

ARKit LabelMaker: A New Scale for Indoor 3D Scene Understanding

Guangda Ji
ETH Zürich
Switzerland

guanji@student.ethz.ch

Silvan Weder
ETH Zürich
Switzerland

silvan.weder@inf.ethz.ch

Francis Engelmann
Stranford University / ETH Zürich
USA / Switzerland

francis.engelmann@ai.ethz.ch

Marc Pollefeys
ETH Zürich
Switzerland

marc.pollefeys@inf.ethz.ch

Hermann Blum
Uni Bonn / ETH Zürich
Germany / Switzerland

blumh@uni-bonn.de

Abstract

The performance of neural networks scales with both their size and the amount of data they have been trained on. This is shown in both language and image generation. However, this requires scaling-friendly network architectures as well as large-scale datasets. Even though scaling-friendly architectures like transformers have emerged for 3D vision tasks, the GPT-moment of 3D vision remains distant due to the lack of training data. In this paper, we introduce ARKit LabelMaker, the first large-scale, real-world 3D dataset with dense semantic annotations. Specifically, we complement ARKitScenes [4] dataset with dense semantic annotations that are automatically generated at scale. To this end, we extend LabelMaker Weder et al. [33], a recent automatic annotation pipeline, to serve the needs of large-scale pre-training. This involves extending the pipeline with cutting-edge segmentation models as well as making it robust to the challenges of large-scale processing. Further, we push forward the state-of-the-art performance on ScanNet and ScanNet200 dataset with prevalent 3D semantic segmentation models, demonstrating the efficacy of our generated dataset.

1. Introduction

Recent progress in deep learning has been mostly focused on language [5, 25, 26] and 2D image generation [27]. Because for these two modalities, there is vast amount of training data available on the web. For text and image generation, you can simply scrape the internet for all available data and train your model in an auto-regressive (in the case of language) or diffusion process (image generation), where the supervision signal comes from self-supervision that does not require any labelling. This type of train-

ing revealed surprising properties [34] when scaling it to billions of data points in both text and image generation and leads to unseen performance gains enabling completely new use cases. Yet, this type of training is not available for scene understanding as objects and other primitives have to be classified in separate categories. This usually requires ground-truth annotations for the training data. While there are efforts to relax this requirement through self-supervision [12, 42] or open-set scene understanding [17, 36], state-of-the-art methods [19, 36] all have some form of direct supervision during training. Thus, annotated data is required for learning these tasks. Yet, creating datasets of similar scales as used in language and image generation is far from trivial. In this paper, we contribute the largest 3D real-world indoor semantic dataset and investigate the following questions: Is there a benefit from preferring real-world data over synthetic data? How can the labeling effort be reduced? Do current models profit from more real-world data?

To answer these questions, we make use of ARKitScenes [4], a large collection of RGB-D trajectories manually captured with consumer tablets. While these trajectories are annotated with sparse object bounding boxes, they are not complete enough to train competitive models for scene understanding and lack dense labels that could potentially serve as supervision for semantic segmentation models. Therefore, we supplement this dataset with dense semantic labels that we automatically generate using an automated pipeline. This allows us to create the first large-scale, dense 3D semantic segmentation training dataset that can be used to (pre-)train any 3D semantic segmentation model. To demonstrate the value of these vast yet imperfect annotations, we use these labels to re-train different models and extensively evaluate them on popular 3D semantic segmentation benchmarks.

More specifically, we build on top of the recent LabelMaker [33] pipeline, which we extend into LabelMakerv2 with more and updated base models, a more general input data structure, scripts to deploy the pipeline onto large clusters through docker or SLURM. Using this pipeline, we process the entire ARKitScenes dataset, which takes 48000 GPU hours on Nvidia 3090 GPUs. Further, we scale this approach beyond ARKitScenes by integrating commonly used scanning software into the pipeline. This allows to robustly generate automatic labels for any scan acquired with ubiquitously available mobile devices. This opens up a road to potentially reach so far unseen scales in 3D datasets for scene understanding. The automatically generated ARKitScenes labels are used to pre-train the currently most-used 3D segmentation methods, MinkowskiNet [6] and PTv3 [36]. We find that without labeling any data manually, extending the scale of real-world training data improves the performance of both models on multiple benchmarks, or achieves the same performance as with even larger synthetic data.

In summary, we answer the following research question: does large-scale pre-training with automatic labels show similar trends as it does for language and image generation tasks? We do that by making the following key contributions:

- Improving LabelMaker [33] to LabelMakerV2 by incorporating new base models and improving robustness to large-scale scenes.
- Generating the largest existing real-world dataset with dense semantic annotations on 186 classes.
- We push forward the state-of-the-art performance of point transformer on ScanNet and ScanNet200 by a large margin which our gerenated dataset.

The ArkitLabelMaker dataset and LabelMakerV2 pipeline code is available at labelmaker.org.

2. Related Works

Datasets for 3D semantic segmentation. 3D semantic segmentation is to classify each point in a 3D point cloud into a set of predefined semantic categories. Prominent datasets for training and evaluation include ScanNet [10]/ScanNet200 [29], and the Stanford 3D Indoor Scene Dataset [1] (S3DIS), which comprises 6 large-scale indoor areas with 271 rooms. Both datasets includes RGB-D frames captured from real-world. In addition to real-world datasets, Structured3D [41] offers a photo-realistic synthetic dataset with 3.5K house designs, and Replica [31] provides 18 high-quality reconstructed scenes. ARKitScenes [4] is the most extensive collection of indoor scenes to date, featuring 5047 captures of 1661 unique scenes. RGB-D data is captured with Apple LiDAR scanner. High-quality surface reconstruction and the bounding box for object detection are also provided. However, dense annotations for semantic or instance segmentation is absent

from this dataset. Therefore, it cannot be directly used in training models for 3D semantic segmentation.

