# Program Synthesis for Fine-Grained Semantic Segmentation of 3D Shapes

*A Thesis submitted in partial fulfillment of the requirements for Honors in the Department of Computer Science at Brown University*

Seung Jean Yoo
Advisor: Daniel Ritchie
Reader: Stephen Bach

## 1 Abstract

This thesis investigates how the performance of learning-based approaches to fine-grained 3D shape segmentation can benefit from rules in the form of label functions. We develop a framework for expressing 3D shape part-level knowledge through programmatic rules and show that (1) weak labels from label functions authored by human domain experts improve performance of downstream models on the 3D shape segmentation task when labeled data are limited; (2) large language models (LLMs) can act as automatic generators of label functions defined over 3D shapes; and (3) strategies such as filtering, parameter search, and label aggregation can help de-noise these automatically generated LFs. Our experiments demonstrate that while LLMs possess rich priors about shape structures, the label functions they produce are much noisier than those authored by human domain experts. As a result, while weak labels from automatically generated LFs offer potential for reducing annotation costs, achieving performance competitive with human domain expert-authored rules still requires additional tuning. These findings highlight both the potential and the challenges of using program synthesis to mitigate the data scarcity bottleneck in the task of fine-grained 3D shape semantic segmentation.

## 2 Introduction

Segmenting a 3D shape into distinct, semantically meaningful parts plays a crucial role in many applications in vision, graphics, and robotics. Tasks such as training structure-aware generative shape models [Gao et al., 2019; Mo et al., 2019]; helping autonomous agents interact with their surroundings [Liu et al., 2022]; and shape editing and manipulation [Ganeshan et al., 2024] often require the part regions of the 3D shapes to be fine-grained and hierarchically-organized (e.g. chair backs decompose into back-frame and back-surface sub-components). However, having humans produce a large number of semantic segmentations at this level is time-consuming and labor-intensive.

Recent work [Qi et al., 2017a; Yi et al., 2018] has focused on learning-based approaches to semantically segmenting 3D shapes. While these methods achieve impressive results for coarse segmentations, their performances fall off when fine-grained segmentations are

desired or access to labeled data is limited. PartNet [Mo et al., 2018] and 3DCompat++ [Slim et al., 2023] are two of the only existing large-scale datasets with fine-grained annotations, but even PartNet covers only 24 common everyday categories.

NGSP [Jones et al., 2022] addresses the challenege of limited labels and fine-grained segmentation by learning segmentations using shape *regions*, rather than shape *atoms* (e.g. point cloud points, mesh faces). After an input shape is segmented into shape regions, a neurally-guided search is used to assign semantic labels to the regions. While NGSP outperforms many end-to-end learning approaches in the limited data, fine-grained region setting, it is still far from solving this problem. One approach to improving its performance is collecting cheap but substantial labeled training datasets. For sources of such labels, we can draw inspiration from how human annotators created labels for the PartNet dataset.

The PartNet interface gave humans instructions for labeling in the form of definitions and examples: for example, to label chair back frames, they would be shown an annotated image of a chair and a definition as a "framing part to outline the backbone of a chair back." The annotators would then decide on a region's label using these rules. This rule usage motivates the question: could we formalize these rules into executable label functions that automatically run over unlabeled shape regions, producing a large number of training labels?

Specifically, we define an API in domain-specific language (DSL) for querying geometric properties of a shape region. These API functions are composed to write label functions that decide whether or not to fire, i.e. whether to label a given region with a particular label. We explore how labels generated from running these LFs can be integrated with training the guide network of NGSP in a weak-learning paradigm.

Encoding domain-specific knowledge into handwritten LFs is still expensive. However, modern large language models (LLMs) such as GPT-4 [Brown et al., 2020] contain semantic knowledge about how objects decompose into hierarchical parts. If this knowledge can be leveraged to automatically author LFs, it would mitigate or eliminate the need for human domain experts to hand-write labeling functions for each type of object. Therefore, this thesis also investigates an approach to addressing the label scarcity bottleneck in end-to-end methods for fine-grained 3D semantic segmentation, using labeling functions authored by LLMs for object categories and their semantic parts. As the LLM-authored LFs provide labels that are too noisy to train on directly, we also investigate a series of options to denoise these LFs and provide more useful signals to the downstream model.

We evaluate performance for a fine-grained chair semantic segmentation task, and find that weak labels from LFs written by a human domain expert are very helpful when labeled data are limited. We also find that it is challenging to achieve the same results by leveraging LLM LFs, but additional components of label models, filtering LFs, random parameter search, and jointly training on the weak and ground-truth labels boost downstream model performance.

# 3 Related Work

**End-to-end approaches to semantic segmentation** Most learning-based approaches to 3D shape segmentation reason over shape atoms such as point clouds, faces, and edges, to assign semantic labels. Examples of architectures include PointNet [Qi et al.,

2017a]/PointNet++ [Qi et al., 2017b], which use point clouds, and MeshCNN [Hanocka et al., 2019], which reasons over mesh face and edges. Approaches in this paradigm perform well for coarse segmentations, but their performance decreases when fine-grained segmentations are desired. Rather than attempting to label each atom, which results in a massive search space, NGSP [Jones et al., 2022] operates over pre-segmented regions and uses a neurally-guided probabilistic model to assign fine-grained labels from a semantic grammar. However, its performance also falls off when labeled data is limited. Recent works in the area of zero-shot 3D semantic segmentation have sought to address this limitation by leveraging off-the-shelf 2D image recognition models or pre-trained vision-language models (VLMs) like CLIP ([Abdelreheem et al., 2023], [Decatur et al., 2022], [Zhou et al., 2023]). In contrast, this thesis explores how rule-based programs can provide supervision for end-to-end models, without heavy reliance on pre-trained image models.

**Programmatic weak supervision** Supervised learning methods require massive labeled training datasets; however, manually collecting and labeling these datasets is time-consuming and labor-intensive. To address this data scarcity bottleneck and collect substantial labeled data cheaply, a suite of programmatic weak supervision (PWS) methods have offered an alternative by incorporating sources of noisy labels to create these training sets [Zhang et al., 2022]. Typically, in a PWS system setup, users create heuristics and combine them to create labeling functions that vote on the label for an example, or abstain. These noisy votes are reconciled, and the annotated data is used to train an end model. These methods have seen success in tasks in the textual domain — spam classification and sequence tagging, for instance [Zhang et al., 2021]. In this thesis, we seek to investigate whether these ideas could be applied to the 3D shape semantic segmentation task.

**Automatic generation of label functions and programs** Prior work has explored automating label function creation, but few methods generate LF as programs. For example, Snuba [Varma and Ré, 2018] generates label functions as learned heuristics (e.g. decision stumps, K-Nearest Neighbor models) that can run on unlabeled datasets and output probabilistic labels for downstream model training. Alfred [Yu and Bach, 2023] leverages pre-trained models for prompted weak supervision, where the label functions are defined as *queries* to pre-trained models, and the answers are mapped to label votes or abstentions. In contrast, we generate LF as programs. Related work in the behavior analysis domain, such as AutoSWAP [Tseng et al., 2022], defines labeling functions in a domain-specific language and synthesizes a diverse set of LF programs by formulating the program synthesis task as a graph search problem with neural completions, following the NEAR [Shah et al., 2021] framework. Along with investigating LF program generation in the visual domain, another key difference in our approach is the use of LLMs to generate candidates LFs, aligned with recent lines of work in LLM program synthesis: for example, in the context of geometric reasoning problems and discovering mathematical concepts [Trinh et al., 2024; Romera-Paredes et al., 2023].

