Automatic Prediction of Affordance-Preserving 3D Meshes to Improve Interactive Robotics Simulation

Emily Broadhurst
ebroad23@stanford.edu

Tommy Bruzzese
tbru@stanford.edu

Sean Bai
seanbai@stanford.edu

Cem Gokmen
cgokmen@stanford.edu

Stanford University

Abstract

*Realistic robotic and virtual interactions require high-fidelity meshes that preserve object affordances like knobs and handles. However, high fidelity meshes require more computation. Our project evaluates how to predict mesh type such that affordances are preserved for complex objects without overshooting the fidelity needed for simpler objects. We evaluate three approaches trained on the BEHAVIOR-1K dataset [6]: using a non-finetuned ResNet18 with cosine similarity predictions, integrating an MLP classifier with frozen ResNet18 weights, and end-to-end training of the ResNet18 and MLP. We also explore weighted sampling to reduce overfitting. In a 1639 object test set from the BEHAVIOR-1K dataset, our best model — a End-to-End ResNet with a MLP Classifier and Weighted Sampling — predicts the expert-labeled correct mesh type with **47.3**%. We also introduce a new metric of "affordance-preservation" where different mesh techniques with the same level of detail (e.g., "V-HACD-32" and "Co-ACD-32") both count as correct, and achieve **83.5**% preservation. Our results indicate that end-to-end training, combined with techniques such as weighted sampling, can substantially improve the generalizability and scalability of automatic mesh prediction for real-world scenes, advancing the capabilities of virtual and robotic systems for more dynamic and responsive interactions.*

1. Introduction

In virtual environments and robotic interactions, the geometric representation of objects is crucial for realistic and functional interactions. A key challenge in this domain is the efficient development of affordance-preserving collision meshes from generic meshes. For example, meshes of drawers need high-fidelity handles and stove-tops mesh need high-fidelity knobs in order for robots or video game users to realistically interact with them. Objects represented in too low of fidelity cannot be readily interacted in a scene, heavily limiting their usefulness.

The creation of realistic interactive scenes is a significant challenge in robotic simulation environments like BEHAVIOR-1K [6]. Traditionally, these scenes are crafted by 3D artists and manually annotated by robotics researchers, a process that is both expensive and time-consuming. Procedural generation methods such as ProcTHOR have attempted to address these issues but typically produce unrealistic and repetitive scenes that do not accurately represent the diversity of real-world environments [3]. This affects the performance of robotic policies trained in such settings, limiting their applicability to real-world conditions.

The generation of specialized affordance-preserving meshes often relies on Approximate Convex Decomposition (ACD) techniques, with tools like V-HACD [9] and Co-ACD being predominant [12]. However, these tools require specific settings and extensive parameter tuning to achieve the best result for each object, presenting a barrier as no single ACD method is universally applicable to all objects. Furthermore, while complex objects may require more complex mesh representations (e.g., with more hulls), it is a waste to use too many hulls on an object that would have all its affordances preserved with only a few hulls. Especially in scenes with hundreds or objects, prioritizing complex meshes for only complex objects becomes critical for efficiency.

In response, our project seeks to develop a scalable and adaptable system that can automatically determine the most effective ACD technique and parameters for creating optimal affordance-preserving meshes across thousands of objects, without needlessly overshooting the hull fidelity needed.

We train on a dataset of 6543 objects from BEHAVIOR-1K. In our pre-processing, we generate 14 differently-

angled renders of each object across various mesh candidates and pass them through various algorithms with ResNet's and MLP's to predict the ACD technique candidate that human experts rated as best for each object.

Our project aims to automate the ACD mesh generation process and ensure the quality and functionality of interactions in both digital and physical environments. By addressing these challenges, our work intends to advance the capabilities of virtual and robotic systems, making them more dynamic and responsive.

2. Related Work

Prior work has explored various traditional ACD and machine learning approaches to improve the scalability and efficiency of mesh generation in simulation. Lien et al. introduced the idea of Approximate Convex Decomposition (ACD) technqiues by providing a robust framework for decomposing polyhedra into approximately convex parts [7]. However, their method can be highly sensitive to initial conditions. Mamou et. al, provided another early approach with V-HACD, providing a relatively less expensive approach that struggled to handle highly concave shapes effectively [8, 9]. Wei et al. explored a Co-ACD method of collision-aware concavity that improved decomposition quality [12]. However, their method requires extensive parameter tuning to best preserve affordances, and is more computationally expensive.

ACD techniques are fundamentally able to create high-fidelity, affordance-preserving meshes, but they do so at the cost of high computational complexity and extensive parameter tuning — which is an issue for scalability in real-world simulation. No single ACD method is universally applicable to all objects in a way that will preserve meshes with overshooting computation.

Li et al. introduce the BEHAVIOR-1K dataset, which includes a large number of manually-created and annotated meshes that highlight the challenge of creating realistic interactive scenes for robotic simulation [6]. Manual mesh creation and selection for objects is time-consuming and not scaleable; however this dataset of highly-curated objects and corresponding affordance-preserving ACD meshes has proven incredibly useful for machine learning training. This is in tandem with the Objaverse dataset from Deitke et al that offers hundreds of thousands of 3D models that can be used to train machine learning models; however at such scale, these objects are notably less curated, making them useful especially for testing robustness [2].

