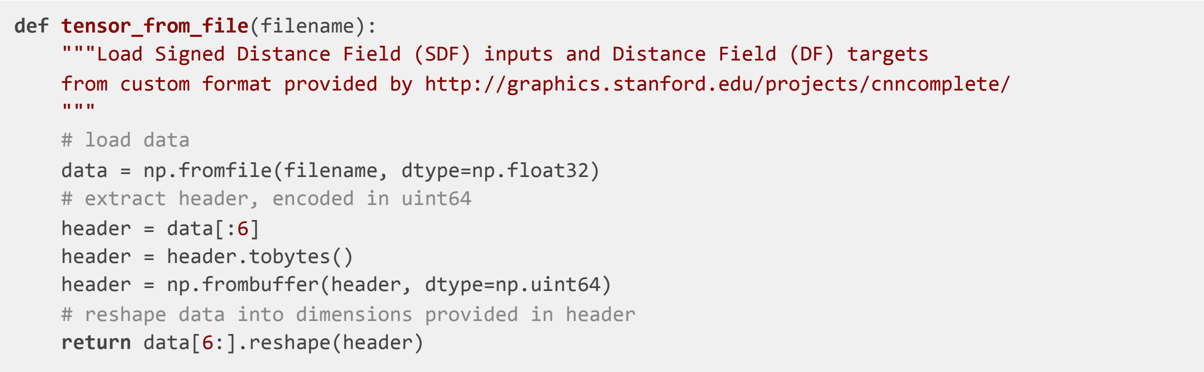
approaches. Contemporary work has brought neural networks to the forefront of such tasks, showing impressive results both in terms of inference times and memory requirements with a capacity to handle variations in objects and scenes [6], [7]. Dai *et al.* [6] proposed the use of a 3D Encoder-Predictor Network, which takes a partial SDF volume of an object and encodes its geometric and classification features into a shared embedding space. The combined latent representation of the object enables the network to learn class-specific dense 3D shapes which are generated in the decoder (e.g. similar to U-Net [8]).

Song *et al.* [7] extends this work to predicting the geometry and semantics of all objects in small indoor scenes. They propose SSCNet, which employs 3D dilated convolutions to increase the receptive field of the network with minimal degradation to the output quality (i.e. pooling reduces output voxel resolution). However, the network still predicts at one-fourth the input resolution which neglects fine-grained details of the scene. Shice *et al*. [9] improves upon this by adding atrous spatial pyramid pooling modules to fully mitigate the need for pooling and enables predictions at full resolution. Garbade *et al.* [10], leverage semantic segmentation from color images as a prior for the scene geometry, further improving the reconstructions.

4. Data Processing

The source of our data is the ShapeNetCore subset of the ShapeNet database [4], which has been curated for our particular task by Dai *et al*. [6, Sec. 4]. In particular, they go through an extensive process to produce signed distance field (SDF) and distance field (DF) input-target pairs, which altogether form the Stanford Graphics Shape Completion Benchmark [11].

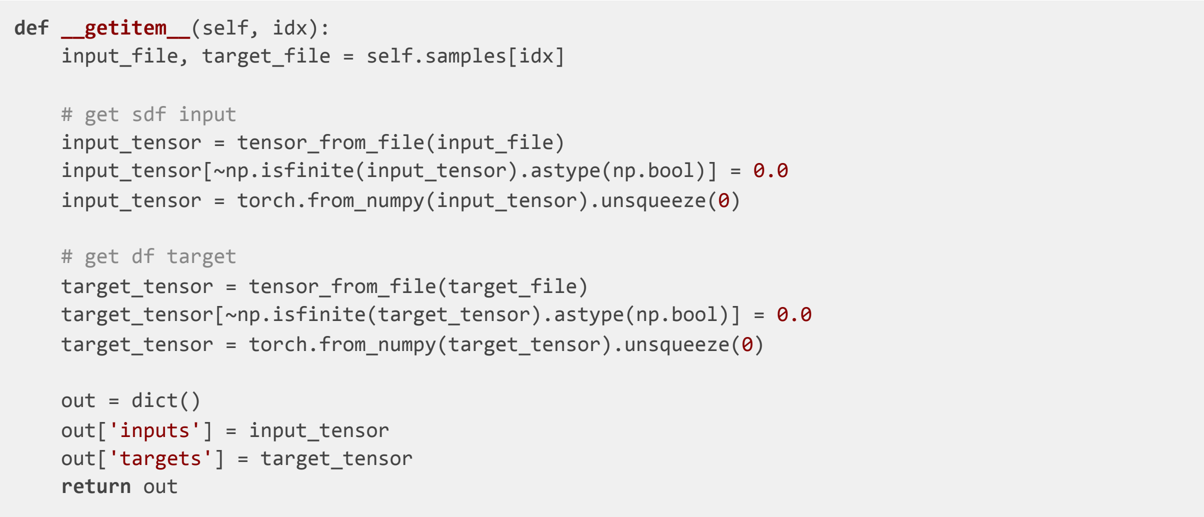
Data Pipeline. The data pipeline is responsible for loading, cleaning and processing the SDF and DF pairs. These files are stored in custom binary format: 192-bit unsigned header containing the 64-bit x 3 for $dimX$, $dimY$, $dimZ$ tensor dimensions, and a $(dimX * dimY * dimZ)$-bit float trailer containing the SDF or DF tensor data. These files are extracted in NumPy tensors as shown in Fig. 2.



```
def tensor_from_file(filename):
    """Load Signed Distance Field (SDF) inputs and Distance Field (DF) targets
    from custom format provided by http://graphics.stanford.edu/projects/cnncomplete/
    """
    # load data
    data = np.fromfile(filename, dtype=np.float32)
    # extract header, encoded in uint64
    header = data[:6]
    header = header.tobytes()
    header = np.frombuffer(header, dtype=np.uint64)
    # reshape data into dimensions provided in header
    return data[6:].reshape(header)
```

Figure 2: Extraction of SDF and DF custom binary files into 32^3 NumPy tensors.