# 4 Background

In this section, we formalize the 3D fine-grained semantic segmentation task. The objective is to segment an input shape $S$ with semantic labels from a grammar $L$. To do so, we assume that the 3D shape input to be labeled is already decomposed into different regions $R$, such that $S = \{R_i\}$. Then, we wish to find the correct labeling assignment

to each region $A = \{a_i\}$ using the labels in our grammar. Figure 1 shows example 3D chair models with some regions of these part labels highlighted. Specifically, we use the following grammar for chairs, where "chair" is the starting node of the grammar and the terminals are the bolded terms.

| | |
|---|---|
| chair | $\rightarrow$ back; **arm**; seat; base; |
| back | $\rightarrow$ **surface**; **frame** |
| seat | $\rightarrow$ **surface**; **frame** |
| base | $\rightarrow$ **foot**; **runner**; **bar-stretcher**; **leg** |

The model $q$ that learns to do this segmentation learns to classify each region as belonging to a label: $q(a|S)$. Each shape $S$ produces $|S|$ training examples, one for each region. Therefore, in the weak supervision framework, each label function takes as input an unlabeled region $R_i$ and either predicts a label from $L$, or abstains – this information is then used to learn $q(a|S)$.

Figure 1: Examples of 3D chair model regions: back frame, seat surface, and arms.

# 5 Method

Figure 2 shows an overview of the approach. The system consists of three main stages. In the first stage, a human domain expert authors the label function for each part, or an LLM is prompted to generate labeling functions for each label using the provided API functions in the DSL we have developed. In the second stage, the label functions are run over unlabeled, segmented shapes in the dataset and every fire is recorded as a (region, weak label) pair. In the third stage, these resulting pairs are used to train a downstream model (NGSP's guide network) for a 3D semantic segmentation task. There are intermediate stages for denoising the LLM-authored LFs, and aggregating possibly conflicting votes, described more in detail in the following sections.

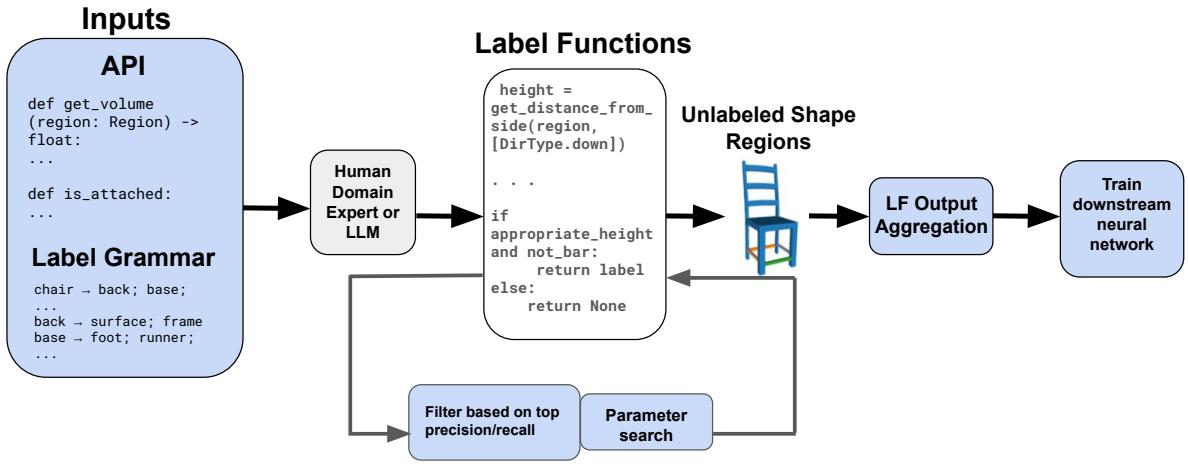



Figure 2: Overview of the system. For each label in our label grammar for a 3D object, either a human domain expert authors a label function, or an LLM is prompted for a corresponding label function using our API for functions that reason over geometric properties of a shape region. Then, the label functions are run over unlabeled shape decompositions. The label function votes are aggregated and the resulting (shape region, votes) are used to train NGSP's guide net for a 3D semantic segmentation task.

## 5.1 Label Functions

zA 3D segmentation task for an object with $k$ part labels can be framed as a $k$-way classification problem. Thus, one set of labeling function contains $k$ LFs, one that votes for a particular label. Specifically, each labeling function takes as input a given region from a shape, and uses a set of criteria to fire its corresponding label for the region, or abstain. For example, a labeling function for a chair leg might check that the region is an elongated bar and located under a chair seat region, and abstain from voting if these conditions are not met. As these criteria often depend on geometric properties of the shape regions, we design an API of functions for geometric reasoning. These functions can reason over individual regions (e.g. a part's distance from the bottom of the object's bounding box) or in relation to other regions (e.g. whether the part is connected to a different part in the object). The label functions can call these helper functions and aggregate the conditions using simple boolean and/or logic to vote on a shape region. The following is an example LF for a chair arm, which returns a label if conditions are met, or abstains by returning None. Please see the Appendix for more details on the API and label functions.

```
def label_function(region, label = "chair/chair_arm"):
    # A chair arm oriented front-back along its principal axis
    elongated_and_depth_aligned = is_oriented_bar(region, dim_ratio
    =2.0, fit_error_threshold=0.1, orientation=DirType.front)

    # Should not touch the ground
    not_on_ground = get_distance_from_side(region, [DirType.down]) >
    0.1

```

```
    # Close to the sides in terms of width (left/right)
    close_to_sides = get_distance_to_center(region, [DimType.width]) >
0.2

    # The region should be closer to the upper part of the chair
    above_seat = get_position(region)[1] > 0.2

    if elongated_and_depth_aligned and close_to_sides and above_seat:
        # Fire the label
        return label
    else:
        # Abstain
        return None
```

## 5.2 Models

The 3D semantic segmentation model in this project is the Guide network from NGSP. This model takes a shape region as input, and predicts a semantic label from the grammar to the region.

As label functions encode heuristics for producing a label for a given shape part, they are often noisy and conflict with each other. For example, a given shape region may have both LFs for a "chair leg" and "chair foot" fire; or, it may have one version of a "chair leg" LF fire, but not a second LF version, depending on the constraints they check for. In the PWS framework, *label models* are used to aggregate and resolve these conflicting votes into probabilistic labels that are useful for training downstream models.

While various label models have been proposed, we explore the most lightweight techniques. The majority voting (MV) approach simply uses the consensus from multiple LF sets to obtain more reliable labels [Zhang et al., 2022]. The second approach, like in Snorkel [Ratner et al., 2017] makes the naive Bayes assumption that the labeling function ouputs are conditionally independent given the true labels, and learns a generative model from the agreements and disagreements of the labeling function outputs. The posterior distribution over the true labels is used in the (shape region, label distribution) training input to the guide network.

## 5.3 LLM-Authored LF Denoising Strategies

LLMs are trained on data that captures rich semantic knowledge about how shapes decompose into parts. However, the resulting LFs often misfire or under-fire, resulting in too few useful weak labels to use as training data. That is, the DSL helps express shape-part knowledge, but it does not guarantee correctness of logic; for example, the LLM can still pick bad thresholds, specify incorrect geometric features, or over- or under-constrain the label functions' firing conditions.