Outside of ACD techniques, there has been research in machine learning approaches to mesh generation. Chen et al. uses a neural network-based approach to automate mesh generation, reducing the need for manual intervention [1]. However, the accuracy of the model is dependent on high quality training data for each object class, making this approach un-scalable in real-world simulations. While Wu et al. leverages 2D images to generate 3D shapes to reducing the need for large 3D datasets across object classes, their overall accuracy remains relatively lower [13].

Furthermore, Kalogerakis et al. present a data-driven approach to 3D mesh segmentation that significantly reduces manual effort by learning from annotated examples [5]. Although effective, again, reliance on high-quality labeled data poses across each object class presents scalability challenges. Huang et al. propose a novel method for 3D shape analysis using deep learning, which improves the precision of mesh generation but still struggles with generalization across diverse object categories [4]. Rusu et al. introduce a method that integrates point cloud processing with machine learning to create detailed 3D representations, providing high accuracy but at the expense of significant computational resources [11]. Building on point cloud processing, Qi et al. develop PointNet, a neural network that directly processes 3D point clouds for object recognition and segmentation, demonstrating robust performance but requiring extensive computational power [10].

While traditional methods like manual annotation and ACD techniques have laid the foundation for generating affordance-preserving meshes, there is no one ACD technique that works across objects. Similarly recent advancements in machine learning offer promise to automatic mesh creation, but they also face challenges related to data requirements and model accuracy that prevent object generalization.

In comparison, our work aims to build a scalable ACD technique classifier for each object, combining the traditional ACD techniques with a machine learning classifier of which technique to use. Unlike the other machine learning methods listed, our approach does not seek a new mesh representation and instead automates the selection of ACD techniques and parameters, reducing the need for extensive manual tuning across each new object class.

3. Data

Stanford's Vision and Learning Lab has worked with human experts in robotics to select, for 8,207 objects in the BEHAVIOR 1K dataset, which mesh was the most affordance-preserving without being unnecessarily complex.

They relied on expert human users to select these meshes, because creating an automatic metric to verify affordances proved too difficult: With the incredible amount of affordances that exist, it is hard to automatically detect if handles, knobs, latches, buttons, and more are preserved.

Each object in the dataset has multiple technique/parameter options that the experts chose between, including Convex Hull, Bounding Box, three variations of V-HACD with distinct parameters (4, 8, 32-hull versions),

two configurations of CoACD (8- and 32-hull versions), or a manual human-made mesh.

Mesh Type	# of Hulls	% of Dataset	Affordance Preserving for
CHull	1	26.72%	CHull, Bbox
Bbox	1	6.95%	CHull, Bbox
V-HACD-4	4	11.22%	CHull, Bbox, V-HACD-4
Co-ACD-8	8	18.36%	CHull, Co-ACD-8, V-HACD-4, V-HACD-8
V-HACD-8	8	7.06%	CHull, Co-ACD-C8, V-HACD-4, V-HACD-8
Co-ACD-32	32	22.97%	CHull, Co-ACD-C8, V-HACD-4, V-HACD-8, Co-ACD-32, V-HACD-H32
VHACD-32	32	4.31%	CHull, Co-ACD-8, V-HACD-4, V-HACD-8, Co-ACD-32, VHACD-32
Manual	32+	2.41%	CHull, Co-ACD-C8, V-HACD-4, V-HACD-8, Co-ACD-32, V-HACD-32, Manual

Table 1. The distribution of meshes in the training and test set (which were created with the same distribution), as well as the mesh hull complexity and list of mesh types it preserves affordances for. Given that "manual" was selected by expert users only when none of the other meshes were satisfactory, we represent it as "32+" meshes. We note however that while we train on Bbox and CHull separately (and find that sampling from them independently is especially useful for weighted sampling), our evaluation for accuracy and affordance-preservation computes CHull/Bbox as the same class given their high similarity, with a total distribution of 33.67%.

We divided this dataset into training,and test sets with $6,543$ objects (80%)in the training set, and $1,639$ objects (20%) in the test set (we threw out 25 objects that failed in our render pipeline due to material file issues).

For additional testing, we also selected 50 objects from the Objaverse dataset to use as an additional metric for the robustness of our approach (however 17 of these also failed in our rendering pipeline, leaving a total of 33 objects). Given that we used the Objaverse dataset to test generalization, we assigned a difficulty score predictability of meshing on a scale of 1 to 4, where 1 represented an object that was easy to mesh and preserve affordances (e.g., a book) and 4 represented more difficult objects to notice all affordances (e.g., a candelabra). Overall though, the meshes in Objaverse are less highly-curated by expert roboticists, meaning they inherently represent a more difficult challenge across all difficulty scroes.