For each complete shape model (DF), the dataset provides seven SDF volume inputs computed by rendering depth maps over a short trajectory. This yields seven input-target pairs per shape model each taken from a different camera perspective. A value of $-\inf$ is assigned to voxels in the SDF volume that do not project (with known intrinsics) onto a depth map pixel. Processing these infinite voxels will corrupt the training of our network, and hence, we filter them as shown in Fig. 3.



```
def __getitem__(self, idx):
    input_file, target_file = self.samples[idx]

    # get sdf input
    input_tensor = tensor_from_file(input_file)
    input_tensor[~np.isfinite(input_tensor).astype(np.bool)] = 0.0
    input_tensor = torch.from_numpy(input_tensor).unsqueeze(0)

    # get df target
    target_tensor = tensor_from_file(target_file)
    target_tensor[~np.isfinite(target_tensor).astype(np.bool)] = 0.0
    target_tensor = torch.from_numpy(target_tensor).unsqueeze(0)

    out = dict()
    out['inputs'] = input_tensor
    out['targets'] = target_tensor
    return out
```

Figure 3: Filtering $-\inf$ voxels in 32^3 SDF NumPy tensors.

Each voxel in the DF target contains a value based on the relative distance to the nearest occupied voxel. While learning to infer DF values promotes stable training (e.g. regression of DF values instead of binary occupancy classification), they are not qualitatively presentable. Thus, we threshold the DF to acquire the volumetric binary occupancy of the shape (see Fig. 4).

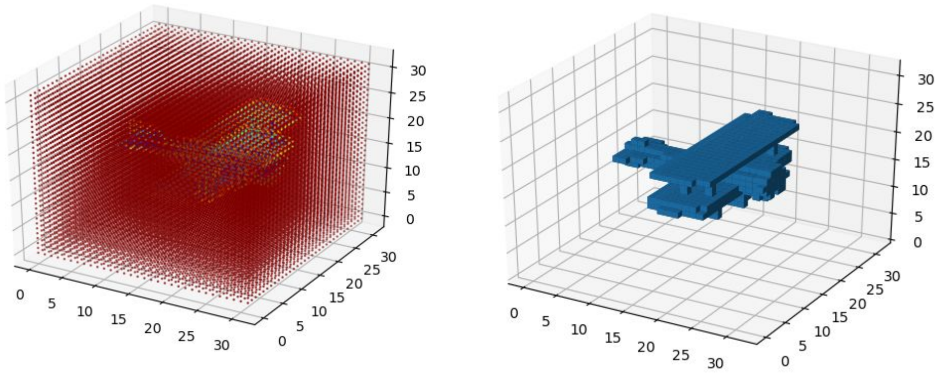


Figure 4: Raw DF (left) and thresholded DF (right) target tensor.

Data Partitioning. The partitioning algorithm creates the training, validation, and test splits at a $60:20:20$ ratio, yielding $18582,\ 6192,\ 6200$ unique shape models across all classes in each split, respectively. Since there are seven samples per shape model, the partitions ensure no overlap of shape model samples in the different splits. The class sample distribution is presented in Tab. 1. Additional visualizations are provided in Appendix A.

Table 1: Class distribution of the partitioned dataset. Shape models are unique within each split.

Dataset	**Chair**	**Table**	**Lamp**	**Car**	**Cabinet**	**Airplane**	**Couch**	**Boat**
Train	28800	28800	11120	28448	7544	19416	15224	9304
Valid	9600	9600	3704	9480	2512	6472	5072	3096
Test	9600	9600	3720	9488	2520	6472	5088	3112

5. Architecture

We propose two CNN models to compare against the baseline that incorporate a unique set of 3D convolutional modules. In particular, the CNNs are modeled after an encoder/decoder architecture which produces outputs with equivalent dimensions to the input. The first model, Residual U-Net, features an encoder composed of *Residual Blocks* [12], a decoder composed of *DoubleConv Blocks*, and skip connections that *add* the encoder's voxel features to those of the decoder at various scale spaces. The skip connections help to maintain the fine-grained details of the shape's input, and in combination with the Residual Blocks promotes fast and stable learning with improved gradient flow to the CNN encoder. Voxel-wise a*ddition* is preferred to *concatenation* for computational efficiency and to reduce the kernel sizes in the decoder. The second model, SE Residual U-Net, has an encoder built from *Squeeze and Excite* (SE) *Residual Blocks* [13], but is otherwise identical to the first model. The SE blocks enable the network to learn inter-channel dependencies that could exploit complementing voxel features at minimal cost. The structure of these CNNs are presented in Fig. 5.