Therefore, we use a small development set, consisting of 40 chairs and their ground-truth labels. We investigate the gains from sampling many LFs from the LLM, and then filtering out LFs with low precision score on the development set, either by choosing a target threshold or taking the top five best performing LFs.

We also conduct a parameter search over the label functions to choose a configuration that yields the highest F1 score on this development set. As an exhaustive grid search takes several days depending on the number of parameters and the spacing of the grid, we choose to conduct a random parameter search for our experiments. In order to isolate

the performance of each label function during parameter search, we assign ground-truth labels to all non-target regions. Specifically, when tuning a label function for a particular part (e.g., arm), we assume that every other region (e.g., seat, back, legs) is correctly labeled according to the ground truth so that API functions that depend on part relations are not impacted by neighboring label accuracies. This ensures that the parameter search focuses solely on the current label function's performance.

After obtaining this set of LFs that perform best on the development set, we run them over the unlabeled shape regions to obtain the (region, weak label) pairs, aggregate using the label models in the previous section, and train the downstream model.

# 6 Results

In this section, we detail the results of integrating weak labels from human-authored and LLM-authored LFs with the NGSP guide network, using different training and denoising strategies. Our experiments use 3D CAD chair shapes from PartNet ([Mo et al., 2018]). PartNet contains approximately 8000 chair shapes; we construct a validation and test set of 200 shapes. We reserve another 200 to use as the "supervised" set of shapes with GT labels for different experiments. Of the remaining shapes, we filter out duplicate shapes that are under different shape IDs but represent the same chair model and label data due to mislabels in PartNet, and treat 3000 of these shapes as "unlabeled" shapes that are used as inputs to the LFs.

We evaluate 3D semantic segmentation performance with the mean Intersection over Union (mIoU) metric, where higher is better. This metric is the average per-node IoU for each node in the hierarchical label grammar, over all of the shapes in the set ([Mo et al., 2018]).

## 6.1 Learning from Human Domain Expert-Authored LF Weak Labels

Before considering the weak labels' impact on the guide network performance, we can consider the performance of the label functions themselves. Table 1 shows the precision and recall performance of running each part's label function on the set of unlabeled chair regions. The human domain expert-authored label functions were able to generate weak labels that covered a majority of the regions (4 out of 9 LFs achieve a recall of around 0.5 or above), and retain high precision (only 2 LFs have a precision lower than 0.5).

| Part | Precision | Recall |
|---|---|---|
| Arm | 0.87 | 0.29 |
| Back frame | 0.68 | 0.25 |
| Back surface | 0.69 | 0.71 |
| Bar stretcher | 0.49 | 0.49 |
| Leg | 0.92 | 0.72 |
| Runner | 0.56 | 0.67 |
| Seat frame | 0.53 | 0.20 |
| Seat surface | 0.65 | 0.18 |
| Foot | 0.48 | 0.36 |

Table 1: Precision and recall for human-authored LFs across 3000 unlabeled chairs' parts.

Figure 3 shows how a model, jointly trained on weak labels and a small number of ground-truth labels (from 40 shapes), benefits from these generated weak labels. We plot the mIoU on the Y-axis, and the number of unlabeled shapes used to generate weak labels on the X-axis. We visualize the baseline of training the model on 40 shapes' ground-truth labels in the red plot, and the bound of training the model on 100 shapes' ground-truth labels in the blue plot. We can see that weak labels are helpful in this limited labeled data setting: as weak-labels generated from 500 shapes, and then 1000 shapes, are added to the model's training data, it outperforms the setting where it only has access to 100 shapes' ground-truth labels. That is, by automatically running these rules on a large number of shapes, we get better downstream performance as opposed to having human annotators manually label many shape regions.

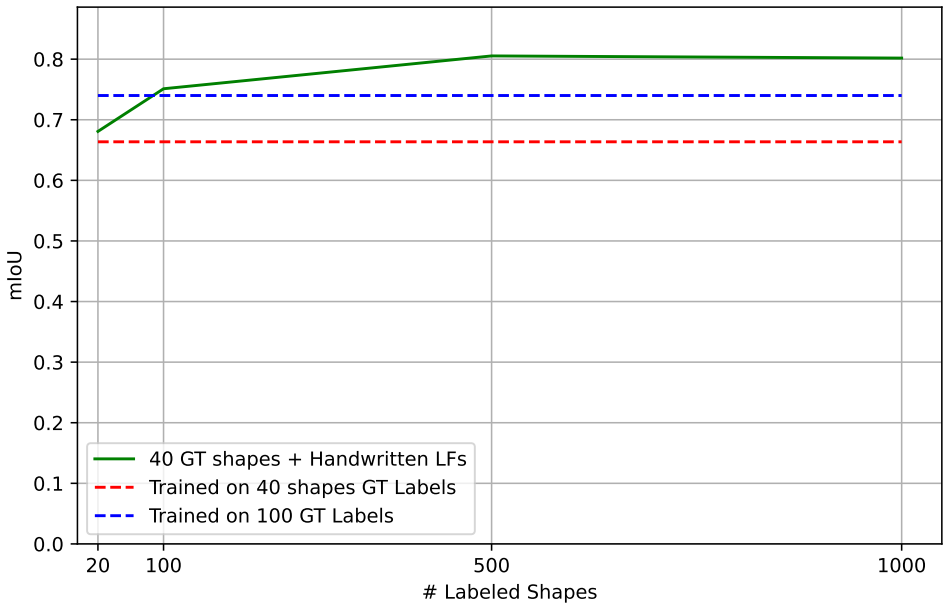



Figure 3: NGSP Guide Network semantic segmentation performance measured in mIoU (y-axis) as the number of weak labels of unlabeled shapes, obtained from human-authored LFs (in addition to 40 shapes' GT labels) varies (x-axis). Compared to training only on 40 and 100 chairs' GT labels.

## 6.2 Learning from LLM-Authored LF Weak Labels

We present the precision and recall results of generating weak labels from LLM-authored label functions, averaged over ten sets of LFs (1 LF for each of the 9 part labels per set, total of 90 LFs), in Table 2. Similar to before, precision and recall of the label functions are organized by part label.

We see that the signals from automatically-generated LFs are much noisier compared to those authored by human domain experts. Although some part LFs achieve high precision, such as "chair arm" or "chair leg," 6 out of 9 labels have less than 0.5 precision. Compared to the handwritten rules, the automatically-generated LFs also have lower recall. In particular, the LLM is able to generate LFs that perform well on relatively simple parts such as the leg, but its performance falls for parts that are harder to distinguish from surrounding parts, like bar stretchers or runners. This suggests that the although the LFs have generally reasonable semantic constraints, many of them may be over-constrained and some of the constraints may not be well-aligned with the actual shape regions, leading to misfires.

| **Part** | **Precision** | **Recall** |
|---|---|---|
| Arm | 0.88 | 0.08 |
| Back frame | 0.18 | 0.19 |
| Back surface | 0.15 | 0.12 |
| Bar stretcher | 0.02 | 0.12 |
| Leg | 0.90 | 0.45 |
| Runner | 0.25 | 0.37 |
| Seat frame | 0.15 | 0.27 |
| Seat surface | 0.55 | 0.33 |
| Foot | 0.13 | 0.58 |

Table 2: Precision and recall for LLM-authored LFs across 3000 unlabeled chairs' parts.