Our dataset included .obj files for the original "visual render" mesh, as well as each of the 7 candidate meshes (e.g., CHULL, V-HACD, etc.). We did not have the "manual" mesh when that was selected by experts. Our first preprocessing step was to generate consistent renders of each object, for each mesh, from various angles. Image capture involved generating 14 images per object, including 6 orthographic views (top, bottom, left, right, back, front) and 8 isometric views at the corners, using pyrender and trimesh. To view an example of this refer to **??**. Each image was normalized using the mean and standard deviation of the ImageNet dataset, and all objects were centered and scaled to fit within a 400x400 pixel frame to maintain uniformity.

Generating usable renders was our first challenge. The source of truth .obj file contained materials and textures that made it easy to see the many faces and hulls of the object. The candidate meshes, however, did not come with textures, and initial renders had difficulty seeing the finer details. Given that the experts selecting the meshes viewed each hull in a different color, our solution was to similarly color individual hulls with a distinct color using the trimesh library. We generated randomized color values, and colored all the faces of the same hull with the same color, which ensured that each hull easily differentiable. This step is critical for the model to understand the number of hulls used in generating the approximation. An example output after this process is illustrated in 1.

Given that we generate 14 renders for each of the 8 mesh types in the dataset for each of the 8182 usable BEHAVIOR-1K objects, we have generated a dataset of 916,384 images — which oftentimes proved difficult to work with for efficient training.

Figure 1. Colored Hull Render vs. Uncolored

4. Methods

For each of our methods, we utilized a pre-trained ResNet18 model and removed its classification layer to use it as a feature extractor. We ran each of the 14 angles through the ResNet model and then concatenated them to form a comprehensive feature vector for a candidate mesh.

The core of our approach then involved training a classifier to identify the mesh technique and parameter combination that best preserves an object's affordances according to experts.

4.1. Non-finetuned ResNet

We first use the ResNet18 model to extract features from the candidate meshes and then, instead of training a classifier, we evaluate model performance based on cosine similarity. This established a reference point without finetuning or additional training, and was instrumental in developing a way to load the nearly 1 million image renders in our dataset efficiently.

The cosine similarity between candidate embeddings and the object's actually embeddings is computed to find the most similar embeddings for evaluation, where between two vectors A and B the similarity is defined as:

$$\text{cosine\_similarity}(A, B) = \frac{A \cdot B}{\|A\|\|B\|}$$

where

$$A \cdot B = \sum_{i=1}^{n} A_i B_i \quad \text{and} \quad \|A\| = \sqrt{\sum_{i=1}^{n} A_i^2}, \quad \|B\| = \sqrt{\sum_{i=1}^{n} B_i^2}$$

4.2. Frozen ResNet Weights with Trained MLP Classifier

For the next approach, we will add an MLP classifier on top of a pre-trained ResNet18, but keep the weights of the ResNet frozen during training. The baseline MLP will have a simple architecture of a linear layer, followed by an activation layer using ReLU, and then a final linear layer. This architecture is very simple to create an easy-to-implement baseline for us to enhance later. The ReLU layer is incredibly important to add non-linearity to help mitigate the vanishing gradient problem, and improve computational efficiency.

We will use binary cross-entropy (BCE) loss for training the MLP and classification selection. BCE loss is given by:

$$L = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)] \quad (1)$$

where:

- N is the number of samples,
- y_i is the true label of the i-th sample,
- p_i is the predicted probability of the i-th sample.

For most multi-category classification tasks, categorical cross-entropy loss is usually the standard over BCE. However, we found during experimentation that BCE consistently outperformed categorical cross-entropy loss in ensuring the selection of affordance preserving meshes. We hypothesize this might be due to: 1) Our dataset was quite imbalanced. The largest mesh category (CHull) out of 8 comprised over 30% of the dataset, while the smallest (manual) only around 2%. BCE may be equipped to handle such imbalances better than, as it evaluates each class independently, 2) Our classes are not meant to only be evaluate in a mutually-exclusive way. The aim of our model is to maximize affordance preservance by predicting the right fidelity of hulls needed, even if it may predict the exact ACD technique wrong. Further, many objects in our dataset are given the label 'CHull', meaning the the simplest mesh possible was sufficient, so in our "affordance-preservation" metric any other mesh predicted over CHull would still offer the same level of affordance preservation.

By freezing the ResNet weights, we ensure that only the MLP classifier learns during training, resulting in $\tilde{4}$ million parameters trained. Again, the pre-trained ResNet18 is used as a feature extractor. The MLP classifier is added on top for prediction of mesh technique and parameter combination.

This approach is more complex than the first, but we predict it will be ultimately worse than fully end-to-end training of also the ResNet feature extractor.

4.3. End-to-End Training with ResNet and MLP

For the final approach, we will train both the ResNet18 and MLP end-to-end with binary cross-entropy loss. This method is very similar to the prior one, except we now additionally train the $\tilde{1}1$ million parameters of the ResNet, leading to a total of around $\tilde{1}5$ million parameters trained.