LabelMaker Weder et al. [33] is an automatic semantic segmentation annotation pipeline that consolidates outputs from state-of-the-art 2D segmentation models with an additional feature for translating frame-wise 2D labels into cohesive 3D point cloud labels. In this work, we employ an enhanced version of LabelMaker to generate dense semantic segmentation annotations for ARKitScenes.

3D semantic segmentation models. The neural network architecture for processing 3D input data can be classified into three main categories: voxel-based, point-based, and transformer-based methods. Voxel-based methods transform points into fixed-sized voxel grids before passing them through the neural network. MinkowskiNet [6] is one of the most well-known model. Mix3D [21] enhances MinkowskiNet through effective 3D data augmentation techniques. PonderV2 [42] explores self-supervised learning from RGB-D data to improve the performance of the MinkowskiNet architecture. Point-based methods includes [3, 11, 15, 16, 23, 24, 32, 38]. However, there is a recent shift from models based on point-wise convolutions to point-based transformer models [14, 22, 30, 40]. Notable examples include Point Transformer [40], PTv2 [35], and PTv3 [36], which are developed towards better efficiency and scaling ability for 3D inputs. Point Prompt Training [37](PPT) introduces a novel training paradigm enabling the simultaneous training of multiple datasets with diverse label spaces. Combining PTv3 with PPT achieves state-of-the-art performance on the ScanNet/ScanNet200 semantic segmentation benchmark.

In this paper, we address the main limitation of existing datasets for 3D semantic segmentation: their limited size. We suspect that this limited size negatively impacts the performance of commonly used models as their performance is limited without additional training data available.

3. method

3.1. LabelMaker

As we build on top of LabelMaker [33], we briefly review the essential steps of the pipeline proposed by Weder et al.. LabelMaker is an automatic labelling pipeline for 2D and 3D semantic annotation. As shown in [33], it generates labels that are on par with the human annotators in [9]. It automatically generates these labels by exploiting an ensemble of base models that predict semantic maps for every input frame in an RGB-D trajectory. Since the base models predict segmentation in different label spaces based on their training data, they are then mapped to a unified label space. Only through this mapping, the different base models can be used in a subsequent ensemble. Thus, [33] defined a mapping from every label space into a carefully curated label

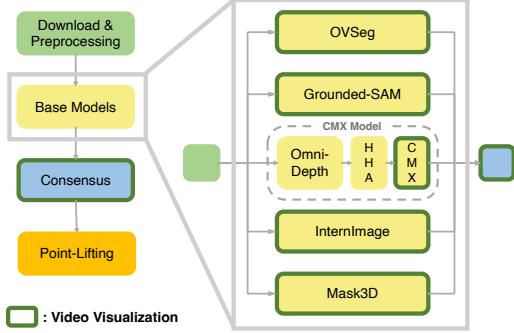


Figure 1. Dependency graph of the LabelMakerv2 pipeline. Our LabelMakerv2 pipeline has a clear dependency structure that has to be handled in the distributed processing of the data. This has to be especially respected when recovering from job failure. There, our recovery strategy checks for unfinished jobs in the dependency graph before submitting any new jobs to avoid unnecessarily wasting compute resources. The boxes with thick yellow frame denotes visualizable tasks. These are used during inspection and job quality assurance.

space based on wordnet synkeys. After mapping all base model predictions into the unified label space, the individual predictions are aggregated into one single consensus for every frame of the RGB-D trajectory. As the RGB-D trajectory provides multi-view information of the scene, the individual frames can be further denoised by lifting them into 3D and aggregate the point-wise predictions across the different frames. The final labels can either be directly used as 3D labels for 3D semantic segmentation or projected into 2D and be used for training or evaluating 2D semantic segmentation models. In this paper, we improve this pipeline to robustly scale to large-scale datasets and show its benefit to pretraining 3D semantic segmentation models. In the following, we describe the improvements and the scaling in more detail.

3.2. Improving Labelmaker to make it scale to ARKitScenes [4]

While LabelMaker [33] presented an automatic labelling tool that produces annotations on par with human annotators, we enhance the pipeline with two changes to further improve its performance that is necessary to robustly generate high-quality annotations for large-scale datasets. The complete pipeline is shown in Figure 1.

Integrating Grounded-SAM LabelMaker [33] used several state-of-the-art base models in its ensemble. Yet, they did not leverage the potential of Segment Anything (SAM) [13], a 2D segmentation model that was trained on a large-scale datasets and robustly generalizes to many scenarios. As we want to scale LabelMaker to any environment, our aim is to integrate this prior into the pipeline. However, it is not straight forward to efficiently use this model for semantic segmentation. To this end, Grounded SAM combines Grounding DINO [18] with SAM [13].

Grounding DINO locates instances’ bounding boxes given semantic labels or natural languages, while the later model generates high quality segmentation masks for these bounding boxes. We integrate this model by adapting it to Label-Maker’s unified labelspace such that it can act as an additional vote in the ensemble.