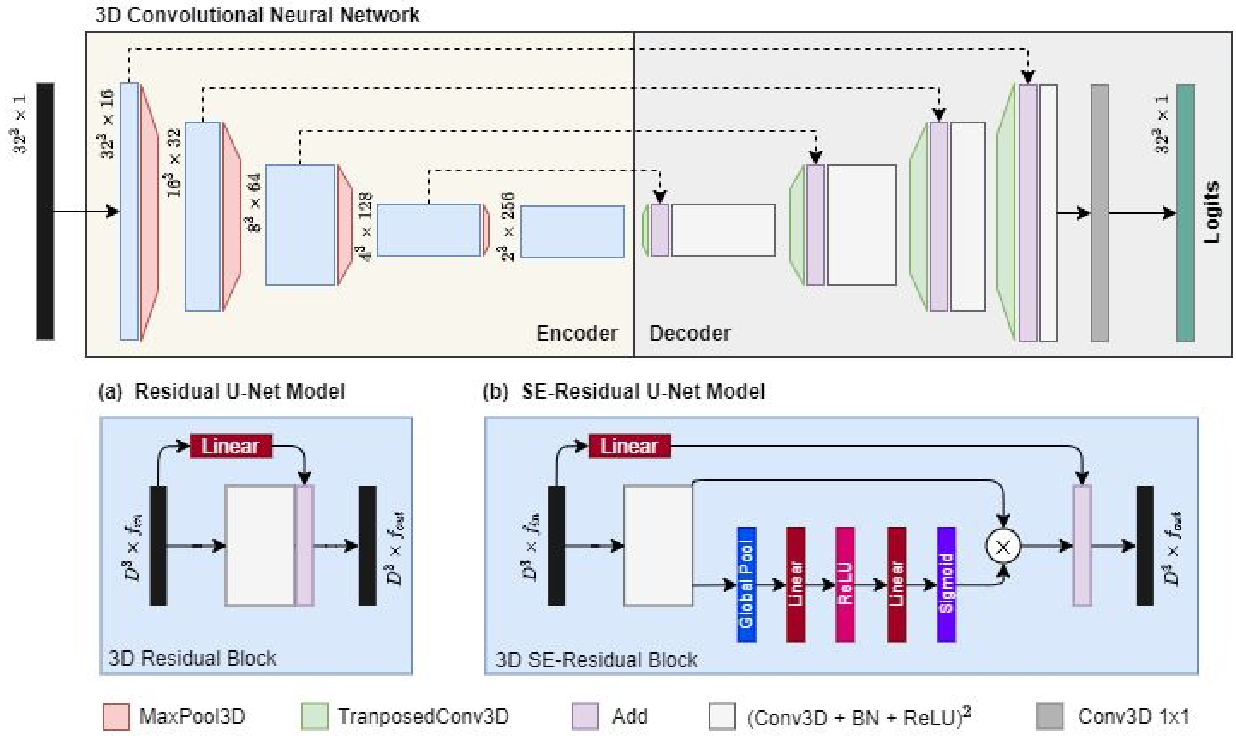


Figure 5: Final shape completion models - Residual U-Net and SE Residual U-Net. Hyperparameters are specified in Appendix B.

6. Baseline Model

For the baseline model we have selected 3D U-Net [14]. This network can learn from densely annotated DF targets and trains in a relatively short time. It has shown impressive results for dense volumetric segmentation in the context of medical imaging and the CNN can be adapted for the task of shape completion with reasonable effort, which makes it a desirable baseline model. The encoder / decoder of the 3D U-Net is constructed from single convolution blocks, and skip connections *concatenate* encoded voxel features to those of the decoder at each scale space. While the network architecture is lightweight, we contend that single convolutional blocks in the decoder is insufficient to reason about the full shape geometry from partial inputs.

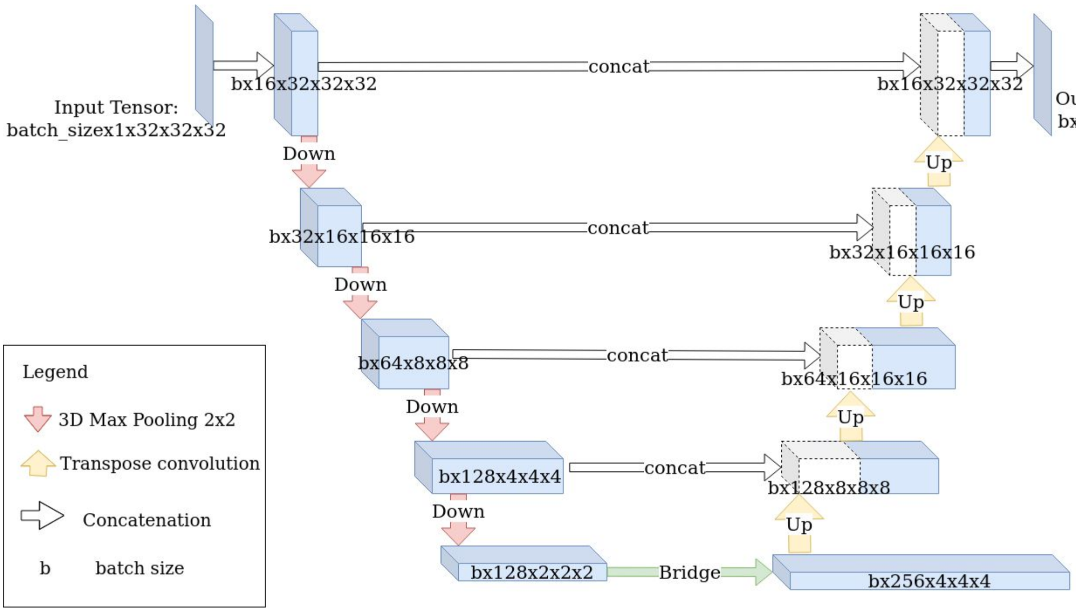


Figure 6: Baseline shape completion model - 3D U-Net [14].

7. Quantitative Results

Since our models learn to predict a dense DF of the completed shape, the L2-error metric over all voxels best quantifies the deviation of our predictions from the ground-truth DF. We directly optimize for low L2-error scores through the Mean Squared Error (MSE) loss function, which we use to train all of our models. For predicted and target DFs $\hat{y}_i$ and y_i, the MSE loss is computed over a set of N samples as follows:

$$L_{MSE} = \frac{1}{N}\sum_{i=1}^{N} L^{i}{}_{2} = \frac{1}{N}\sum_{i=1}^{N}\sum_{v=1}^{V}(\hat{y_{i,v}} - y_{i,v})^2$$

Here, $V = 32^3$ are the number of voxels in one sample. While the MSE loss and L2-error are computed over the dense DFs, we consider two more performance metrics that more closely reflect the shape of the object. The voxel-wise accuracy and intersection of union (IoU) are two strong indicators of the models' ability to predict visually meaningful shapes. However, they require the DFs to be thresholded into the form of binary volumetric occupancy (as shown in Fig. 4). The accuracy is computed over thresholded DF predictions ($\hat{f}_i$) and targets ($f_{i,}$) with:

$$Acc = \frac{1}{N\times V}\sum_{i=1}^{N}\sum_{v=1}^{V}\hat{f_{i,v}} \oplus f_{i,v}\text{, where } a \oplus b = 1 \text{ if } a == b\text{, else } 0$$

As most of the volume is empty after thresholding, the accuracy tends to quickly converge to high scores which can be misleading. This is addressed in the voxel-wise IoU which jointly considers the precision and accuracy of the positive class (e.g. occupied):

$$IoU = \frac{1}{N}\sum_{i=1}^{N}\frac{\hat{f}_i \cap f_i}{\hat{f}_i \cup f_i} = \frac{TP_i}{TP_i+FP_i+FN_i}$$

Where the number of true positives, false positives, and false negatives are denoted by, TP , FP , and FN, respectively. We compare the learning curves for each performance metric in Figures 7-10.