The goal of this architecture is to understand the benefits of specifically fine-tuning the feature extractor for our task. The pre-trained ResNet on ImageNet is a substantially different task than selection of the best ACD technique for objects, and so we predicted that our additional fine-tuning to the task would significantly improve our performance.

This is our most computationally intensive and time-consuming approach, as it optimizes both feature extraction and the classification layer.

4.4. Training on Just Visual Renders

Given all of our pre-processing in producing the 14 angled renders for each of the 8 candidates meshes, we wanted to also evaluate the effects of this intensive dataset augmentation on our performance. Thus, while the majority of our trained models utilize the full dataset across all angles and all candidate meshes, we also trained a model using just 1 standard angle from the true visual render of the obj

4.5. Weighted Sampling

Given the unequal distribution of mesh classes (seen in Table 1, which is especially imbalanced towards a very low-fidelity mesh type, we perform weighted sampling during training over 7 class labels (because visual render is not a predicted class and manual is not an available class in training). This is done in an effort for the model to better learn to predict higher-fidelity meshes, an important goal of our work.

While we later describe how we combine CHull and BBox as the same class for evaluation, we continue to conduct our weighted sampling with CHull and Bbox as separate classes to pull from. We found that combining them in our sampling significantly reduced our accuracy as it too heavily redistributed the classes away from the low-fidelity objects.

5. Experiments and Discussion

Before we delve into the specifics of each experiment, we want to lead a brief discussion on our hyperparameter choices and primary metrics.

We decided to focus most of our time on configuring our architecture, which took longer than anticipated to due to the pre-process of all our 1 million images, leaving less time for hyperparameter search. We chose Adam as our optimizer due to its adaptive learning rates and its popularity in many deep-learning projects. We typically used a batch size of 4 objects in our training, as our computing power was not able to handle larger object batches: A batch size of 4 is actually 4 objects x 8 meshes x 14 rendered angles = 448 rendered images.

We have two primary metrics: the overall accuracy of predicting the exact expert-labelled mesh, as well as percentage of predicted meshes that preserved affordance fidelity. This metric is calculated by comparing the number of hulls in the predicted mesh compared to the true label. We choose to do Top-1 accuracy as only one mesh was selected by the expert evaluators.

We note as well that while we trained on Bbox and CHull as separate classes (particularly as described for weighted sampling to not too heavily reduce the distribution of this class by overlumping it together), our evaluation metrics treat CHull and Bbox as the same class for expert prediction and affordance preservation. We refer to this combined class just as "CHull" for simplicity.

If the predicted mesh has a hull number greater than or equal to the true mesh then it is affordance preserving (e.g., Co-ACD-32 preserves Co-ACD-8). The list of which predicted meshes preserve which true labels is provided in Table 1.

We note that it may seem overly generous to correctly label any mesh with equal or more hulls as correctly "affordance-preserving" as always predicting a model like Co-ACD-32 would offer near perfect preservation. However, it is a significant issue if an object's mesh has too low of fidelity to render its affordances: Drawers cannot be opened, stoves cannot be utilized, etc. A fundamental goal of our project is to ensure that as many objects are rendered with meshes that preserve their affordances. However, to also highlight the need for not universally picking such high-fidelity meshes like Co-ACD-32, we also combine our analysis with overall accuracy in an effort to show that our approach is able to better balance avoiding both under-fidelity and over-fidelity.

Other metrics such as precision and recall averaged across of the classes were evaluated as well.

5.1. Baselines

Before conducting any experiments, we calculated two baselines: (1) always predicting CHull, which maximizes accuracy, and (2) always predicting Co-ACD-32 which highly maximizes "affordance-preservation" without being unfair to accuracy. That is, while V-HACD-32 and Manual also are very high on "affordance-preservation" (with Manual being 100% affordances preserved), they are much worse on accuracy given their low-class distribution and are therefore less robust baselines to compare against.

We end up treating Frozen Resnet with Cosine Similarity as a baseline because it is not a trained model. It ended up performing so poorly, which makes sense given that cosine similarity between ResNet-extracted features is an improper loss metric, especially given that the ResNet is used for a very different task of ImageNet classification.

Our Frozen Resnet with Cosine Similarity achieved a final Top-1 accuracy of 24.16% and 46.97% affordance-preservation. We note that our preservation is better than CHull baseline, but is worse on accuracy, highlighting the limited effectiveness of the ResNet without fine-tuning in transfer learning. Notably, we can also see the trade-off between a worse accuracy score actually often being able to improve affordance preservation as incorrectly predicting the low-fidelity classes with higher-fidelity meshes still preserves them, even if it is inefficent. We also that without a separate classification layer, this Frozen Resnet with Cosine Similarity baseline could never have predicted "manual."

5.2. Frozen ResNet with MLP Classifier

Adding an MLP classifier led to significant improvements in accuracy over all the baselines with a Top-1 accuracy of 49.7%, and a significant improvement over the affordance preservation of CHull and Cosine Similarity baselines with a result of 62.48%. We trained this model for 1.8 hours over 10 epochs to reach this. The results of the experiment are pictured in Figures 2 and 4.