Figure 4 shows how the model, trained only on weak labels from the LLM-authored LFs, performs as more weak labels are added. As before, the mIoU is plotted on the Y-axis, and the X-axis shows how many shapes' weak labels are used as training data. As the comparison baseline, we train the model on 40 shapes and their GT labels (in green). We see the noisy signals, indicated by the precision and recall metrics in the previous paragraph, translate to the downstream model performance. The model, trained on 3000 shapes and their weak labels, underperforms compared to the baseline. This suggests that while LLM-authored label functions can provide some gains in settings where no ground-truth annotations are available, their efficiency is limited. In particular, that weak labels cannot match the performance achievable with a relatively small amount of manual annotation, shows that there are some performance improvements left to be desired – especially when considering the "upper bound" of performance achievable with the handwritten label functions.

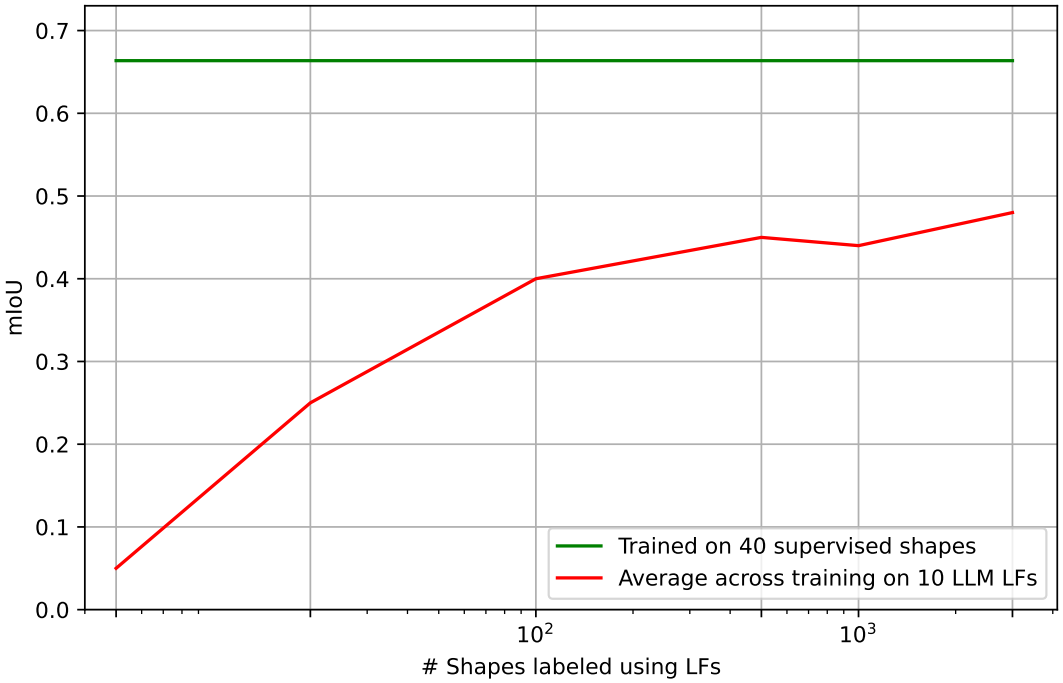
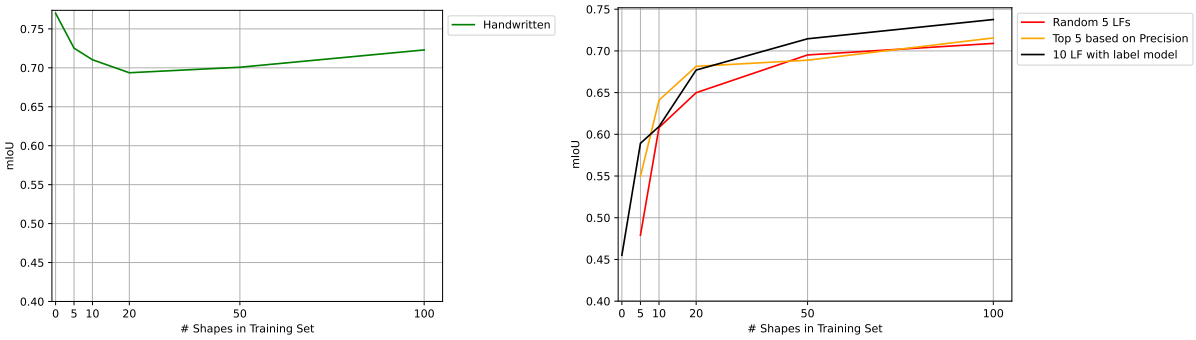



Figure 4: NGSP Guide Network semantic segmentation performance measured in mIoU (y-axis) as the number of weak labels of unlabeled shapes, obtained from LLM-authored LFs (red), varies (x-axis). Compared to training on 40 chairs' GT labels (green).

## 6.3 Jointly Training on Weak and Ground Truth Labels

To implement strategies for de-noising LLM-authored LFs, we assume access to a small set of shapes and their ground truth labels, referred to as the "development set". There are different options for integrating these GT labels into model training: we first explore (1) jointly training on the GT and weak labels, and (2) pre-training on the weak labels and fine-tuning using the GT labels.

Figure 5a shows the model performance when pre-trained on handwritten LF weak labels, and fine-tuned on the GT labels. The Y-axis shows mIoU, and the X-axis shows the number of shapes used to fine-tune the model. Interestingly, we see that although the model starts off with 0.8 mIoU after training on 3000 shapes and their weak labels, the plot shows a trend where as the mIoU decreases as more GT labels are added. The information that the model learns from pre-training is undone by the fine-tuning process.




(a) Pre-trained on weak labels from *hand-written* rules.

(b) Pre-trained on weak labels from different subsets of *LLM-generated* rules.

Figure 5: NGSP Guide Network semantic segmentation performance measured in mIoU (y-axis), as the guide net, pre-trained on weak labels, is fine-tuned on increasing numbers of supervised set shapes (x-axis).

We see a similar trend with the LLM-generated rules in Figure 5b, which plots the mIoU against the number of shapes used for fine-tuning. We visualize the model performance of training on 5 LFs (out of 10) with the best precision on the development set (orange), a randomly-selected set of 5 LFs (red), and using all 10 LFs (black), all aggregated with the label model. There are some differences in the model performance at the end of pre-training, as seen in the left-hand side of the plot; however, at the end of fine-tuning, the model performances all middle around the same value, and at the same value that Figure 5a ends at. These results strongly suggest that when weak labels and ground-truth are labels are integrated by pre-training and fine-tuning, the model undergoes catastrophic forgetting [Goodfellow et al., 2015]. Informed by these results, we jointly train the model on weak labels and GT labels in the following experiments.

## 6.4 Learning from De-Noised LLM-Authored LF Weak Labels

To mitigate the noisy training signals from the LLM-authored LFs, we investigate several strategies for improving the performance of LFs and the downstream model. In Figure 6, we visualize the downstream model performance of jointly-training on weak and GT labels, after resolving conflicting weak labels through majority voting (in orange) and using a Naive Bayes label model (in blue).

First, in this joint training setup, we see that as more weak labels are added, there is a trend of the mIoU decreasing as the noise overpowers any useful information from the GT labels. Therefore, the model underperforms when compared to the baseline of just using the GT labels of 40 shapes. Despite this trend, there is an overall performance boost from resolving conflicting votes among different LF versions for each part label probabilistically, using the Naive Bayes label model. This suggests that LLMs can act as easily-available sources to automatically generate LFs, but some LF candidates are more accurate than others; therefore, using a label model essentially weighs these LFs accordingly. Further, this suggests that training the model on probabilistic labels as opposed to hard labels allows the guide network to learn from signals that account for this uncertainty in noisy LF samples, rather than committing to a single, potentially erroneous label.

