Reproducibility Companion Paper: Campus3D: A Photogrammetry Point Cloud Benchmark for Outdoor Scene Hierarchical Understanding

Yuqing Liao¹ Xinke Li² Zekun Tong² Yabang Zhao²
 Andrew Lim³ Zhenzhong Kuang⁴ Cise Midoglu⁵
 yuqingliao@whu.edu.cn, {xinke.li; zekuntong; zhaoyabang}@u.nus.edu
 i@limandrew.org, zzkuang@hdu.edu.cn, cise@simula.no

¹School of Computer Science, Wuhan University, China ²National University of Singapore, Singapore

³Southwest Jiaotong University, China ⁴Hangzhou Dianzi University, China ⁵SimulaMet, Norway

ABSTRACT

This companion paper is to support the replication of paper “Campus3D: A Photogrammetry Point Cloud Benchmark for Outdoor Scene Hierarchical Understanding”, which was presented at ACM Multimedia 2020. The supported paper’s main purpose was to provide a photogrammetry point cloud-based dataset with hierarchical multilabels to facilitate the area of 3D deep learning. Based on this provided dataset and source code, in this work, we build a complete package to reimplement the proposed methods and experiments (i.e., the hierarchical learning framework and the benchmarks of the hierarchical semantic segmentation task). Specifically, this paper contains the technical details of the package, including file structure, dataset preparation, installation package, and the conduction of the experiment. We also present the replicated experiment results and indicate our contributions to the original implementation.

CCS CONCEPTS

• **Computing methodologies** → **Scene understanding**; **Neural networks**; **3D imaging**.

KEYWORDS

Reproducibility; point cloud; scene understanding; hierarchical learning; semantic segmentation; instance segmentation; dataset

ACM Reference Format:

Yuqing Liao, Xinke Li, Zekun Tong, Yabang Zhao, Andrew Lim, Cise Midoglu, Zhenzhong Kuang. 2021. Reproducibility Companion Paper: Campus3D: A Photogrammetry Point Cloud Benchmark for Outdoor Scene Hierarchical Understanding. In *Proceedings of the 29th ACM International Conference on Multimedia (MM '21)*, October 20–24, 2021, Virtual Event, China. ACM, New York, NY, USA, 5 pages. <https://doi.org/10.1145/3474085.3477934>

This work is licensed under a Creative Commons Attribution International 4.0 License.

MM '21, October 20–24, 2021, Virtual Event, China.

© 2021 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-8651-7/21/10.

<https://doi.org/10.1145/3474085.3477934>

1 INTRODUCTION

To facilitate the research of 3D deep learning, the original paper “Campus3D: A Photogrammetry Point Cloud Benchmark for Outdoor Scene Hierarchical Understanding” [4] presents a richly annotated 3D point cloud dataset for multiple outdoor scene understanding tasks. The dataset is generated via photogrammetry processing on unmanned aerial vehicle (UAV) images from a campus of 1.6 km² area. One key feature of this dataset is the hierarchical multilabel annotation for each point, which introduces a novel task to the field of 3D scene understanding field, namely, the hierarchical segmentation of 3D point clouds. Based on such annotations, the original paper proposed a two-stage method of including **multi-task (MT)** learning on each level of the label hierarchy and **hierarchical ensemble (HE)** of predicted results to cope with the hierarchical consistency issue. Along with this framework, the original paper also established a benchmark of the hierarchical segmentation task containing three commonly used 3D deep models: PointNet++ [9], PointCNN [5], and DGCNN [10]. Please refer to the original paper for the details of the framework and the benchmark. In this paper, we provide a comprehensive and handy open-source package for reproducing the results in the original paper.

The software package is available at [6]. The program is written in Python Language based on the original package [2]. Other GitHub repositories are also referenced for the implementation of 3D point cloud models (i.e., PointNet++ [12], PointCNN [1], and DGCNN [11]). Figure 1 illustrates the file structure of the whole package. We briefly introduce the important folders and files in the root folder as below:

- **"checkpoints"**: This folder includes output records and the trained models with ".t7" format.
- **"configs"**: This folder includes the configuration files for the experiments.
- **"data"**: This folder includes the downloaded data files and file lists for the conduction of experiments.
- **"dataset"**: This folder includes the processing codes for the dataset. Those files inside it have two main functions: loading and sampling the point cloud dataset.
- **"models"**: This folder includes the implementations of three different 3D point cloud models.
- **"utils"**: This folder includes some helper functions, e.g., the calculation of loss and IoU, point cloud processing, and configuration settings.

- "eval.py": This file includes the codes of model evaluation.
- "run.py": This file is the main run file for all experiments.