Aligning to gravity For ideal performance, many semantic segmentation models require alignment of the gravity direction with the coordinate system of the data they were trained on. Yet, large-scale data is not aligned with gravity by default. For example in ARKitScenes, occasional phone rotation leads to inconsistencies in the orientation of 2D images. Passing those rotated images to LabelMaker’s base models results in decreased performance and misclassifications (e.g. mistaking the floor with the ceiling). Therefore, we project sky direction, which corresponds to the z-axis of the pose coordinate system from ARKit (that uses the IMU), onto each 2D frame. Then, we compute the angle between sky direction and upward direction α . Given this angle, we rotate the image by $k \cdot \frac{\pi}{2}$, where $k = \arg \min_s (|s \frac{\pi}{2} - \alpha|)$ to make the sky direction roughly upward, and rotate the predicted segmentation back to its original orientation after passing it through model to align it with its coordinate system.

Optimizing compute resource scheduling. We optimize the code to deploy each individual piece of the pipeline of Figure 1 as individual jobs to a GPU cluster, with SLURM as a dependency manager between the pipeline pieces. To optimize the overall execution time, it is therefore important to be able to estimate the processing time of each piece of the pipeline at the point of job submission. ARKitScenes contains scenes of a wide range of sizes, spanning from a minimum of 65 frames to a maximum of 13796, and different parts of the pipeline scale differently with increasing scene size. To figure out the minimum resources requirements, we select 11 scenes of varied sizes uniformly distributed within the minimum and maximum range and record their resources usage. While most jobs are not sensitive to scene size and can suffice with a fixed resource allocation, the base models exhibit greater sensitivity to scene size. We interpolate resource needs with respect to scene size and summarize the empirical numbers in the Appendix. Through this, we ensure that we request minimal-required resources, so that we have lowest job waiting time and less idle compute power.

3.3. Scaling beyond existing datasets

In this paper, we demonstrate the effectiveness of the automated labels generated for a large-scale dataset for pre-training 3D semantic segmentation models. Yet, ARKitScenes is still limited in terms of variance in scene type. Through modern mobile devices, RGB-D scanning is ubiquitously available and a excellent source for 3D data. Yet,

it has not been possible to exploit this data for 3D semantic segmentation without expensive human annotation. While LabelMaker [33] provided an automatic annotation pipeline, it was not readily available for data provided by mobile devices. To overcome this limitation, we integrated a well-established scanning platform for iOS (Scanner 3D) into our imported LabelMaker pipeline. This allows anyone with an iOS device to collect data and automatically generate human-quality annotations that can either be used for training or evaluation of 3D semantic models at an unseen scale.

To this end, we built a integration of Scanner 3D into LabelMakerv2. This involves two essential steps. First, we align the coordinate system of Scanner3D with the coordinate system of LabelMakerv2 by transforming all individual frames into the LabelMakerv2 system. Second, we have to solve the challenge that does not provide direct access to the individual depth frames but only provides a dense reconstructed mesh. To overcome this hurdle, we render a depth map from the mesh for every camera view in the scanned trajectory. Therefore, we also align the mesh with the LabelMakerv2 coordinate system and use Open3D to render the corresponding depth maps. Together with the corresponding RGB images and camera poses, these depth maps are subsequently transformed into the LabelMakerv2 format such that the trajectory can be processed by LabelMakerv2.

4. Results

4.1. Baselines

We evaluate the effectiveness of our ARKitScenes LabelMaker dataset on two distinct network architectures: MinkowskiNet [6] and Point Transformer [8, 35–37]. MinkowskiNet is the most established architecture. Many modifications [19, 42] have been proposed to MinkowskiNet and it is still the underlying architecture of most top-performing models in 3D Semantic Segmentation benchmarks. Point Transformer [36] is a very recently proposed architecture and the current state-of-the-art on the ScanNet and ScanNet200 benchmarks. Since transformers are in general known to profit from large-scale training data, we also train on this architecture. From these two architectures, we derive our three relevant baselines:

Vanilla MinkowskiNet. This is the standard MinkowskiNet model based on [6], which most 3D semantic segmentation methods compare to. In this paper, we use the commonly used ‘Res16UNet34C’ variant of MinkowskiNet to guarantee fair comparison to all other baselines.

PonderV2 [42]. PonderV2 [42] is an unsupervised pre-training strategy for semantic segmentation. For large-scale training data, manual labelling of training data is infeasible and leaves essentially two alternatives: Automatic labelling

of the training data, which is what we investigate in this work, or an unsupervised strategy that learns useful features on the data from a different loss. PonderV2 is the state-of-the-art method for unsupervised training. While the best performing ponder-based model combines different training datasets, model architecture modifications, and the unsupervised pretraining, our evaluation focuses on comparing the two pretraining paradigms.

PointTransformerV3 (PTv3) [36]. Point Transformer V3 (PTv3) [36] is a recently proposed method that aims to speed up transformer architecture and scale up. Point Prompt Training (PPT) is a large-scale training paradigm aiming to jointly train multiple datasets of various label spaces together. This is a different approach than the LabelMaker [33] approach of translating label spaces and then training on a common label space. The combination of PTv3 and PPT achieves state-of-the-art results on ScanNet/ScanNet200 semantic segmentation benchmark.

4.2. Datasets and Metrics for Evaluation

ScanNet [9]. ScanNet comprises 1513 densely annotated scans across 707 distinct indoor scenes, totaling 2.5 million RGB-D frames. It stands as one of the most widely used and influential benchmark datasets for indoor 3D scene understanding. ScanNet is annotated by humans using the NYU40 label space and evaluated on a subset of 20 classes from NYU40.