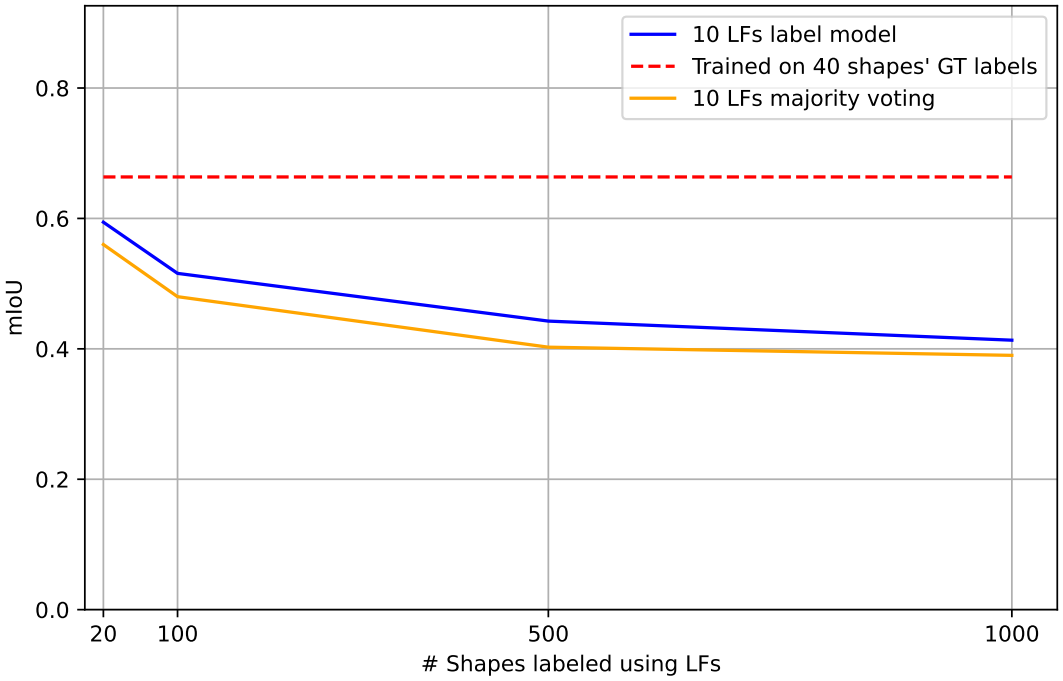



Figure 6: NGSP Guide Network semantic segmentation performance measured in mIoU (y-axis) as the number of weak labels of unlabeled shapes varies (x-axis), with conflict resolution using majority voting (orange) and the naive Bayes label model (blue).

We further visualize the impact of adding the filtering process to the system in Figure 7. We compare the result of using a label model without the filtering process (blue) and choosing 5 sets of LFs for each part label with the highest precision (purple). We see an additional boost in overall mIoU performance when filtering out noisy LFs, further supporting that when the LLM is sampled multiple times, it generates a mixture of more useful and more noisy LFs, and these de-noising strategies can help mitigate the noise. To further address the gap between the best-performing model trained on weak labels from LLM-authored LFs and the handwritten LF weak labels, we investigate random parameter research.

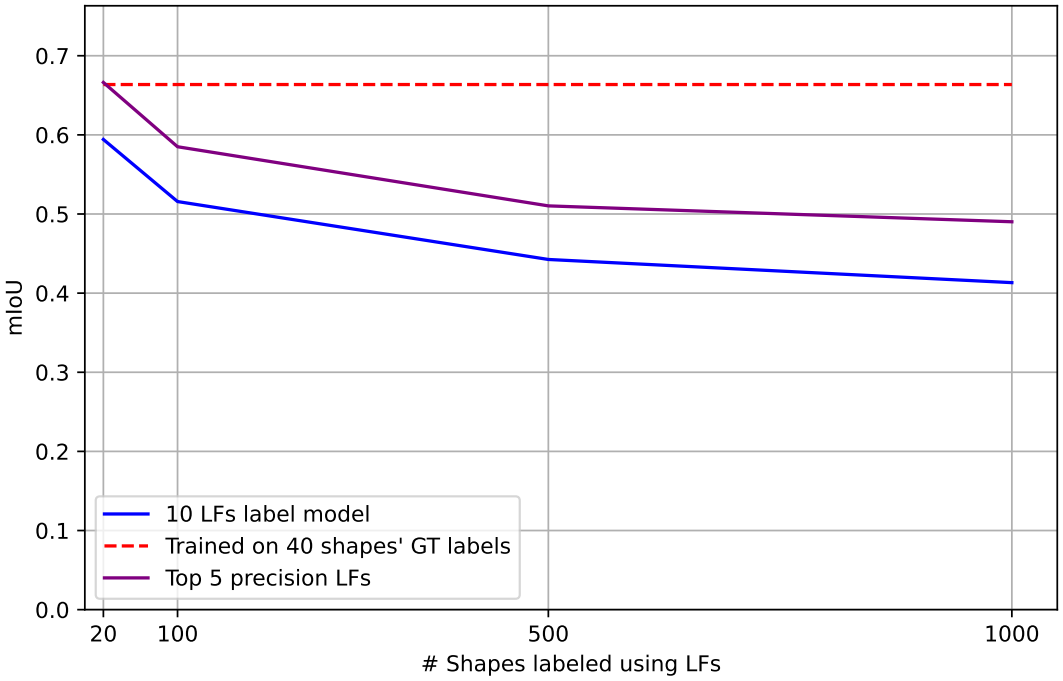



Figure 7: NGSP Guide Network semantic segmentation performance measured in mIoU (y-axis) as the number of weak labels of unlabeled shapes varies (x-axis), without precision filtering (blue) and with precision filtering (purple).

Table 3 shows the F1 score before and after running a random parameter search over an LLM-generated label function, using the development set. The "Handwritten" column shows the F1 score of the human-authored rules on the same set of shapes, and the last column shows the change in F1 from running the random search. Though the delta from random search is not significantly high across the parts, ranging from as low as 0 and 0.04 to a boost of 0.375, some of these improvements are large enough to result in LLM-generated LFs that achieve higher F1 than the handwritten rules on this set of shapes (bar stretcher, leg, foot). However, parts such as the seat frame, where there was no gain from the parameter search, suggest that a more sophisticated strategy than random search may be required for larger improvements.

| **Part** | **Handwritten** | **with LLM constants (A)** | **After random search (B)** | **Δ(B - A)** |
|---|---|---|---|---|
| Arm | 0.67 | 0.00 | 0.375 | +0.375 |
| Back frame | 0.60 | 0.32 | 0.36 | +0.04 |
| Back surface | 0.77 | 0.19 | 0.55 | +0.36 |
| Bar stretcher | 0.78 | 0.53 | 0.82 | +0.29 |
| Leg | 0.78 | 0.53 | 0.82 | +0.05 |
| Runner | 0.94 | 0.89 | 0.94 | +0.20 |
| Seat frame | 1.00 | 0.80 | 1.00 | +0.17 |
| Seat surface | 0.38 | 0.42 | 0.59 | 0.00 |
| Foot | 0.52 | 0.76 | 0.87 | +0.11 |

Table 3: Comparison of F1 score of running handwritten and LLM-authored LFs on the dev set, before and after random parameter search, organized by part label.