In the following, we present the environment installation and dataset preparation procedures in Section 2, and the experiments including the details of training and evaluation in Section 3.1. To realize the evaluation and training, we show the execution commands in Section 3. The dataset and pretrained models can be downloaded from [3] and the links in the **README.md** file.

Figure 1: File structure

2 INSTALLATION

2.1 Environment Installation

The complete installation package can be downloaded by the following command,

```
$ git clone https://github.com/Yuqing-Liao/reproduce-campus3d.git
```

The requirements for reproducing our results are listed below.

- **System and Hardware** We note that the whole reproducibility package was tested on Ubuntu 20.04 operation system. The testing environment consisted of Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz CPU, 128GB memory, 1TB free space, and the graphics processing unit (GPU) NVIDIA GeForce RTX 2080Ti.

- **CUDA Toolkit and CuDNN** Tested with CUDA==10.1 and CuDNN==7.6.3 developed by NVIDIA[7, 8].
- **Version and Dependencies:** The tested python version is 3.7.8, and the dependencies of python packages are listed below. To easily install the dependencies, it is recommended to run "pip install -r requirements.txt" in the root folder of the package.

```
h5py==3.1.0
numba==0.52.0
numpy==1.17.0
open3d==0.12.0
PyYAML==5.4.1
scikit-learn==0.24.1
scipy==1.5.4
torch==1.6.0+cu101
```

2.2 Dataset Preparation

In the original paper, all experiments were conducted on a reduced dataset called *Campus3D-reduced*, which was derived by voxel sampling the dense *Campus3D* dataset of 10 billion points with a sampling size of 0.15 meter. The file "dataset/preprocessing.py" shows how we downsampled the points and readily provided a label to each sampled point. For convenience, we have already presented the downsampled dataset on the official website <https://3d.dataset.site>. Thus, preprocessing the dataset is not a requirement for running the experiments presented in the original paper.

To prepare the dataset for the reproducibility experiments, the "campus3d_reduced.zip" file should be downloaded from the official website <https://3d.dataset.site> after registering for access with a valid profile. The "campus3d_reduced.zip" file should be placed in the "data" folder under the root directory, with at least 30GB disk space, and unzipped by the following command,

```
$ cd data
$ unzip campus3d_reduced.zip
```

The unzipped "data/campus3d_reduced" folder contains six subfolders named "PGP", "UCC", "FOE", "FASS", "RA" and "YIH" indicating the names of the data acquisition region. Each subfolder has the following structure:

```
<region name>
├── <region name>.pcd
├── <region name>label12.npy
├── <region name>label13.npy
├── <region name>label15.npy
├── <region name>label18.npy
└── <region name>label14.npy
```

where the ".pcd" format file stores the point cloud data and ".npy" format files store the labels for each level of the label hierarchy. We configure the dataset split in the file "data/data_list.yaml", which appoints "FASS", "YIH", "RA", "UCC" as the training set, "PGP" as the validation set and "FOE" as the test set following the split in the original paper.

We note that there is a "data\h_matrices" folder containing the label hierarchical relationship in ".csv" file format. Each ".csv" file stores a binary matrix M_i that projects the one-hot format of label y_i of the current level i into the label y_l of the leaf level,

$$onehot(y_i) = M_i \cdot onehot(y_l), \text{ for } i = 1, 2 \dots, l$$

3 EXPERIMENTS

After the installation of the required environment and preparation of the *Campus3D* dataset, the experiments of the original paper can be conducted. To perform all experiments, the "run.py" is the main file to run with arguments listed in Table.1. All parameters are set in the configuration files "configs\train_setting.yaml" and "configs\dataset.yaml", among which some of the important parameters are listed in Table.1. It is possible to customize the training by tuning the parameters in the configuration file.

3.1 Evaluation and Training Details

3.1.1 Evaluation of trained models. To evaluate the performance of the trained model for hierarchical segmentation tasks on the *Campus3D* dataset, compressed model files can be downloaded from the links in the README.md file from **README.md**. The compressed model in ".zip" format needs to be placed in the "checkpoints" folder and unzipped there. The unzipped folder contains the saved model as "models\model.t7" or "models\model_final.t7" inside. The evaluation can be done by running the command below:

```
$python run.py --exp_name <exp folder>
--eval True --model <framework>
```

where <exp folder> is the name of the unzipped folder in "checkpoints", and <framework> indicates the name of the used deep learning framework. A special case is running a multi-classifier for each level. There are 5 unzipped folders with the sequence number as a suffix including 5 models for each level, i.e., <exp name>_<level>. To run this experiment, the argument --mc_level 0 needs to be added to the command. The evaluation scores and descriptions of trained models can be found under **MODEL_ZOO**.