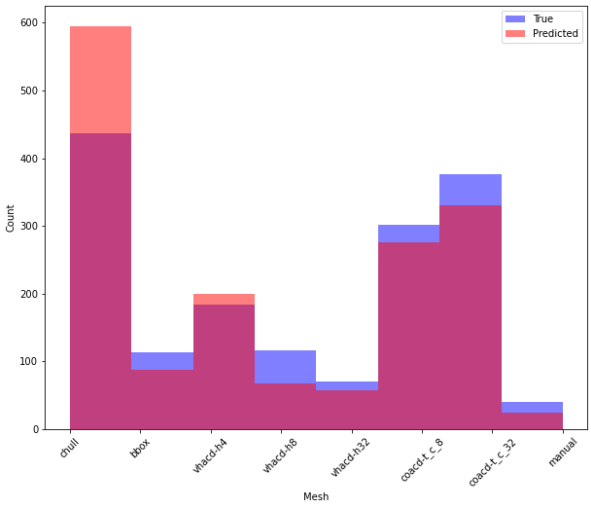

Figure 2. Frozen ResNet w/ MLP predicted counts across Classes

The largest failure case as illustrated in Figure 4 where the true label was Co-ACD-32 and the model predicted CHull. This is a common failure case with more complex objects where Co-ACD-32 is required to properly retain all

the affordances. This theory was confirmed by digging into the specific cases the model failed on, illustrated in 3. As we can see urinal-pfxmpd-base_link is classified as being best generated through CHull when it actually was Co-ACD-32.

The model is over-picking Chull which gives us improved accuracy but we fail in our task of generating affordance-preserving meshes. This is occurring because the CHull option is over-represented in the dataset. Next, we will try training end-to-end to see if that will boost preservation and solve our imbalanced class problem.

Before moving onto more complex models, we will analyze our training loss as shown in 8. The training loss is generally decreasing which indicates our model is learning and improving performance over time. There are, however, noticeable fluctations due to the stochastic nature of gradient descent. Towards the end of training, we see the loss is generally stablizing. The fluctuations suggests there is more room for hyperparameter tuning for smoother convergence.

5.3. End-to-End Training with Just Visual Renders

As we moved into our End-to-End training, which significantly adds to the amount of trained parameters, we first wanted to run our model with a reduced dataset to see how necessary all of our augmentation of rendered angles was. This approach reduced our accuracy and preservation rate from the Frozen ResNet w/ Cosine Similarity, which shows the large impact that our dataset augmentation has. This result verifies that our intesenve pre-processing was worth the trade-off of reducing time to do hyperparameter training and search.

5.4. End-to-End Training with ResNet18 and MLP

When we first trained this approach we got a Top-1 accuracy of 46.6% accuracy. This is extremely close to our previous Frozen ResNet model with the full dataset (slightly lower), which suggests that ResNet18 is not significantly boosting accuracy. This model took 2.2 hours to run over 10 epochs, however, the additional training time is not necessarily worth it. This can be further illustrated in Figure 5, which show the confusion matrix and predicted counts.

Again, we have our problem where we have the over-reliance on predicting CHull, illustrative of overfitting, as mentioned in the previous experiment, and observed again in Figure **??**. Our next approach to boost affordance preservation is to integrate weighted sampling during training to handle our imbalanced class problem that over-learns CHull.

5.5. End-to-End Training with ResNet and Weighted Sampling

Weighted sampling helped reduce the overfitting observed previously to the low-fidelity CHull mesh. We can see that this change was significant, leading to an increase of affordance preservation to 83.5%. We also saw an increase in our precision and recall scores to 0.435 and 0.373, respectively. The results can be analyzed more closely by looking at Figure 6 and 7. Plus this can be substantiated by digging into the specific cases the model failed on, illustrated in 3. As we can see urinal-pfxmpd-base_link was previously predicted with the Frozen ResNet as CHull when it actually was Co-ACD-32.

By adding weighted sampling it came at the cost of underpredicting CHull, as illustrated in 3 where object cupcake-pfwrlw-base_base_link was previously predicted correctly during our Frozen ResNet experiment and now is predicted as Co-ACD-8 when is it best generated through CHull. However, this trend was a cost that is outweighed by the benefit it added.

6. Limitations

After developing our best model as constructed during Experiment 5.5 we ran on our aforementioned Objaverse dataset. When picking out the dataset we found many of the objects in the dataset were pretty complex which we predicted could potentially fail as the average object in the Objaverse dataset seemed to be more difficult to preserve affordances due to geometric complexity. Thus, we ended up picking 6 objects rated 4 in difficulty, and 11 objects rated 3 in difficulty (medium-to-hard complexity) predicted due to the complexity of the geometry needed to be afforded. The remaining objects were rated 2 or 1 in difficulty.

Before we could even run, we found that 17 of those images, corresponding to the 17 objects that we initially picked as the nice-to-have more complex geometries failed, showing the limited technological capability of current tools to be able to create an affordance-preserving mesh.

We had some trouble generating these results, and hope to do so in our future work.