Importantly, we also find that despite the improved performance of parameter optimization on the development set, this trend does not fully transfer to an unsupervised

setting. As detailed in the methods section, this parameter search process assumes access to other GT labels to isolate the performance of the LF in question. However, when these labels are replaced with noisy predicted labels—as would be the case in practice—the previously optimized configurations often lead to a drop in precision and recall for some part labels. In Table 4, we show the change in F1 when running the LF with the "optimal" configurations on the same development set, with and without GT access assumption. 5 out of 9 part labels show a drop in F1 when this GT access goes away, suggesting that these parameters are not truly optimal, as they are overfit to an overly optimistic assumption of neighboring regions' label accuracy.

This outcome is also seen in the unsupervised shapes set, where Table 5 shows that the gain from parameter search compared to LLM-given constants is unclear; few parts (back surface, seat surface) see improvements, but several other parts' F1 scores for the unsupervised shapes drop with the "optimized" constants from parameter search (runner, stretcher).

| **Part** | **GT Neighbor Labels (C)** | **Predicted Neighbor Labels (D)** | **Δ (D - C)** |
|---|---|---|---|
| Arm | 0.49 | 0.38 | -0.11 |
| Back frame | 0.44 | 0.35 | -0.09 |
| Back surface | 0.65 | 0.59 | -0.06 |
| Bar stretcher | 0.43 | 0.43 | 0.00 |
| Leg | 0.91 | 0.91 | 0.00 |
| Runner | 0.83 | 0.77 | -0.06 |
| Seat frame | 0.42 | 0.48 | +0.06 |
| Seat surface | 0.36 | 0.37 | +0.01 |
| Foot | 0.74 | 0.71 | -0.03 |

Table 4: Average F1 scores of top-5 parameter-optimized LFs on the dev set, comparing performance when ground-truth neighbor labels are given vs. using LF predictions to assign neighbor labels.

| **Part** | **LLM Constants (E)** | **Param Opt (F)** | **Δ (F - E)** |
|---|---|---|---|
| Arm | 0.18 | 0.15 | -0.03 |
| Back Frame | 0.24 | 0.24 | 0.00 |
| Back Surface | 0.32 | 0.56 | +0.24 |
| Stretcher | 0.15 | 0.000002 | -0.15 |
| Leg | 0.63 | 0.74 | +0.11 |
| Runner | 0.45 | 0.11 | -0.34 |
| Seat Frame | 0.33 | 0.29 | -0.04 |
| Seat Surface | 0.45 | 0.59 | +0.14 |
| Foot | 0.23 | 0.16 | -0.07 |

Table 5: F1 scores of top-5 LFs applied to the unsupervised dataset, after label model aggregation, comparing original LLM-constant LFs to configurations found through parameter search.

This degradation in weak label quality also impacts downstream model performance. In Figure 8, we visualize the mIoU after training on weak labels from the 5 highest-precision LFs with LLM-given constants (purple) and after parameter search (cyan). We

observe that using the resulting configurations in a more realistic setting leads to slightly lower performance than using the original parameters when few shapes are labeled, and slightly improved performance when more shapes are labeled. These results suggest that future work in parameter tuning and LF denoising should focus on strategies that are robust to noisy LFs for neighboring labels, and avoid overfitting to artificially clean settings.

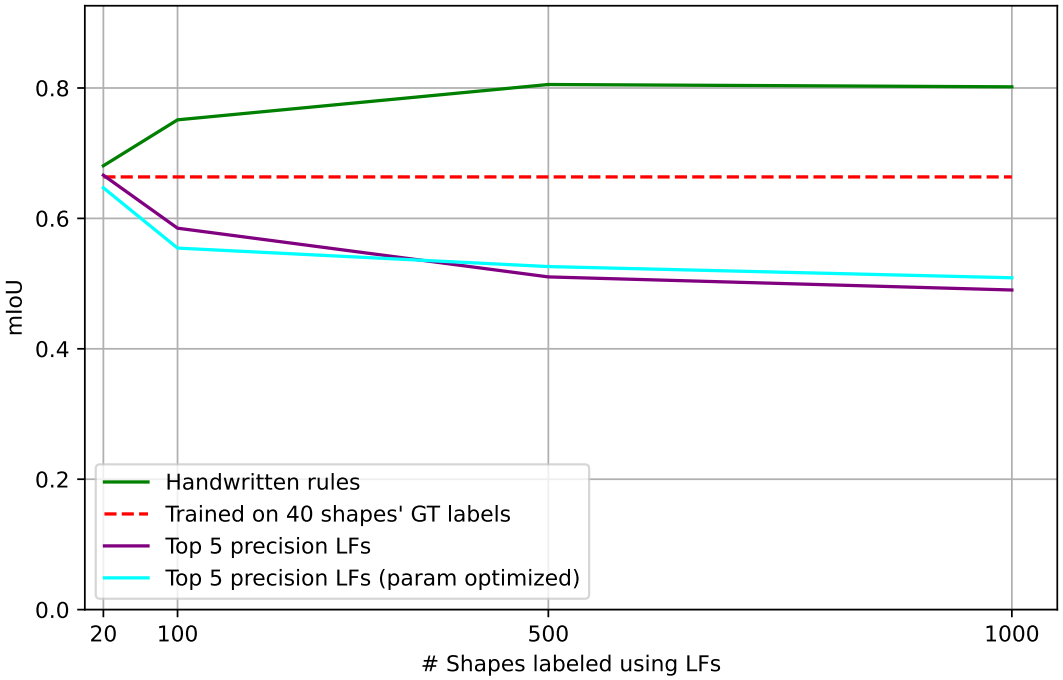



Figure 8: NGSP Guide Network semantic segmentation performance measured in mIoU (y-axis), as the guide net, is jointly trained on GT labels and increasing numbers of weak labels (x-axis) from the top 5 precision LFs before and after parameter search (purple and cyan, respectively). Compared to handwritten LF weak labels (green).

# 7 Limitations and Future Work

We show that label functions authored by human domain experts provide useful information when labeled data is limited, but we have found that it is challenging to achieve similar results by relying solely on LLM-authored LFs. While the strategies explored above (random parameter search, filtering, naive Bayes label model) provide some boost in performance, there is still a large gap between handwritten and LLM LFs, indicated by the decreasing mIoU trend as more weak labels are added. To address this limitation, directions for future work include:

**Parameter search strategies** Parameter optimization through exhaustive grid search is time-consuming; depending on the number of parameters and grid spacing, it can take up to several days to run this search per label function. As a result, we utilized random search in our experiments. However, the results suggest that more principled parameter optimization methods could be employed to tune the thresholds of LLM-authored LFs. For example, future work could implement parameter search using covariance matrix adaptation (CMA-ES) [Hansen and Ostermeier, 2001], which maintains candidate solutions sampled according to a multivariate normal distribution, and iteratively updates

the distribution based on the best-performing samples to guide the search toward more promising regions of the parameter space (in this case, measured by F1 score on the development set). CMA-ES is well-suited for our setup because it is a derivative-free optimization method. Parameter tuning strategies that are more robust to noisy predicted neighboring labels could also be explored, such as tuning each LF using only the predicted neighboring labels, and repeatedly sweeping through the LFs until the performance saturates or converges.

**LLM self-repair** While there are gains from conducting parameter optimization, there are some LF samples where the gap between LLM vs. handwritten rules cannot be closed by simply tuning constants. For instance, for a "chair arm" LF that specifies that arm regions should be oriented left-right (as opposed to forward-backward or vertical), larger changes in LF's constraints are required. Incorporating mechanisms for the LLM to iteratively revise or self-correct [Pan et al., 2023] its output by providing it instances of false positive and false negative regions, could improve LF performance.