3.1.2 Train from scratch. After setting the parameters properly, training a model from scratch can be performed by the following command:

```
$python run.py --exp_name <exp folder>
--model <framework>
```

Particularly, to train a model only on one level of labels, an additional argument --mc_level <level> should be appended to the above command, where <level> is the indicated label level from 0 to 4. The model will be stored under folder <exp folder>_<level> in "checkpoint".

3.1.3 Train from pretrained model. As described in the original paper[4], the training process of hierarchical multi-task learning consists of two stages: train without consistency loss (CL) and train with CL. This can be realized by using CL to retrain the model which was pretrained without CL. To load the pretrained

model, one can set the parameter TRAIN.IS_PRETRAINED==TRUE and PRETRAINED_MODEL_PATH as the path to the model in the configuration files and run the above command.

Table 1: Optional argument for executing "run.py"

Argument	Description	Type	Default
exp_name	Name of the selected model	str	"exp"
model	Choice of the model	str	"pointnet2"
eval	Whether to evaluate model	bool	FALSE
mc_level	Label level to use, -1 means all	int	-1

3.2 Experiments of Original Paper

3.2.1 Hierarchical learning. Both the proposed hierarchical learning framework and the comparison model multi-classifier are realized in our package. For the first stage of training a hierarchical learning model, we set TRAIN.CONSISTENCY_LOSS==FALSE in the configuration file and run the command below to train the model in "checkpoints\mt":

```
$python run.py --exp_name mt
```

Secondly, we run the above command with these configurations:

```
TRAIN.CONSISTENCY_LOSS==TRUE
TRAIN.IS_PRETRAINED==TRUE
PRETRAINED_MODEL_PATH=="checkpoint/mt/models
/model.t7"
```

On the other hand, execution of the mc_level command with values from 0 to 4 will train multiple classifiers with a single level of labels as described in Section 3.1.2.

3.2.2 Benchmark result. The semantic segmentation benchmarks can be calculated using three commonly-used models, and they are all realized by PyTorch in our package. To perform the training of multiple models on the dataset, the following command can be executed as <framework> indicating the model name:

```
$python run.py --exp_name mt_<framework>
--model <framework>
```

Table 2: Semantic segmentation benchmarks (mIoU% and OA%) for three feature learning models with MT+HE

Dataset	Learning Model	Granularity level				
		C^1	C^2	C^3	C^4	C^5
OA%	PointNet++	90.5	86.3	78.8	83.7	71.3
	PointCNN	89.6	79.5	78.8	76.9	65.1
	DGCNN	93.7	88.6	82.4	80.5	74.7
mIoU%	PointNet++	82.3	72.3	55.6	34.4	19.8
	PointCNN	81.0	62.1	48.1	29.6	17.5
	DGCNN	87.9	76.8	57.6	39.7	27.1

Table 3: Runtime of our script with different configurations. The alternative runing configurations compared to original one (A) are 4 GPUs instead of single GPU (B), NVIDIA TitanV GPU instead of 2080Ti (C), E5-2630 CPU instead of E5-2680 (D), CUDA version 11.1 instead of 10.1 (E) and Python 3.8 with its latest packages instead of Python 3.7 (F). The train time and eval time indicate the proposed train-from-scratch and evaluation steps, respectively.

Configuration	PointNet2		PointCNN		DGCNN	
	Train (min)	Eval (min)	Train (min)	Eval (min)	Train (min)	Eval (min)
A (Original)	1856	7	4478	22	511	6
B	1259	7	3780	22	227	6
C	3265	11	3073	17	478	5
D	1901	9	4625	19	502	5
E	1765	7	4377	20	498	5
F	1832	8	4592	19	532	5

3.3 Runtime Benchmarks

We report the runtime of our code in different running environments. To be specific, we tested three models with five different variants of the operation environment in hardware or software and present the runtime of them in Table 3. As for hardware, we ran the experiments with the number of GPU(s) set to one or four, and different types of GPUs (NVIDIA TitanV and 2080Ti) and CPUs (E5-2630 and E5-2680) were adopted respectively. As for software, we tested the latest CUDA version 11.1 and Python 3.8 for conducting experiment. All experiments went on smoothly, and the runtimes of the experiments were recorded.

It can be seen that the impact on training speed is mainly due to hardware and model selection, while the software has little impact. This benchmark reflects the good reproducibility and compatibility of our code implementation.

4 CONTRIBUTION AND RESULTS

4.1 Reproducibility Contribution

In the original paper [4], we provided a repository [2]. The repository in this paper [6] is developed from the original one, which contains the instructions and code required for reproduction. In this package, we not only reimplement the original experiments, but also vary the software and hardware environments, such as the version and number of GPUs. All experiments run well, suggesting that our work is highly reproducible, and the results are robust and convincing. The reproduced benchmarks are presented in Table 2, which can serve as a baseline for future work in the field of 3D scene understanding. Other results and links can be found in the **README.md** file.