ScanNet200 [28]. While only 20 classes are used in the ScanNet benchmark, the original dataset is annotated with many more classes. ScanNet200 [28] leverages these annotations and organizes them into a new benchmark with 200 classes that are of higher-resolution than the original ScanNet classes. Given the large-number of different categories generated by our LabelMakerv2 pipeline, we also pre-train the models for this task and evaluate them on the ScanNet200 benchmark.

ScanNet++ [39]. ScanNet++ is a dataset of 460 high-resolution 3D indoor scenes with dense semantic and instance annotations, captured using a high-precision laser scanner and registered images from a DSLR camera and RGB-D streams. It focuses on long-tail and multi-labeled annotations. In its semantic segmentation benchmark, models are evaluated over 100 labels.

S3DIS [2]. S3DIS is a 3D semantic dataset containing 6 large-scale indoor areas from 3 different buildings, labeled with 13 semantic classes. We follow the practice of [36] and create a dataset with an effective size of 406. We only use this dataset during the training of the PTv3 [36] baseline.

Structured3D [41] is a large-scale indoor synthetic RGB-D dataset featuring 6519 training scenes and 1697 test scenes. It is annotated with a label space of 25 classes. Structured3D and S3DIS only used in PTv3+PPT joint training and we adopt pre-processed version of these two

datasets from [36]. The actual dataset size used in training is different from the original, and we refer to the actual number in Table 1.

ARKit LabelMaker (Ours). This is the dataset generated with our method described above. The resulting dataset contains 5019 trajectories, from which we take 4471 for training and 548 for validation according to the official train-val split provided by the original ARKitScenes [4] dataset. For every scene, we created 3D pointcloud associated dense semantic labels in the original LabelMaker wordnet label space (186 classes). We denote this dataset as ALC. Moreover, we utilized the provided mapping from wordnet label space to the ScanNet200 label space and mapped all labels to this space. This resulted in our ALS200 dataset. We use this data to train the different baseline models described above. To increase efficiency and make the experimental settings comparable to previous studies, we perform down-sampling on 3D meshes to a voxel size of 2cm. Normal information is preserved and down-sampled simultaneously. Table 1 illustrates the scale of each dataset. ARKitScenes LabelMaker dataset is the largest diversely annotated real-world indoor semantic dataset.

Metrics. We follow the standard metrics of the ScanNet semantic segmentation task and compute the mean and per-class intersection-over-union ratio (IoU) on ScanNet/ScanNet 200.

Dataset	#train	#val	#test	real	#label
S3DIS	406	-	-	✓	13
ScanNet/ScanNet200	1201	312	100	✓	20 / 200
ScanNet++	230	50	50	✓	100
ARKit LabelMaker	4471	548	-	✓	186
Structured3D	6519	-	1697	✗	25

Table 1. **The size of dataset that is used for training and evaluation in this work.** We provide by far the largest real-world labeled training dataset compared to existing real-world datasets. We provide automatically generated dense semantic annotations for 4471 training trajectories and 548 validation trajectories.

4.3. Experiment Settings

We adopt three approaches to evaluate the effectiveness of our ARKitScenes LabelMaker dataset.

Pre-training. To investigate whether automatic labels are useful to learn strong features from imperfect annotations, we pre-train both, MinkowskiNet [7] and PointTransformerV3, on our generated ALS200 dataset. Afterwards, we fine-tune the pretrained models on the ScanNet and ScanNet200 dataset, respectively.

For MinkowskiNet, we employ the Res16UNet34C architecture as our backbone model. During pre-training, we utilize the AdamW optimizer with a learning rate of 0.01 and OneCycleLR scheduler, training the network for 600

epochs. If the label space is changed for fine-tuning, we replace the classification head and exclusively train it with the same learning rate setting until convergence while the rest of the model is fixed. Then, the entire network undergoes fine-tuning with a learning rate of 0.001, while other settings are kept unchanged.

For PTv3, we adhere to the settings outlined in [36] employing the AdamW optimizer with OneCycleLR for 800 epochs of training. However, the learning rate during pre-training is adjusted to 0.0016. Similar to the fine-tuning of MinkowskiNet, we initially freeze the backbone during fine-tuning and solely train the classification head until convergence. Then, we fine-tune on ScanNet or ScanNet200 with a reduced learning rate of 0.0006. Besides ALS200, we also pre-train PTv3 with ALC label space as the mapping from wordnet to ScanNet200 may reduce neural stimulation.

Co-training with ALS200. With this experiment, we aim to investigate if our ALS200 dataset can be simply combined with existing datasets for increasing the dataset size and, therefore, the performance of the resulting model. To this end, we combine ALS200 with the ScanNet200 dataset and train a MinkowskiNet from scratch. Due to resource limitations, we could only train MinkowskiNet with this combined dataset. The training setting is exactly same as the pre-training stage of MinkowskiNet described above.

Joint-training. We employ PTv3+PPT for joint training on multiple datasets across multiple label spaces. Besides ScanNet/ScanNet200, ScanNet++, S3DIS and Structured3D, we add our ALC dataset. We choose LabelMaker’s wordnet label space in order to provide the network with maximum possible semantic class stimulation. We use the exact same setting of PTv3+PPT from [36]. We use AdamW optimizer with OneCycleLR scheduler and a learning rate of 0.05. Additionally, we incorporate the LabelMaker WordNet label space into the norm layer and the final classification head.