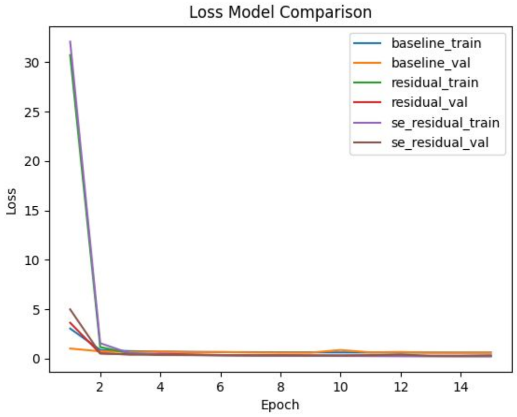


Figure 7: MSE Loss Curves

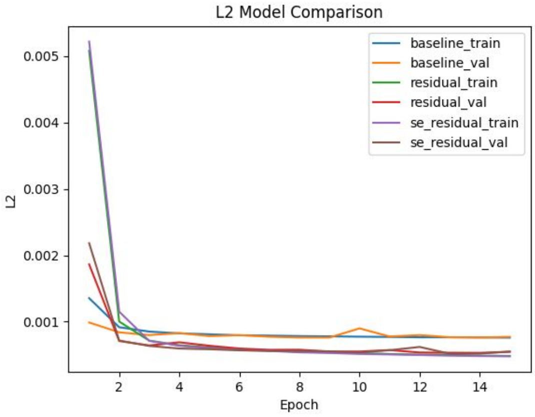


Figure 8: L2 Error Curves

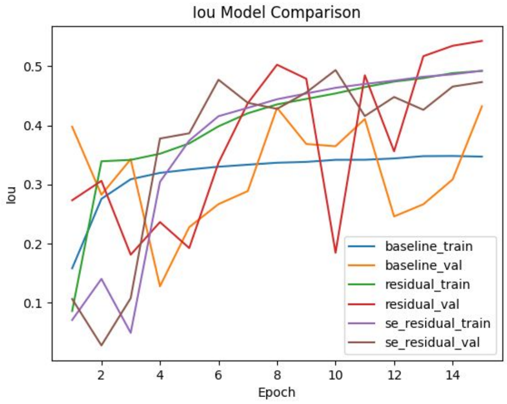


Figure 9: IoU Curves (*threshold* = 0.5)

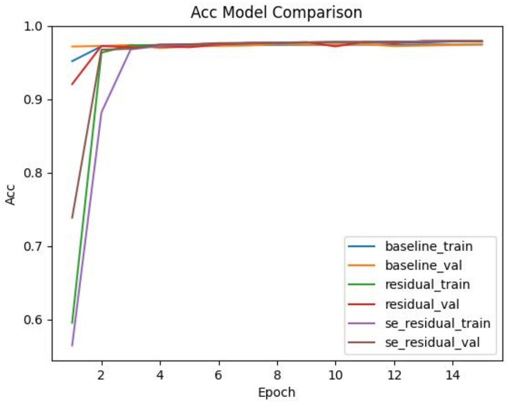


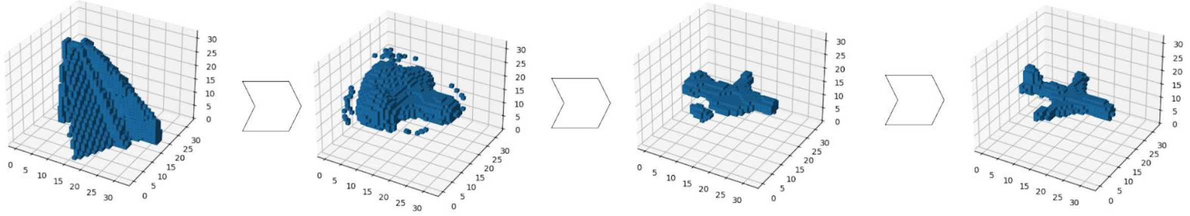
Figure 10: Acc Curves (*threshold* = 0.5)

Table 2: Validation set results of top performing model checkpoints for each architecture. Speeds correspond to deployment on an Nvidia GPU RTX-2080 Super.

Model	**L2 Error**	**IoU**	**Accuracy**	**Loss (MSE)**	**Speed (Hz)**	**Parameters**
3D U-Net Baseline	7.72e-04	4.32e-01	9.75e-01	6.35e-01	85.47	2.12e+07
Residual U-Net	5.47e-04	**5.43e-01**	**9.79e-01**	3.22e-01	**91.91**	**9.64e+06**
SE-Residual U-Net	**5.36e-04**	4.77e-01	**9.79e-01**	**3.09e-01**	90.09	9.66e+06

8. Qualitative Results

We generate visualizations of our model's predictions at the end of each epoch. A sample airplane being predicted over the course of training is presented in Figure 11.



Epoch 0 (Original)