7. Conclusion

Our results demonstrate that we have developed a classifier capable of better balancing prediction of expert-selected meshes, while also significantly increasing affordance-preservation close to universally selecting 32-hull meshes. This is of major significance as it shows we can preserve affordances well, without necessarily sacrificing on computation efficiency and needlessly over-predicting the meshes needed.

We have also demonstrated the effectiveness of our intensive pre-processing of the renders, as well as the trade-offs of weighted sampling in largely improving affordance preservation with doing slightly worse on accuracy than other trained models without weighted samples.

Future work can include further hyperparameter tuning to enhance accuracy without significantly altering the ar-

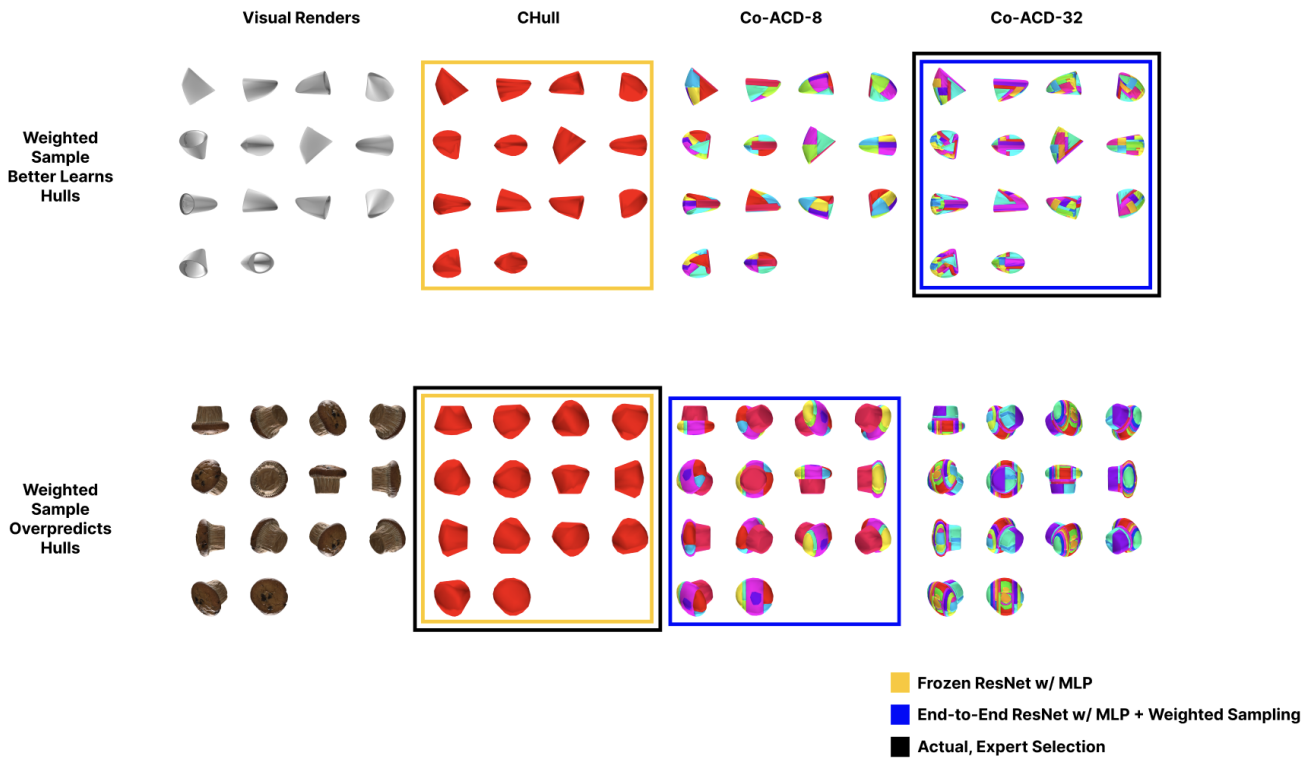

Figure 3. Examples of when End-to-End W/ MLP and Weighted Sampling can outperform Frozen ResNet on Affordance Preservation by correctly predicting a higher hull count in the urinal (top row), but how this can sometimes lead to overpredicting the number hulls for the affordance of the cupcake (bottom row)

Model Architecture	Accuracy	Precision	Recall	Affordance Preserved	Balanced
Baselines					
Always CHull	33.67%	0.337	0.337	33.67%	
Always Co-ACD-32	22.97%	0.223	0.223	**97.59%**	
Frozen ResNet + Cosine Sim	24.16%	0.219	0.208	46.97%	
Trained Models					
Frozen ResNet w/ MLP	49.7%	-	-	62.48%	
End-to-End ResNet w/ MLP (Just Visual Render)	40.0%	0.385	0.245	65.47%	
End-to-End ResNet w/ MLP	46.6%	0.373	0.343	76.80%	
End-to-End ResNet w/ MLP + Weighted Sampling	47.3%	0.435	0.373	83.5%	✓

Table 2. The consolidated metrics of each of our trained models, as compared to the baselines. Note that while the baseline of affordance preservation surpasses our model, overly predicting such a high-fidelity mesh always results in significant computational costs. Our models, especially the End-to-End ResNet w/ MLP and Weighted Sampling, are better able to balance between the two goals.

chitecture, as well as testing with more powerful baselines, such as ResNet50, as our experiments indicated that training ResNet18 did not substantially contribute to accuracy over keeping its weights frozen.