**Including pre-trained VLMs in API** While LLMs can author LFs using general semantic knowledge about shapes, the noisy signals in its LFs indicate a need to further ground LLMs in the visual world. In a similar spirit to ViperGPT [Surís et al., 2023], which queries VLM modules to answer questions about images, future work could investigate integrating VLMs into the API. These queries would allow the LLM to reason about the shape regions beyond geometric attributes like orientation or volume. Instead, it could ask about higher-level properties, such as whether a region supports sitting, allows rocking, and so on.

# 8 Conclusion

We propose a framework for formalizing rules used for labeling fine-grained semantic segmentation data into executable label functions, automatically producing a large number of training data for end-to-end models that learn to semantically segment 3D shapes. Our results show that useful label functions can be defined over 3D shapes, and weak labels from human domain expert-authored LFs are useful when manually-labeled data are limited. There is potential to further alleviate reliance on human labor by leveraging LLMs, which have reasonable high-level priors for how shapes decompose into parts and can generate a large number of LFs efficiently. However, as automatically-generated LFs can provide labels that are too noisy to train on directly, we also present strategies to denoise LLM-authored LFs and produce more reliable supervision for downstream model training.

# 9 Acknowledgements

This thesis is part of a larger project being explored in the Brown Visual Computing group with Vivian Lu, advised by Daniel Ritchie and Kenny Jones.

I would like to thank my advisor Daniel Ritchie for introducing me to the world of visual computing, and for his kind, thoughtful mentorship throughout my time in his group. I'm also grateful to Kenny Jones; this thesis would not be possible without his guidance and insights. I would also like to thank my reader Stephen Bach for his willingness to help and for his feedback. Lastly, thanks to my family and friends at Brown and elsewhere.

# References


Ahmed Abdelreheem, Ivan Skorokhodov, Maks Ovsjanikov, and Peter Wonka. Satr: Zero-shot semantic segmentation of 3d shapes, 2023. URL https://arxiv.org/abs/2304.04909.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Dale Decatur, Itai Lang, and Rana Hanocka. 3d highlighter: Localizing regions on 3d shapes via text descriptions, 2022. URL https://arxiv.org/abs/2212.11263.

Aditya Ganeshan, Ryan Y. Huang, Xianghao Xu, R. Kenny Jones, and Daniel Ritchie. Parsel: Parameterized shape editing with language, 2024. URL https://arxiv.org/abs/2405.20319.

Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao Zhang. Sdm-net: Deep generative network for structured deformable mesh, 2019. URL https://arxiv.org/abs/1908.04520.

Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks, 2015. URL https://arxiv.org/abs/1312.6211.

Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: a network with an edge. *ACM Transactions on Graphics*, 38 (4):1–12, July 2019. ISSN 1557-7368. doi: 10.1145/3306346.3322959. URL http://dx.doi.org/10.1145/3306346.3322959.

Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001. doi: 10.1162/106365601750190398.

R. Kenny Jones, Aalia Habib, Rana Hanocka, and Daniel Ritchie. The neurally-guided shape parser: Grammar-based labeling of 3d shape regions with approximate inference, 2022. URL https://arxiv.org/abs/2106.12026.

Minghua Liu, Xuanlin Li, Zhan Ling, Yangyan Li, and Hao Su. Frame mining: a free lunch for learning robotic manipulation from 3d point clouds, 2022. URL https://arxiv.org/abs/2210.07442.

Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding, 2018. URL https://arxiv.org/abs/1812.02713.

Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas J. Guibas. Structurenet: Hierarchical graph networks for 3d shape generation, 2019. URL https://arxiv.org/abs/1908.00575.

Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies, 2023. URL https://arxiv.org/abs/2308.03188.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017a. URL https://arxiv.org/abs/1612.00593.

Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space, 2017b. URL https://arxiv.org/abs/1706.02413.

Alexander Ratner, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment*, 11(3):269–282, November 2017. ISSN 2150-8097. doi: 10.14778/3157794.3157797. URL http://dx.doi.org/10.14778/3157794.3157797.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 2023. doi: 10.1038/s41586-023-06924-6.

Ameesh Shah, Eric Zhan, Jennifer J. Sun, Abhinav Verma, Yisong Yue, and Swarat Chaudhuri. Learning differentiable programs with admissible neural heuristics, 2021. URL https://arxiv.org/abs/2007.12101.

Habib Slim, Xiang Li, Mahmoud Ahmed Yuchen Li, Mohamed Ayman, Ujjwal Upadhyay Ahmed Abdelreheem, Suhail Pothigara Arpit Prajapati, Peter Wonka, and Mohamed Elhoseiny. 3DCoMPaT++: An improved large-scale 3d vision dataset for compositional recognition. In *arXiv*, 2023.

Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning, 2023. URL https://arxiv.org/abs/2303.08128.

Trieu Trinh, Yuhuai Wu, Quoc Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 2024. doi: 10.1038/s41586-023-06747-5.

Albert Tseng, Jennifer J. Sun, and Yisong Yue. Automatic synthesis of diverse weak supervision sources for behavior analysis, 2022. URL https://arxiv.org/abs/2111.15186.

Paroma Varma and Christopher Ré. Snuba: automating weak supervision to label training data. *Proc. VLDB Endow.*, 12(3):223–236, November 2018. ISSN 2150-8097. doi: 10.14778/3291264.3291268. URL https://doi.org/10.14778/3291264.3291268.

Li Yi, Wang Zhao, He Wang, Minhyuk Sung, and Leonidas Guibas. Gspn: Generative shape proposal network for 3d instance segmentation in point cloud, 2018. URL https://arxiv.org/abs/1812.03320.

Peilin Yu and Stephen H. Bach. Alfred: A system for prompted weak supervision, 2023. URL https://arxiv.org/abs/2305.18623.

Jieyu Zhang, Yue Yu, Yinghao Li, Yujing Wang, Yaming Yang, Mao Yang, and Alexander Ratner. Wrench: A comprehensive benchmark for weak supervision, 2021. URL https://arxiv.org/abs/2109.11377.

Jieyu Zhang, Cheng-Yu Hsieh, Yue Yu, Chao Zhang, and Alexander Ratner. A survey on programmatic weak supervision, 2022. URL https://arxiv.org/abs/2202.05433.

Yuchen Zhou, Jiayuan Gu, Xuanlin Li, Minghua Liu, Yunhao Fang, and Hao Su. Partslip++: Enhancing low-shot 3d part segmentation via multi-view instance segmentation and maximum likelihood estimation, 2023. URL https://arxiv.org/abs/2312.03015.

# Appendix

## A.1 DSL for Geometric Reasoning

As each label function must be able to reason over 3D regions, we develop an DSL for functions that can be combined to express the constraints for each part label. The API allows specification of (1) global geometric properties, such as axis-alignment and volume; (2) relational geometric peroperties, such as attachment; and (3) rule utils, such as finding all regions of a certain label. Example functions and their docstrings are given below.

```
def is_axis_aligned(shape, region, threshold):
    """
    Returns the probability that the given region is axis-aligned,
    according to its minimum oriented bounding box, where threshold is a
    parameter that specifies the allowable cosine error against the
    axis-aligned directions.
    """

def get_volume(shape, region):
    """
    Calculates the volume of the region.
    """

def get_attached_regions(shape, key_region, query_regions, threshold):
    """
    Returns list of every element in query_regions that physically
    attaches to some sub-region of the key_region. threshold is a
    parameter that specifies the distance at which two regions are
    considered attached.
    """

def find_region_by_label(shape, shape, label_assignment, query_label):
    """
```

```
    Consumes a shape, where each region of the shape has a corresponding label from the label_assignment. Filters for sub-regions that match a query_label, and returns this set of sub-regions as a list. Matching implies the labels match, or that label in the label_assignment is a child label of the query label.
    """
```

## A.2 Label Functions

Examples of more LLM-generated and human-authored rules are given below. Each function consumes a segmented mesh region and fires a label, or returns None if the conditions are not met. Note that the human-authored were implemented using an earlier version of the API. While the syntax differs slightly from the final API described in Appendix A.1, the core semantics remain the same.