Epoch 5

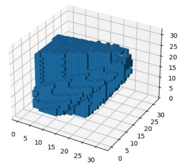
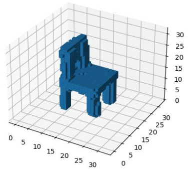
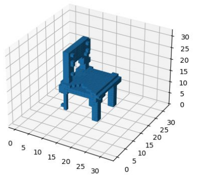
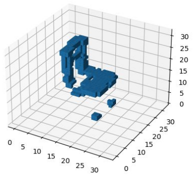
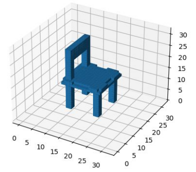
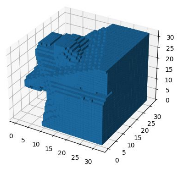
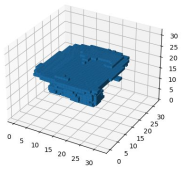
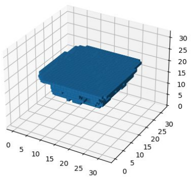
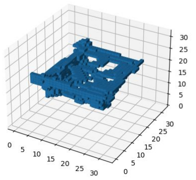
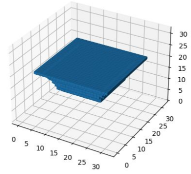
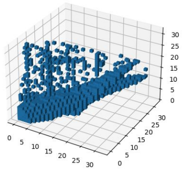
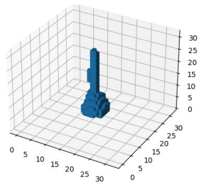
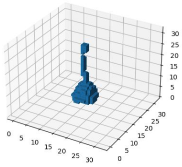
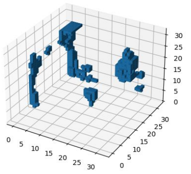
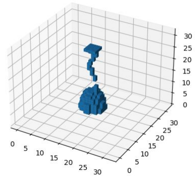
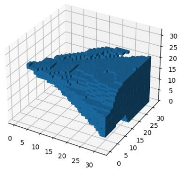
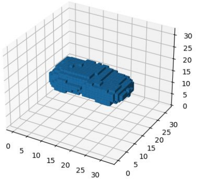
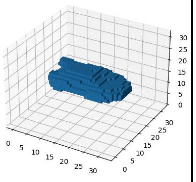
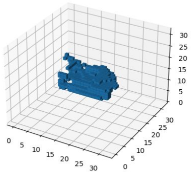
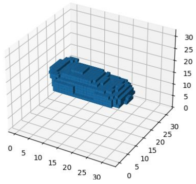
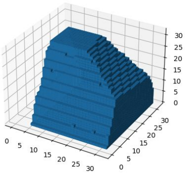
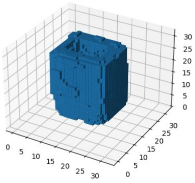
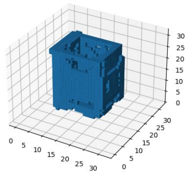
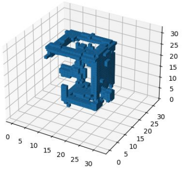
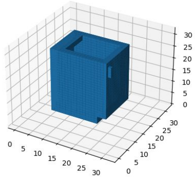
Epoch 10

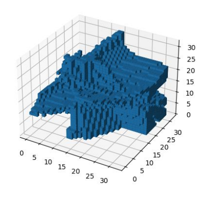
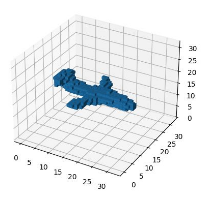
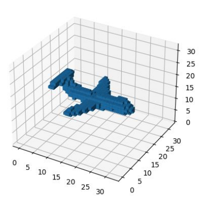
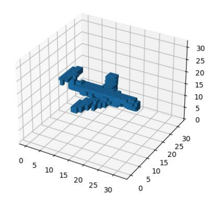
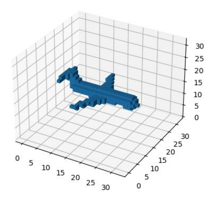
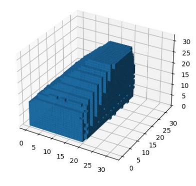
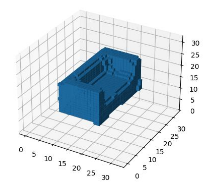
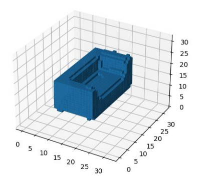
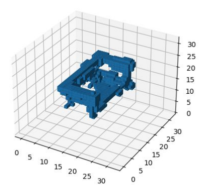
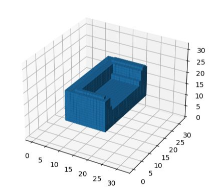
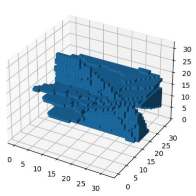
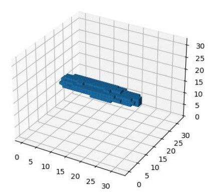
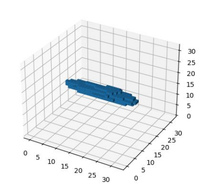
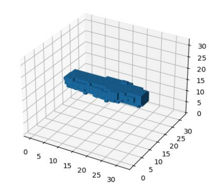
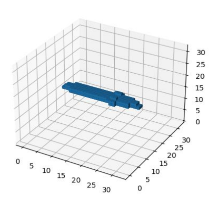
Epoch 15

Figure 11: Training Evolution of a Sample Airplane

We take the same models that produced metrics in quantitative results to show the final visualizations Several of the results of the individual categories are hence modelled in Tab. 3. We chose these selections because they were a clear representation of every class in our dataset. Furthermore, they also demonstrated the limitations of class that had a fine grained structure.

Table 3: Sample qualitative results on the validation split across all models.

Class	**SDF Input**	**3D U-Net**	**Residual U-Net**	**SE Residual U-Net**	**DF Target**
Chair					
Table					
Lamp					
Car					
Cabinet					

Airplane					
Couch					
Boat					

9. Model Evaluation on New Data

Integrity of Testing Process. Model generalization was a fundamental component of this project. Several measures were taken to ensure that reliable quantitative and qualitative results could be acquired at our convenience, and which would accurately reflect the models' ability to generalize to new samples. As described in Sec. 4, the test partition of the dataset does not contain any samples from shape models present in the training or validation splits. In all experiments, we limited the testing of each model to a single run to prevent the possibility of indirectly fitting to the test set samples.