We should stress that the replication release fixes some problems occurred in Campus3D. First, Campus3D was initially implemented using Tensorflow. The dependency was complex, and some precompiling operations were required. So it was quite painstaking to start the experiments. In pursuit of simplicity, we switch to PyTorch for its built-in packages and ready-made modules. In this way, there is no need to precompile and the environment configuration becomes much handier. Second, in the original paper, we only used the model PointNet++. In order to amplify our package, we add two models (PointCNN and DGCNN). For each model, we run the whole set of experiments, making the result a more comprehensive benchmark. What's more, in Campus 3D, the reduced dataset was generated by sparse voxel sampling. When sampling the dataset, we found it to

be high memory-consuming. To address this problem, we replaced it with KDtree in scikit-learn. As a result, we cut down the memory usage. An additional merit is that we only need to build the KDtree of the dataset once for all. For later use, the built KDtree can be readily loaded through ".pkl" file. Last, there are redundant libraries and code in the original repository. We remove them and refactor the code structure so as to build a more refined package for reproduction and further work.

4.2 Reviewing Process

Two co-authors of this paper are reviewers of this companion paper. Their suggestion improved the reproducibility work in terms of efficiency and accuracy. And we would like to stress the following improvements made during the reviewing process.

On the one hand, there existed several flaws in the proposed codes and documents. First, the reviewers noted that some package requirements were not appropriately specified in the environment setting. And the authors swiftly verified and changed the needed package versions. Second, the reviewers reported the difficulty of loading the pretrained models. The authors corrected the necessary functions to resolve the issue. Finally, certain setup difficulties were discovered by the reviewers, and the authors revised the **README.md** file in response to the comments. On the other hand, the reviewers identified errors and incorrect phrases in the original article and generously gave detailed correction suggestions. The authors checked and polished the work meticulously.

In conclusion, the two reviewers and the authors shared joint efforts for this companion paper. The revised code and paper now not only enable third-party researchers to reproduce the experiments, but also facilitate further research on 3D deep learning.

5 CONCLUSION

In this paper, we present a package for the replication of ACM Multimedia 2020 paper "Campus3D: A Photogrammetry Point Cloud Benchmark for Outdoor Scene Hierarchical Understanding". We elaborate on the steps from scratch to final results in implementing the proposed package, including software installation, dataset downloading, and command execution. This reproduced package is fully open-sourced and handy to utilize, which is expected to support the reproduced benchmarks further. The results demonstrate the effectiveness of the proposed hierarchical learning framework and serve as a baseline for the Campus3D dataset.

REFERENCES

- [1] Austin J. Garrett. 2018. PointCNN. <https://github.com/agarret7/PointCNN>.
- [2] Xinke Li. 2020. Campus3D: A photogrammetry point cloud benchmark for hierarchical understanding of outdoor scene. <https://github.com/shinke-li/Campus3D>.
- [3] Xinke Li, Chongshou Li, Zekun Tong, and Andrew Lim. 2021. Campus3D Dataset. <https://3d.dataset.site>.
- [4] Xinke Li, Chongshou Li, Zekun Tong, Andrew Lim, Junsong Yuan, Yuwei Wu, Jing Tang, and Raymond Huang. 2020. Campus3d: A photogrammetry point cloud benchmark for hierarchical understanding of outdoor scene. In *Proceedings of the 28th ACM International Conference on Multimedia*. 238–246.
- [5] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. 2018. PointCNN: Convolution on χ -transformed points. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 828–838.
- [6] Yuqing Liao and Xinke Li. 2021. Reproducibility of "Campus3D:A Photogrammetry Point Cloud Benchmark for Outdoor Scene Hierarchical Understanding". <https://github.com/Yuqing-Liao/reproduce-campus3d>.
- [7] Kate Anderson Nicholas Hull and Nir Benty. 2017. NVIDIA Emerald Square, Open Research Content Archive (ORCA). <http://developer.nvidia.com/orca/nvidia-emerald-square>
- [8] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. 2020. CUDA, release: 10.2.89. <https://developer.nvidia.com/cuda-toolkit>
- [9] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *Advances in Neural Information Processing Systems* 30 (2017).
- [10] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. 2019. Dynamic graph CNN for learning on point clouds. *Acm Transactions On Graphics (tog)* 38, 5 (2019), 1–12.
- [11] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. 2019. Dynamic Graph CNN for Learning on Point Clouds. *ACM Transactions on Graphics (TOG)* (2019).
- [12] Xu Yan. 2019. Pointnet/Pointnet++ Pytorch. https://github.com/yanx27/Pointnet_Pointnet2_pytorch