We consider our work a significant step towards developing an end-to-end system capable of automatically generating ACD affordance-preserving meshes across many domains in a way that surpasses manual ACD selection by experts and is more generalizable and scaleable than other machine learning techniques that do not utilize ACD.

8. Contributions & Acknowledgements

Cem Gokmen provided invaluable guidance on designing the system architecture and offered feedback on the implementation and helped troubleshoot technical challenges.

Emily constructed the entire first draft of the paper, con-

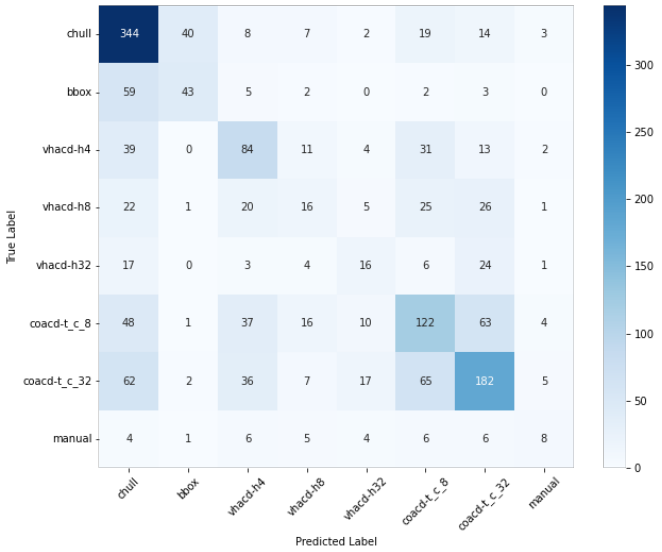

Figure 4. Frozen ResNet w/ MLP classifier. We note the 62 mispredictions of Co-ACD-32 to be Chull

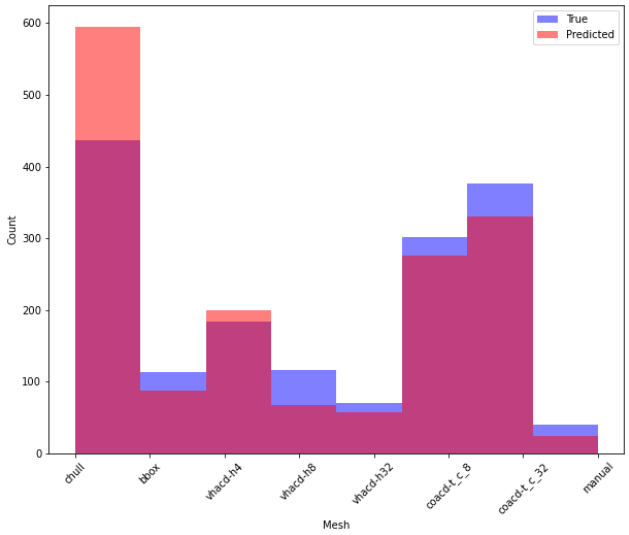

Figure 5. End-to-End ResNet w/ MLP predicted counts across the class distributions

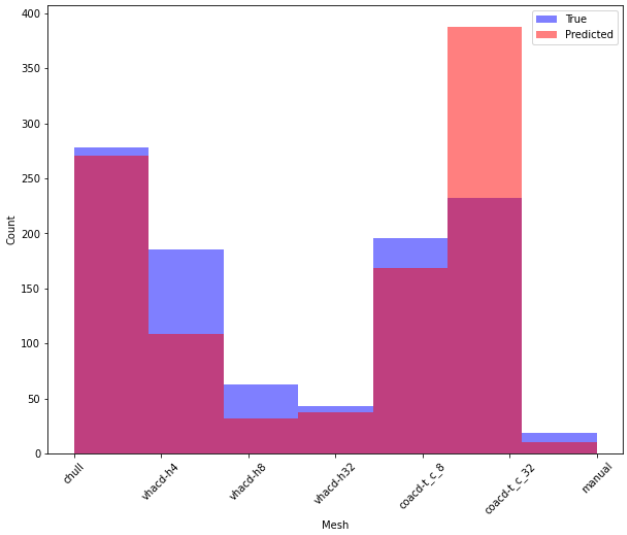

Figure 6. End-to-End ResNet w/ MLP and Weighted Sampling predicted counts across the class distributions. We note the large increase in predicting Co-ACD-32