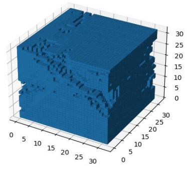
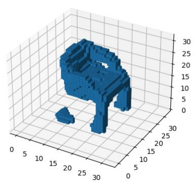
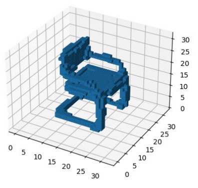
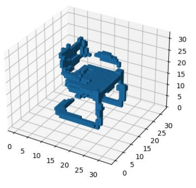
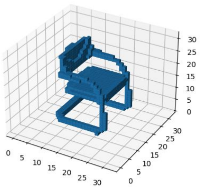
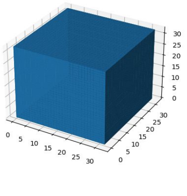
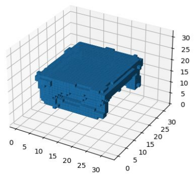
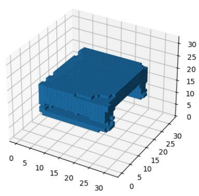
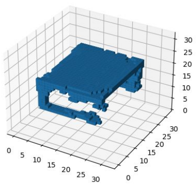
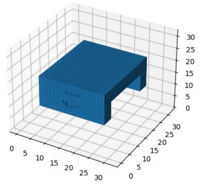
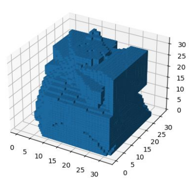
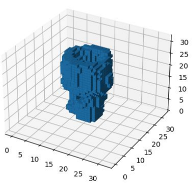
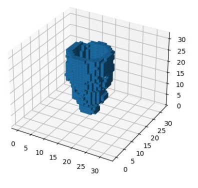
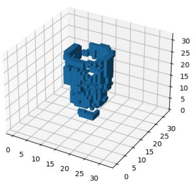
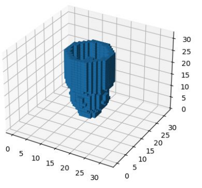
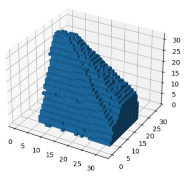
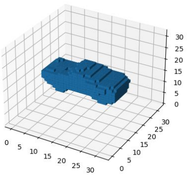
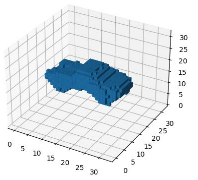
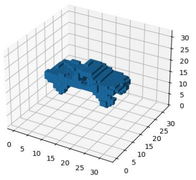
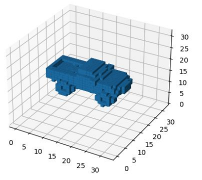
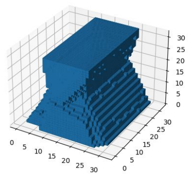
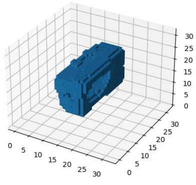
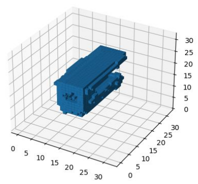
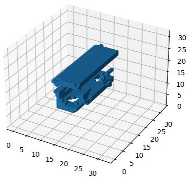
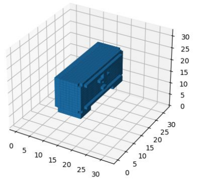
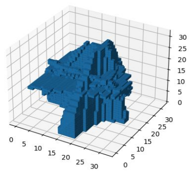
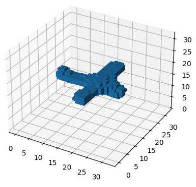
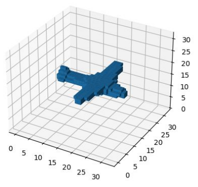
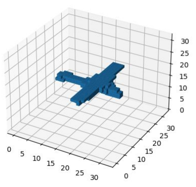
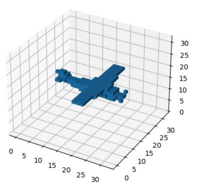
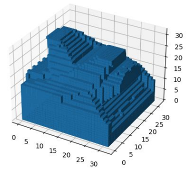
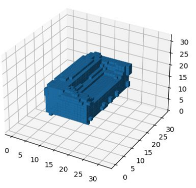
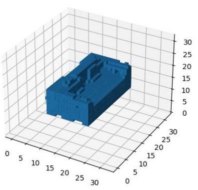
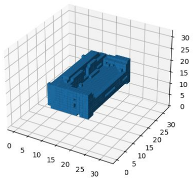
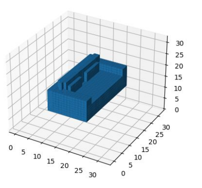
Test Results. The results of the top performing model checkpoints (based on validation IoU) are listed in Tab. 4 for each of the considered models. Both Residual U-Net and SE Residual U-Net outperform the baseline model over all metrics. These results were generated via an independent evaluation pipeline that loads the provided model checkpoint and tracks its performance over the test set (*evaluate.py*).

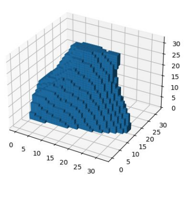
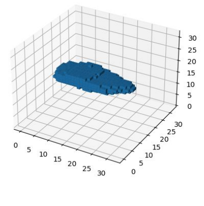
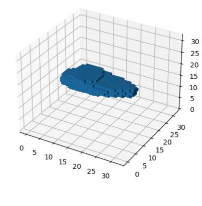
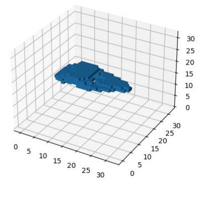
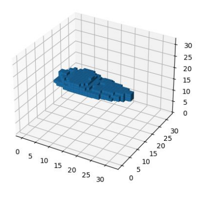
Table 4: Test set results of top performing model checkpoints for each architecture.

Model	**L2 Error**	**IoU**	**Accuracy**	**Loss (MSE)**
3D U-Net Baseline	7.92e-04	4.66e-1	9.70e-1	6.69e-01
Residual U-Net	5.50e-04	**5.42e-1**	**9.79e-1**	3.26e-01
SE-Residual U-Net	**5.41e-04**	4.75e-1	**9.79e-1**	**3.15e-01**

The evaluation pipeline also generates visualizations of a given number of samples per class. Note that these samples were selected at random initially, but are consistent across all tests to ensure that we can adequately compare our models qualitatively. We provide the predictions from our networks and the baseline model on one sample per class in Tab. 5.

Table 5: Sample qualitative results on the test split across all models.