4.4. Results

In Table 2, we present the results for the ScanNet dataset. In the case of MinkowskiNet [7], we can not only show that pre-training on our large-scale, real-world ALS200 dataset significantly improves the mean intersection-over-union compared to vanilla training, but it also significantly outperforms to other variants of pre-training. Pre-training on our imperfect yet automatically generated labels is superior to self-supervised pre-training (PonderV2 [42]) and extensive data augmentation (Mix3D [20]). This indicates that direct supervision with scale in training data for supervised learning is essential for 3D semantic segmentation. Additionally, the ALS200/ALC pre-trained PTv3 exhibits comparable or superior performance to large-scale multi-dataset joint training. This proves that our dataset is more

Method	Training Data	val	test
MinkUNet [7]			
vanilla	ScanNet	72.4	73.6
PonderV2 [42]	ScanNet (self-supervised) → ScanNet	73.5	-
Mix3D [20]	ScanNet	73.6	78.1
fine-tune (Ours)	ALS200 → ScanNet	77.0	-
PTv3 [36]			
vanilla	ScanNet	77.5	77.9
fine-tune (Ours)	ALS200 → ScanNet	81.2	-
fine-tune (Ours)	ALC → ScanNet	80.6	79.0
PPT [36]	ScanNet + S3DIS + Structure3D	78.6	79.4
PPT (Ours)	ScanNet+ ScanNet200 + ScanNet++ + Structure3D + ALC	81.1	79.8

Table 2. Semantic Segmentation Scores on ScanNet20. We compare different training strategies for two top-performing models (PointTransformerv3 [36] and MinkowskiNet [7]) on the ScanNet20 dataset. We can show for both models adding ALS200 through pre-training and co-training improves the performance for both models. With PonderV2 [42] and Mix3D [20], we compare large-scale pretraining to two other training strategies. We can show that large-scale pre-training is superior to both, extensive data augmentation (Mix3D) and self-supervised pre-training (PonderV2).

Method	Training Data	val	test
MinkUNet [7]			
vanilla	ScanNet200	29.3	25.3
fine-tune (Ours)	ALS200 → ScanNet200	30.1	27.4
co-training (Ours)	ALS200 + ScanNet200	30.6	-
PTv3 [36]			
vanilla	ScanNet200	35.2	37.8
fine-tune (Ours)	ALS200 → ScanNet200	38.4	-
fine-tune (Ours)	ALC200 → ScanNet200	38.7	38.4
PPT [36]	ScanNet200 + S3DIS + Structure3D → ScanNet200	36.0	39.3
PPT(Ours)	ScanNet+ ScanNet200 + ScanNet++ + Structure3D + ALC	40.3	41.4

Table 3. Semantic Segmentation Scores on ScanNet200 [29]. To investigate our large-scale dataset also helps with long-tail categories, we evaluate it on the ScanNet200 dataset. For both, MinkowskiNet [7] and PointTransformerv3 [36], we compare it to vanilla training as well as training procedure proposed in [36]. We can show that common neural networks benefit from pre-training on automatically generated large-scale annotations compared.

data-efficient compared with synthetic dataset.

Similar trends are also visible in Table 3, where we show the results on the ScanNet200 dataset. The co-training of MinkowskiNet on the ScanNet200 dataset demonstrates that our ALS200 dataset facilitates training without inducing a domain gap in comparison to the ScanNet200 dataset.

We also incorporate our ARKit LabelMaker dataset into the joint training for PTv3+PPT, leading to a significant boost in validation and test mIoU. This version of PTv3 achieves state-of-the-art performance on both ScanNet and ScanNet200, with the latter benefiting greatly from improved tail class mIoU (Table B1). Our dataset’s effectiveness serves as a good complementary for rare classes in ScanNet200.

In Table 4 we show the results on ScanNet++ dataset. As on the other datasets, pre-training on ALC improves vali-

PTv3 Variant	Training Data	#Data	val mIoU	test top-1/3 mIoU
vanilla	ScanNet++	713	41.8	45.8 / 69.7
fine-tune (Ours)	ALC200 → ScanNet++	4471 → 713	42.5	43.7 / 65.5
PPT [36]	ScanNet200 + ScanNet++ + Structure3D	45868	45.3[†]	46.5 / 71.1
PPT (Ours)	ScanNet200 + ScanNet++ + ALC	11168	44.5	46.1 / 70.8
PPT (Ours)	ScanNet+ ScanNet200 + ScanNet++ + Structure3D + ALC	30386	44.6	46.1 / 68.5

Table 4. Semantic Segmentation Scores on ScanNet++ [39]. We evaluated the efficacy of our ALC dataset on the ScanNet++ benchmark using both pre-training and joint training methods. †: this number comes from Wu et al..



Figure 2. Self-captured scenes and the semantic segmentation generated by LabelMakerv2. This figure shows visually appealing segmentation results, therefore illustrates the effectiveness of our pipeline. The color mapping is defined in original Label-Maker’s repository.

dation set performance compared to vanilla PTv3 training on ScanNet++. However, test set performance deteriorates. For joint training, we conducted two trials: one with a small and one with a larger training scale. In the first, we excluded Structured3D, as PTv3’s updated version (28,212 samples) was too large for our machines. In the second trial, we use the old version of Structured3D which has 8216 samples and followed Wu et al.’s suggestion to exclude S3DIS due to its high memory consumption. These results suggest that the ALC dataset may have limited benefits for ScanNet++ training.