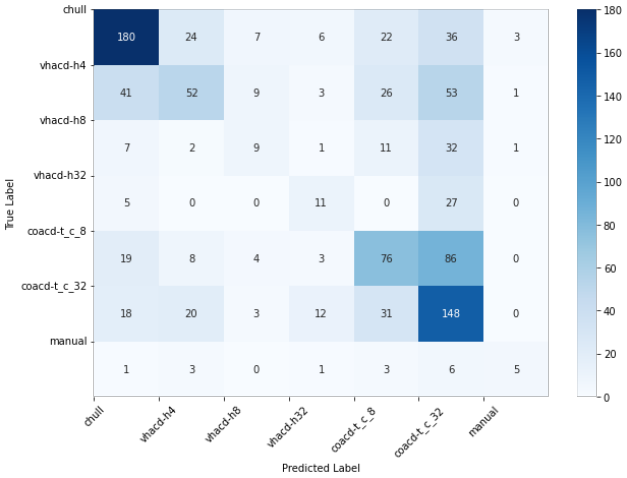

Figure 7. End-to-End ResNet w/ MLP and Weighted Sampling confusion matrix. Note the reduction of misprediction of Co-ACD-32 as CHull down to 18.

ducted qualitative analysis based on the objects' outputs, selected the objects for the Objaverse dataset, and evaluated them.

Tommy helped lead the weekly meetings, implemented the efficient Dataloader used across all models, co-implemented the initial baseline frozen ResNet cosine similarity model, and implemented the frozen ResNet w/ MLP model. He also wrote and edited significantly to the paper across all sections including Abstract, Introduction, Related Work, Data, Methods, Experiments, and Conclusion.

Sean generated the renders used for training and testing, co-implemented the initial baseline frozen ResNet cosine similarity model, End-to-End Training with Visual Renders Model, and weighted sampling model. He also created the graphs and tables in the Experiments and Discussion section, as well as contributed to the Data, Methods, and Experiments portions of the report.

Additionally, we made use of the Stanford Vision Lab's GPU resources. This allowed us to efficiently manage computational tasks and expedite the training and evaluation processes.

References

[1] W. Chen, J. Gao, H. Ling, E. J. Smith, J. Lehtinen, A. Jacobson, and S. Fidler. Learning to predict 3d objects with an

interpolation-based differentiable renderer, 2019.

[2] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023.

[3] M. Deitke, E. VanderBilt, A. Herrasti, L. Weihs, J. Salvador, K. Ehsani, W. Han, E. Kolve, A. Farhadi, A. Kembhavi, and R. Mottaghi. Procthor: Large-scale embodied ai using procedural generation, 2022.

[4] J. Huang, H. Zhang, L. Yi, T. Funkhouser, M. Nießner, and L. Guibas. Texturenet: Consistent local parametrizations for learning from high-resolution signals on meshes, 2019.

[5] E. Kalogerakis, A. Hertzmann, and K. Singh. Learning 3d mesh segmentation and labeling. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, New York, NY, USA, 2010. Association for Computing Machinery.

[6] C. Li, R. Zhang, J. Wong, C. Gokmen, S. Srivastava, R. Martín-Martín, C. Wang, G. Levine, W. Ai, B. Martinez, H. Yin, M. Lingelbach, M. Hwang, A. Hiranaka, S. Garlanka, A. Aydin, S. Lee, J. Sun, M. Anvari, M. Sharma, D. Bansal, S. Hunter, K.-Y. Kim, A. Lou, C. R. Matthews, I. Villa-Renteria, J. H. Tang, C. Tang, F. Xia, Y. Li, S. Savarese, H. Gweon, C. K. Liu, J. Wu, and L. Fei-Fei. Behavior-1k: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation, 2024.

[7] J.-M. Lien and N. M. Amato. Approximate convex decomposition of polyhedra and its applications. *Computer Aided Geometric Design*, 25(7):503–522, 2008. Solid and Physical Modeling.

[8] K. Mamou and F. Ghorbel. A simple and efficient approach for 3d mesh approximate convex decomposition. *International Journal of Virtual Reality*, 8(1):35–41, 2009.

[9] K. Mamou, E. Lengyel, and A. Peters. Volumetric hierarchical approximate convex decomposition. *Game engine gems*, 3:141–158, 2016.

[10] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017.

[11] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE International Conference on Robotics and Automation*, pages 3212–3217, 2009.

[12] X. Wei, M. Liu, Z. Ling, and H. Su. Approximate convex decomposition for 3d meshes with collision-aware concavity and tree search. *ACM Transactions on Graphics*, 41(4):1–18, July 2022.

[13] R. Wu and C. Zheng. Learning to generate 3d shapes from a single example. *ACM Transactions on Graphics*, 41(6):1–19, Nov. 2022.

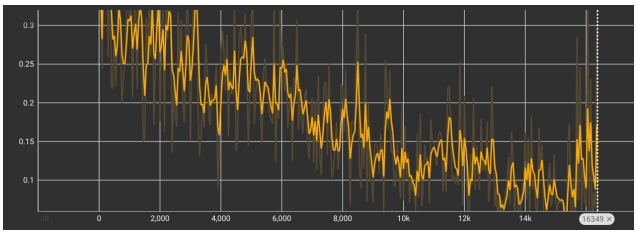

Figure 8. Training loss for Frozen ResNet w/ MLP

9. Appendix