Class	**SDF Input**	**3D U-Net**	**Residual U-Net**	**SE Residual U-Net**	**DF Target**
Chair					
Table					
Lamp					
Car					
Cabinet					
Airplane					
Couch					

Boat					

10. Discussion

The two models under consideration both outperformed the baseline quantitatively and qualitatively as seen in Sec. 7-8. With less than half the parameters of 3D U-Net baseline, Residual U-Net achieved a 6.7% increase in IoU, the key metric in evaluating the models' quantitative performance, as described in Sec 7. This indicates that implementing the skip connections through *addition* (ours) instead of *concatenation* (baseline) is more beneficial for shape completion, as well as more memory efficient. Interestingly, the increase in inference speeds of our models did not exceed 7% despite the substantial memory savings. We hypothesize that this is a result of architectural similarity; since convolving the relatively large kernels in the baseline decoder can be highly parallelized in the GPU, these additional parameters do not heavily degrade the network's forward pass speed. Overall, Residual U-Net and SE Residual U-Net are evidently a stronger choice for shape reconstruction in resource constrained operational settings.

We observed that the IoU scores of SE Residual U-Net were marginally worse than the standard Residual U-Net, although it's loss and L2-error scores on the validation and test sets were lower (see Tab. 2 and 4). As IoU and accuracy are computed over the voxelized DF, this leads us to believe that the empirically determined threshold value (i.e. transforming the DFs to volumetric occupancy grid) contains minor imprecision. Thus, while SE Residual U-Net has the strongest capacity to generalize and produce the highest quality DFs, it's predictions are not as visually aesthetic when compared to standard Residual U-Net.

We observe that the models have difficulty capturing grained-details of several objects. For instance, the tail of the airplane predicted by the SE-Residual U-Net in Tab. 5 is missing a rudder. The rudder, despite its small size, is a crucial detail of this object . The importance of small details shows that 3D reconstruction is inherently a difficult problem. Furthermore, there were some objects categories that performed better than others. While the model performed well on planes, it had difficulties constructing an accurate output for a chair; there were more transportation related objects in the dataset.

In the future, the model can be modified to incorporate classification predictions as a categorical prior for shape completion. This would mean building a "two step" model where the first step would be a classifier, and the second step would be the current encoder/decoder model. The outputs of the classifier would be used as an additional input to the reconstruction network.

11. Ethical Considerations

The dataset is mainly catered to the Western (North America + Europe) point of view and consists of broad categories. Underrepresentation of objects from other cultures and other categories may result in learning bias toward a more Western depiction of missing parts of those objects (for example, Japanese table kotatsu vs European table). The same reasoning can be applied to unique design solutions that the model may attempt to cater to the “average” case, and destroy their essential tangible and intangible parts.

A deep learning-based solution for 3D-to-3D reconstruction is unlikely to affect the job market as reconstruction is typically generated from 2D samples. However, other models that extend our work may harm opportunities for 3D animation artists; a sector, which in 2018 employed 70,000 individuals in Canada alone [15].

References

[1] A. Monszpart, N. Mellado, G. J. Brostow, and N. J. Mitra, “RAPter,” *ACM Transactions on Graphics*, vol. 34, no. 4, pp. 1–12, 2015.

[2] Y. M. Kim, N. J. Mitra, D.-M. Yan, and L. Guibas, “Acquiring 3D indoor environments with variability and repetition,” *ACM Transactions on Graphics*, vol. 31, no. 6, pp. 1–11, 2012.

[3] Y. Li, A. Dai, L. Guibas, and M. Nießner, “Database-Assisted Object Retrieval for Real-Time 3D Reconstruction,” *Computer Graphics Forum*, vol. 34, no. 2, pp. 435–446, 2015.

[4] D. Werner, A. Al-Hamadi, and P. Werner, “Truncated Signed Distance Function: Experiments on Voxel Size,” *Lecture Notes in Computer Science Image Analysis and Recognition*, pp. 357–364, 2014.

[5] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, and J. Xiao, "Shapenet: An information-rich 3d model repository." *arXiv preprint arXiv:1512.03012* (2015).

[6] A. Dai, C. R. Qi, and M. Niebner, “Shape Completion Using 3D-Encoder-Predictor CNNs and Shape Synthesis,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[7] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, “Semantic Scene Completion from a Single Depth Image,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[8] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” *Lecture Notes in Computer Science Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pp. 234–241, 2015.

[9] L. Shice, Y. Hu, Y. Zeng, Q. Tang, B. Jin, Y. Han, and X. Li, "See and think: Disentangling semantic scene completion," In *Advances in Neural Information Processing Systems*, pp. 263-274. 2018.

[10] Garbade, M., Chen, Y., Sawatzky, J., & Gall, J. (2019). Two Stream 3D Semantic Scene Completion. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. doi:10.1109/cvprw.2019.00055

[11] Shape Completion using 3D-Encoder-Predictor CNNs and Shape Synthesis. (n.d.). Retrieved December 05, 2020, from http://graphics.stanford.edu/projects/cnncomplete/data.html

[12] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. doi:10.1109/cvpr.2016.90

[13] Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-Excitation Networks. *2018 IEEE/CVF*