4.5. LabelMakerv2 is generalizing beyond ARKitScenes

In order to demonstrate the effectiveness of our LabelMakerv2, we utilize it to process two self-captured scenes—a kitchen and a fireplace—taken in a holiday cottage using an iPhone 12 Max. In Figure 2, we present the reconstructed colored scenes alongside their semantic segmentation. It demonstrates visual accuracy and plausibility, confirming the effectiveness of our pipeline.

4.6. Limitations & Broader Impact

While we extend LabelMaker [33] with a better pointcloud pipeline, we leave out the part that generates 2D segmentation maps. The computational cost of the NeRF-based lifting over the whole ArKitScenes dataset is beyond our available resources. It would be an interesting future direction of research to explore if training 2D models on this data yields similar performance gains as it is the case for 3D models.

Furthermore, 20 scenes in ARKitScenes processing are excluded due to lack of pose data. Our LabelMakerv2 requires accurate poses, but future iterations could integrate techniques such as bundle adjustments to reconstruct missing pose data.

Like the original LabelMaker [33], also our improved pipeline does not have perfect accuracy. While [33] showed that the accuracy is on par with crowd-sourced human annotations, there is always a danger of introducing systematic mistakes when training on noisy labels. For safety critical applications, rigorous testing on accurately annotated data is even more important when using tools like ours to source training data.

Does large-scale pre-training with automatic labels show similar trends as it does for language and image generation tasks? The discussed results point in this direction, with a measurable improvement to different models when pretraining on ArKitLabelMaker. However, training on large-scale real-world data ‘only’ achieves test results on par with the current SOTA based on synthetic data of even larger scale. Our conclusion is that real-world data is much more effective than synthetic data, but even larger overall scale is necessary to push the performance beyond state-of-the-art. Our developed pipeline makes it easy to provide training data once more scans are available.

5. Conclusion

In this paper, we presented the largest, real-world 3D RGB-D dataset with dense semantic annotations. The dense annotations are automatically generated using an improved version of LabelMaker [33], which we dub LabelMakerv2. While these labels are automatically generated and therefore imperfect, we demonstrate their value to pre-training commonly used 3D semantic segmentation methods significantly improving the performance of existing models trained with traditional, self-supervised, or augmentation-heavy training strategies. This allows to draw parallels to recent advances in language and image generation, where scaling up the size of training data led to huge gains in performance. Therefore, we also provide an integration of a commonly available 3D scanning software for iOS to enable the usage of mobile devices to easily generate more data for training and evaluation.

Acknowledgement

We give our many thanks to Xiaoyang Wu, one of the authors of Point Transformer V3 [36], who gave us many advice on the training of PTv3.

ARKit LabelMaker: A New Scale for Indoor 3D Scene Understanding

Supplementary Material

A. Dataset and code download availability

Except from labelmaker.org, our dataset is available on huggingface at https://huggingface.co/datasets/labelmaker/arkit_labelmaker with DOI of 10.57967/hf/2389. The pipeline code for LabelMakerV2 is available at <https://github.com/cvg/LabelMaker/>. The preprocessing and training code for MinkowskiNet model and PointTransformerV3 is documented in <https://github.com/quantaji/labelmaker-mix3d> and <https://github.com/quantaji/LabelMaker-Pointcept>.

B. Head, common and tail split mIoU scores for ScanNet200

In Table B1, we present the statistics for head, common, and tail classes. Our ALS200-pretrained MinkowskiNet significantly outperforms in common class mIoU and the jointly trained version of PTv3 shows a large increase in tail class mIoU.

Method	Training Data	Validation			Test	
		head	common	tail	head	common
MinkUNet [7]						
vanilla	ScanNet200	52.3	22.5	13.2	46.3	15.4
fine-tune (Ours)	ALS200 → ScanNet200	53.9	24.2	12.5	49.0	19.4
co-training (Ours)	ALS200 + ScanNet200	55.1	24.7	12.4	-	-
PTv3 [36]						
vanilla	ScanNet200	56.5	30.1	19.3	-	-
fine-tune (Ours)	ALS200 → ScanNet200	58.6	33.0	23.8	-	-
fine-tune (Ours)	ALC200 → ScanNet200	58.2	33.1	25.0	58.2	30.9
PPT [36]	ScanNet200 + S3DIS + Structure3D → ScanNet200	-	-	-	59.2	33.0
PPT(Ours)	ScanNet+ ScanNet200 + ScanNet++ + Structure3D + ALC	60.9	35.48	24.6	61.0	32.2
						27.1

Table B1. ScanNet200 validation and test mIoU for head, common and tail classes. For MinkowskiNet, ARKit LabelMaker pre-trained network shows significant improvement on head and common classes. For PTv3, we see improvements across all three splits.

C. Detailed process of applying LabelMaker to ARKitScenes

ARKitScenes is one of the largest indoor 3D scenes dataset. It consists of 5047 parsable scenes of various size. We consider a scene parsable if the RGB-D trajectory comes with associated pose data. Processing these scenes with our improved LabelMaker pipeline requires deliberate engineering with the following goals: a) Bring the data in to the format required by LabelMaker [33] b) Robust processing to not waste compute on failures, c) Improved parallelization to speed up processing. d) Accurate resource estimation to prevent waste of compute resources and longer job waiting time. e) Fast failure identification and results inspection.