**Example LLM-Authored Rules**

```
def arm(region, label="chair/chair_arm"):
    # A chair arm is typically an elongated bar oriented along the depth (front-back) direction
    aligned_bar = is_oriented_bar(region, orientation=DirType.front)

    # The arm should not touch the ground
    not_on_ground = get_distance_from_side(region, [DirType.down]) > 0.1

    # The arm should be relatively close to the sides of the chair along the width direction
    close_to_sides = get_distance_to_center(region, [DimType.width]) > 0.2

    # The arm should be positioned in the upper half of the chair (above the seat)
    above_mid_height = get_position(region)[1] > 0.5

    if aligned_bar and not_on_ground and close_to_sides and above_mid_height:
        # fire the label
        return label
    else:
        # don't fire
        return None

def bframe(region, label = "chair/chair_back/back frame"):
    # The back frame is usually upright and well fit by a vertical bar
    upright_bar = is_oriented_bar(region, orientation=DirType.up)

    # The back frame should be higher compared to the seat and not touching the ground
    position = get_position(region)
    height_above_ground = get_distance_from_side(region, [DirType.down]) > 0.3
    not_on_ground = get_distance_from_side(region, [DirType.down]) > 0.05

    # The back frame should be attached to the seat
```

```
    seat_surface = find_region_by_label(None, None, 'chair/chair_seat/seat_surface')
    attached_to_seat = any(is_attached(region, seat) for seat in seat_surface)

    # Should be centered along width (not extending very far left or right)
    centered_width = get_distance_to_center(region, [DimType.width]) < 0.2

    if len(seat_surface) != 0:
        if upright_bar and height_above_ground and not_on_ground and attached_to_seat and centered_width:
            # fire the label
            return label
        else:
            return None
    else:
        # Partial condition: if no seat surfaces are labeled
        if upright_bar and height_above_ground and not_on_ground and centered_width:
            return label
        else:
            return None

def bsurface(region, label = "chair/chair_back/back_surface"):
    # The back surface is generally a large flat surface
    is_flat = is_axis_aligned(region, threshold=0.05)

    # The back surface should be oriented vertically and face the back direction
    is_vertical = is_oriented_bar(region, orientation=DirType.back) or is_oriented_bar(region, orientation=DirType.front)

    # The back surface should be relatively high up on the chair
    above_middle_height = get_distance_from_side(region, [DirType.down]) > 0.5

    # The back surface should be centrally located along the width dimension
    centrally_located_width = get_distance_to_center(region, [DimType.width]) < 0.1

    if is_flat and is_vertical and above_middle_height and centrally_located_width:
        # fire
        return label
    else:
        # don't fire
        return None

def leg(region, label="chair/chair_base/regular_leg_base/leg"):
        # The region should be well-fit by an upright bar: use orientation up or down to cover both cases
    is_upright_bar = is_oriented_bar(region, dim_ratio=2.0, orientation=DirType.up) or is_oriented_bar(region, dim_ratio=2.0, orientation=DirType.down)

```

```
    # The region should touch the ground
    touches_ground = get_distance_from_side(region, [DirType.down]) < 0.05

    # The region should NOT touch the top of the bounding box
    not_touches_top = get_distance_from_side(region, [DirType.up]) > 0.1

    # The region should be in one of the corners along width and depth dimensions
    in_corner = get_distance_to_center(region, [DimType.width, DimType.depth]) > 0.2

    if is_upright_bar and touches_ground and not_touches_top and in_corner:
        # fire the label
        return label
    else:
        # don't fire
        return None
```

### Example Human-Authored Rules

```
def arm_lf(ri, data):
    label = "chair/chair_arm"
    conds = [
        [(ru.onEdge, {'sind':0, 'thresh': 0.05})],
        [(ru.relToSeat, {'part': 'top', 'rel':'above', 'thresh': 0.15})],
        [(ru.relToSeat, {'part': 'bot', 'rel':'above', 'thresh': -0.025})],
        [(ru.touchesTop, {'NOT': True, 'thresh': 0.25})],
        [
            (ru.onSide, {'sind': 2, 'NOT':True}),
            (ru.isOrientedBar, {'ori': 'vert'}),
        ]
    ]
    return lf_and_or_helper(ri, data, conds, label)
BACK_CONDS = [
    [
        (ru.relToSeat, {'part': 'top', 'rel': 'above', 'thresh': 0.2}),
        (ru.touchesTop, {})
    ],
    [
        (ru.relToSeat, {'part': 'bot', 'rel': 'above', 'thresh': -0.05}),
        (ru.touchesTop, {})
    ],
    [(ru.onSide, {'sind': 2, 'gap_perc': 0.2, 'mode': 'p1'})]
]

def back_frame_lf_1(ri, data):
    label = "chair/chair_back/back_frame"
    conds = BACK_CONDS + [
        [(ru.isOrientedBar, {'ori': 'vert'})],
        [(ru.onEdge, {'sind':0, 'thresh': 0.05})]
    ]
    return lf_and_or_helper(ri, data, conds, label)

```

```
def back_frame_lf_2(ri, data):
    label = "chair/chair_back/back_frame"
    conds = BACK_CONDS + [
        [(ru.isOrientedBar, {'ori': 'horiz'})],
        [(ru.touchesTop, {})],
        [(ru.dimAbsComp, {'sind': 1, 'thresh': 0.1, 'mode': 'smaller'})
    ]
    ]
    return lf_and_or_helper(ri, data, conds, label)

def back_surface_lf(ri, data):
    label = "chair/chair_back/back_surface"
    conds = BACK_CONDS + [
        [(ru.lfFired, {'lfs': ('bf_1', 'bf_2'), 'NOT': True})]
    ]
    return lf_and_or_helper(ri, data, conds, label)

def base_leg_lf(ri, data):
    label = "chair/chair_base/regular_leg_base/leg"
    conds = [
    (ru.touchesGround, {'thresh': 0.05}),
    (ru.inCorner, {'plane': 'XZ'}),
    (ru.isOrientedBar, {'ori': 'vert'}),
    ]
    return lf_and_helper(ri, data, conds, label)
```

## A.3 Weak Label Generation Details

Many of the label functions utilize helper functions in the API that express relational constraints; for example, a chair back frame might be *attached* to both a back surface and the seat frame. This constraint depends on some regions already being labeled with the back surface or seat frame label. However, these respective label functions might have similar relational constraints that must be satisfied to fire. To avoid this type of circular dependency, we use two sets of label functions for each part: one function that uses purely non-relational constraints to generate initial labelings, and a second function that may use relational constraints for higher-level geometric reasoning. We run the relational label functions for all parts until saturation, i.e. no new label function fires, and record all of the fires from both the non-relational and relational functions.