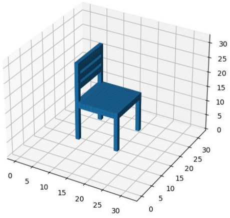
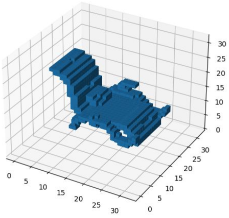
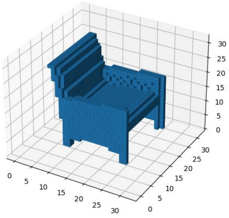
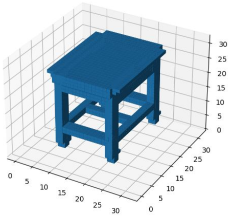
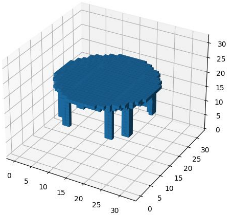
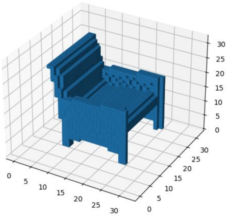
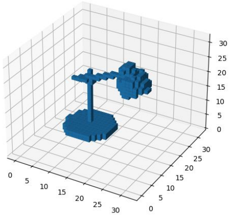
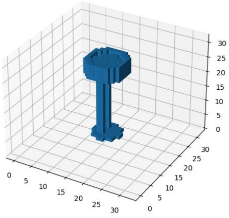
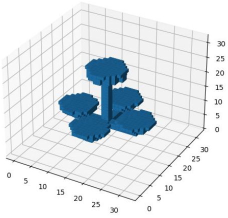
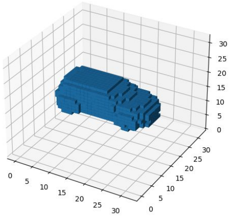
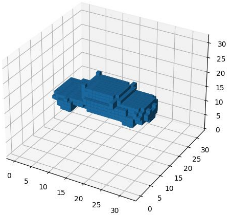
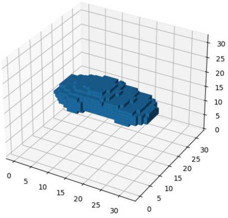
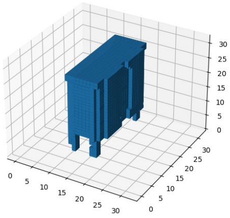
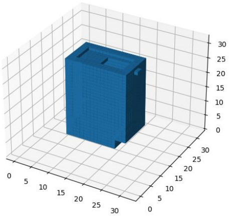
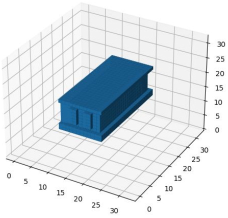
[14] *Conference on Computer Vision and Pattern Recognition*. doi:10.1109/cvpr.2018.00745

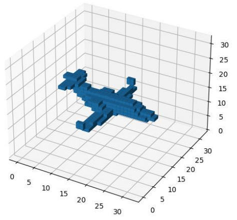
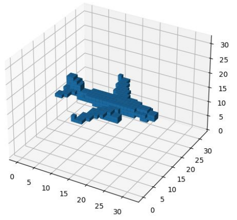
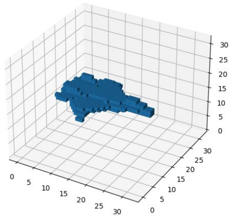
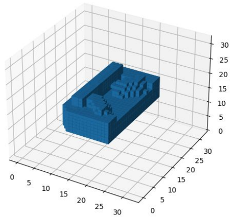
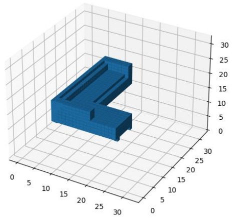
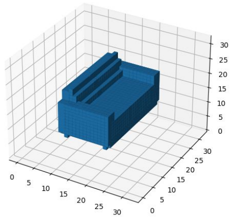
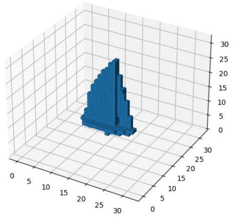
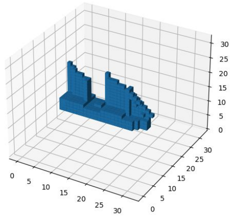
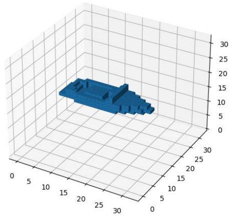
Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, “3D U-Net:
Learning Dense Volumetric Segmentation from Sparse Annotation,” *Medical Image
Computing and Computer-Assisted Intervention – MICCAI 2016 Lecture Notes in*
[15] *Computer Science*, pp. 424–432, 2016.

“Government of Canada / Gouvernement du Canada,” *3D Animation Artist in Canada | Job outlook*, 02-Oct-2020. [Online]. Available: https://www.jobbank.gc.ca/marketreport/outlook-occupation/5730/ca. [Accessed: 13-Oct-2020].

Appendix A. Dataset Visualization

Table A-1. Sample thresholded distance field targets per class.

Chair	 0 5 10 15 20 25 30 0 5 10 15 20 25 30 0 5 10 15 20 25 30	 0 5 10 15 20 25 30 0 5 10 15 20 25 30 0 5 10 15 20 25 30	 0 5 10 15 20 25 30 0 5 10 15 20 25 30 0 5 10 15 20 25 30
Table	 0 5 10 15 20 25 30 0 5 10 15 20 25 30 0 5 10 15 20 25 30	 0 5 10 15 20 25 30 0 5 10 15 20 25 30 0 5 10 15 20 25 30	 0 5 10 15 20 25 30 0 5 10 15 20 25 30 0 5 10 15 20 25 30
Lamp	 0 5 10 15 20 25 30 0 5 10 15 20 25 30 0 5 10 15 20 25 30	 0 5 10 15 20 25 30 0 5 10 15 20 25 30 0 5 10 15 20 25 30	 0 5 10 15 20 25 30 0 5 10 15 20 25 30 0 5 10 15 20 25 30
Car	 0 5 10 15 20 25 30 0 5 10 15 20 25 30 0 5 10 15 20 25 30	 0 5 10 15 20 25 30 0 5 10 15 20 25 30 0 5 10 15 20 25 30	 0 5 10 15 20 25 30 0 5 10 15 20 25 30 0 5 10 15 20 25 30
Cabinet	 0 5 10 15 20 25 30 0 5 10 15 20 25 30 0 5 10 15 20 25 30	 0 5 10 15 20 25 30 0 5 10 15 20 25 30 0 5 10 15 20 25 30	 0 5 10 15 20 25 30 0 5 10 15 20 25 30 0 5 10 15 20 25 30

Airplane			
Couch			
Boat			

Appendix B. Training Details

Table B-1. Training scheme details and hyper-parameters for all models. LR: Learning rate; DR: Weight decay (L2); BSZ: Batch Size; EPO: Number of epochs.

Model	**LR**	**DR**	**BSZ**	**EPO**	**Details**
3D U-Net	1e-3	5e-4	32	15	N/A
Residual U-Net	1e-4	3e-4	32	15	N/A
SE Residual U-Net	5e-5	5e-4	32	15	SE Residual Block squeeze rate range from $r = [4,\ 8,\ 16,\ 16,\ 16]$ at each level in the encoder.