Transforming the data LabelMaker [33] requires a specific data format to be able to reliably process all data. All trajectories require posed RGB-D data and a denoised 3D model that is used by Mask3D. ARKitScenes comprises data of varying resolutions and sampling rates. Depth data is captured at 256×192 and 60 FPS, while the RGB frames are recorded at 640×480 and 30 FPS. Therefore, synchronization is required to process the data with LabelMaker. To this end, we match each RGB frame with the closest depth frame in time and we resize the depth frame to RGB frame's resolution. Pose data, originally at 10 FPS, is interpolated using rotation splines to synchronize with each RGB frame. To obtain a 3D mesh of each scene that can be processed by Mask3D, we reconstruct the 3D model by fusing the synchronized posed RGB-D data using TSDF fusion and then extract mesh with marching cube algorithm. We empirically chose a voxel size of 8mm and a truncation distance of 4cm for fusion.

Building a scalable pipeline LabelMaker [33] can be decomposed into individual modules such as the individual base models, the consensus computation, and the 3D lifting. This modular nature allows to build a scalable pipeline using popular high-performance computing toolboxes. As the different base models have different runtimes, we had to leverage dependency management system that can handle different dependencies of the pipeline steps. This architecture allows us to effectively leverage large-scale computing and distribute the processing across many different nodes.

In more detail, we decompose the pipeline into several jobs (ordered by dependency) for each scene:

1. Preprocessing: Downloading the original scene data, transforming it into LabelMaker format, and run TSDF fusion to get the 3D mesh of the scene.
2. Forwarding 2D images or 3D meshes to each base models: Grounded-SAM, Mask3D, OVSeg, CMX, InternImage. (all jobs depends on step 1.)
3. Getting the consensus label from base models' labels. (depends on all elementary jobs in step 2.)
4. Lifting the 2D consensus label into 3D. (depends on step 3.)
5. Rendering the outputs of base models or consensus into videos for visualization. (depends on steps 2. or 3.)
6. Post-processing, including removing temporary files and get statistics of each tasks. (depends on all steps above)

Optimizing compute resource scheduling. ARKitScenes contains scenes of a wide range of sizes, spanning from a minimum of 65 frames to a maximum of 13796, and different parts of the pipeline scale differently with increasing scene size. To figure out the minimum resources re-

Task	#CPU	RAM	Time	GPU
Download & Prepossessing	2	24G	4h	-
Video Rendering	8	32G	30min	-
Grounded-SAM	2	12G	6h	3090 × 1
OVSeg	2	8G	8h	3090 × 1
InternImage	2	10G	8h	3090 × 1
Mask3D	8	16G	1h 30min	3090 × 1
OmniData	8	8G	2h	3090 × 1
HHA	18	9G	2h	-
CMX	2	8G	3h	3090 × 1
Consensus	16	16G	2h	-
Point Lifting	2	72G	4h	-

Table C2. Requested resources for each task. We report the average resources required by the individual steps of the LabelMakerv2 pipeline. The required cores, RAM, and GPU time varies across the different jobs. Through our job scheduling mechanism, we ensure that the required compute is optimally distributed across all jobs.

uirements, we select 11 scenes of varied sizes uniformly distributed within the minimum and maximum range and record their resources usage. While most jobs are not sensitive to scene size and can suffice with a fixed resource allocation, the base models exhibit greater sensitivity to scene size. We interpolate resource needs with respect to scene size and summarize the empirical numbers into Table C2. Through this, we ensure that we request minimal-required resources, so that we have lowest job waiting time and less idle compute power.

Assuring the quality of the individual processings. In order to assure high-quality labels produced by our improved pipeline, we have built tooling to efficiently check for failures of the processed scenes. To this end, we store the logs and statistics of each job and built visualization tools for this data as well as the intermediate predictions. This allows us to conveniently browse at scale through the predictions and identify individual failures.

Failure handling and compute resource optimization. When doing large-scale processing on a high-performance compute cluster, a common issue is the failure of jobs. This can happen for several reasons such as node crashing, unexpected memory usage, and many more. Therefore, the processing pipeline has to be robust to these failures. Additionally, compute should not be wasted when recovering from these failures. Therefore, we designed a simple yet effective strategy for efficiently recovering from job failures. Before every restart is triggered for a scene, we analyze both the logs and file system to identify the jobs that have not finished for this scene. Once these jobs have been identified, we only resubmit these jobs. This ensures that no compute resource is used in rerunning completed tasks.

C.1. Implementation Details

We use a CentOS 7 based SLURM cluster to process all the data, which is capable of handling task dependencies and parallel processing. Before submitting jobs for a single scene, we first check the progress of each job and generate a SLURM script to submit only those unfinished jobs. This ensures that no compute resource is used in rerunning completed tasks.

We employ test time augmentation by forwarding all models twice, with Mask3D using two different random seeds and other models being mirror flipped. Following the practice of LabelMaker [33], we assign equal weights to each model when calculating the consensus, although these weights are configurable in our pipeline code. Since we are primarily interested in the 3D labels that can be used for pre-training 3D semantic segmentation models, SDFStudio training and rendering are omitted due to their lengthy processing times. Further, we enhance the pipeline by storing both the most and second most voted predictions alongside their respective vote counts. This information is useful when we want to investigate on the uncertainty across the base models. We leave the exploitation of this information as potential future direction of research.

References

- [1] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3D Semantic Parsing of Large-Scale Indoor Spaces. In *CVPR*, 2016. [2](#)
- [2] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese. Joint 2D-3D-Semantic Data for Indoor Scene Understanding. *ArXiv e-prints*, 2017. [4](#)
- [3] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. *arXiv*, 2018. [2](#)
- [4] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Yuri Feigin, Peter Fu, Thomas Gebauer, Daniel Kurz, Tal Dimry, Brandon Joffe, Arik Schwartz, et al. Arkitscenes: A diverse real-world dataset for 3d indoor scene understanding using mobile rgb-d data. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. [1, 2, 3, 5](#)
- [5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. [1](#)
- [6] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019. [2, 4](#)
- [7] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *CVPR*, 2019. [5, 6, 1](#)

- [8] Pointcept Contributors. Pointcept: A codebase for point cloud perception research. <https://github.com/Pointcept/Pointcept>, 2023. 4
- [9] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017. 2, 4
- [10] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes. In *CVPR*, 2017. 2
- [11] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise Convolutional Neural Network. In *CVPR*, 2018. 2
- [12] Li Jiang, Zetong Yang, Shaoshuai Shi, Vladislav Golyanik, Dengxin Dai, and Bernt Schiele. Self-supervised pre-training with masked shape prediction for 3d scene understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1168–1178, 2023. 1
- [13] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023. 3
- [14] Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia. Stratified Transformer for 3D Point Cloud Segmentation. In *CVPR*, 2022. 2
- [15] Loic Landrieu and Martin Simonovsky. Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs. In *CVPR*, 2018. 2
- [16] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhuan Di, and Baoquan Chen. PointCNN: Convolution on X-transformed Points. In *NeurIPS*, 2018. 2
- [17] Feng Liang, Bichen Wu, Xiaoliang Dai, Kunpeng Li, Yinan Zhao, Hang Zhang, Peizhao Zhang, Peter Vajda, and Diana Marculescu. Open-vocabulary semantic segmentation with mask-adapted clip. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7061–7070, 2023. 1
- [18] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023. 3
- [19] Alexey Nekrasov, Jonas Schult, Or Litany, Bastian Leibe, and Francis Engelmann. Mix3D: Out-of-Context Data Augmentation for 3D Scenes. In *International Conference on 3D Vision (3DV)*, 2021. 1, 4
- [20] Alexey Nekrasov, Jonas Schult, Or Litany, Bastian Leibe, and Francis Engelmann. Mix3D: Out-of-Context Data Augmentation for 3D Scenes. 2021. 5, 6
- [21] Alexey Nekrasov, Jonas Schult, Or Litany, Bastian Leibe, and Francis Engelmann. Mix3d: Out-of-context Data Augmentation for 3D Scenes. 2021. 2
- [22] Chunghyun Park, Yoonwoo Jeong, Minsu Cho, and Jaesik Park. Fast Point Transformer. In *CVPR*, 2022. 2
- [23] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *CVPR*, 2017. 2
- [24] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *NeurIPS*, 2017. 2
- [25] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018. 1
- [26] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 1
- [27] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 1
- [28] David Rozenberszki, Or Litany, and Angela Dai. Language-grounded indoor 3d semantic segmentation in the wild. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 4
- [29] David Rozenberszki, Or Litany, and Angela Dai. Language-Grounded Indoor 3D Semantic Segmentation in the Wild. In *ECCV*, 2022. 2, 6
- [30] Jonas Schult, Francis Engelmann, Alexander Hermans, Or Litany, Siyu Tang, and Bastian Leibe. Mask3D for 3D Semantic Instance Segmentation. 2023. 2
- [31] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv*, 2019. 2
- [32] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. KPConv: Flexible and Deformable Convolution for Point Clouds. In *ICCV*, 2019. 2
- [33] Silvan Weder, Hermann Blum, Francis Engelmann, and Marc Pollefeys. LabelMaker: Automatic Semantic Label Generation from RGB-D Trajectories. In *International Conference on 3D Vision (3DV)*, 2024. 1, 2, 3, 4, 7
- [34] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022. 1
- [35] Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point transformer v2: Grouped vector attention and partition-based pooling. In *NeurIPS*, 2022. 2, 4
- [36] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler, faster, stronger. In *CVPR*, 2024. 1, 2, 4, 5, 6, 7

- [37] Xiaoyang Wu, Zhuotao Tian, Xin Wen, Bohao Peng, Xihui Liu, Kaicheng Yu, and Hengshuang Zhao. Towards large-scale 3d representation learning with multi-dataset point prompt training. In *CVPR*, 2024. [2](#), [4](#)
- [38] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. SpiderCNN: Deep Learning on Point Sets with Parameterized Convolutional Filters. In *ECCV*, 2018. [2](#)
- [39] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2023. [4](#), [6](#)
- [40] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point Transformer. In *ICCV*, 2021. [2](#)
- [41] Jia Zheng, Junfei Zhang, Jing Li, Rui Tang, Shenghua Gao, and Zihan Zhou. Structured3d: A large photo-realistic dataset for structured 3d modeling. In *Proceedings of The European Conference on Computer Vision (ECCV)*, 2020. [2](#), [4](#)
- [42] Haoyi Zhu, Honghui Yang, Xiaoyang Wu, Di Huang, Sha Zhang, Xianglong He, Tong He, Hengshuang Zhao, Chun-hua Shen, Yu Qiao, and Wanli Ouyang. PonderV2: Pave the way for 3d foundation model with a universal pre-training paradigm. *arXiv preprint arXiv:2310.08586*, 2023. [1](#), [2](#), [4](#), [5](#), [6](#)