Masked Autoencoder for Self-Supervised Pre-training on Lidar Point Clouds

Marzieh Farahani
Eye4AI, AI Sweden

April 14, 2023

Duration of the Project: 6 months
Department: Technology and collaboration-Zenseact
Contact Information:
LinkedIn
Work-email
marzieh.farahani@ai.se
Personal-email
marziehphi@gmail.com

Contents

1 ZoD Dataset For 3D Object Detection

1.1 Installation Process Locally

1.1.1 Install Python

PyTorch requires Python to be installed on your system. You can download and install Python from the official website here. In our Project, we used Python version 3.7.10 which works perfectly with Software Development Kit for the latest Zenseact Open Dataset (ZOD).

```
conda create --prefix py37 python==3.7.10
conda activate py37
```

1.1.2 Install PyTorch And Other Requirements

You can install PyTorch using pip, which is a package manager for Python. Open a terminal or command prompt and run the following command to install PyTorch. This command will install PyTorch along with its dependencies.

```
pip install torch==1.9.0+cu111 torchvision==0.10.0+cu111 torchaudio==0.9.0 -f
    https://download.pytorch.org/whl/torch_stable.html
```

After installation, you can verify that PyTorch is working correctly by opening a Python shell or a Jupyter notebook and running the following code.

```
import torch

# create a tensor
x = torch.Tensor([1, 2, 3, 4])

# print the tensor
print(x)
```

Then create a Bash script that installs requirements for MMdetection3D base.

```
#!/bin/env bash

pip install mmcv-full==1.4.0 -f
    https://download.openmmlab.com/mmcv/dist/cu111/torch1.9.0/index.html
pip install mmdet==2.14.0
pip install mmsegmentation==0.14.1
pip install -r requirements/build.txt
pip install --no-cache-dir -e .
pip install ipdb
pip install numba==0.48
pip install h5py dataclass_wizard rich
pip install zod==0.1.0
pip uninstall pycocotools --no-cache-dir -y
pip install mmpycocotools --no-cache-dir --force --no-deps

# optional
pip install jupyterlab
```

If there is an error with the GPG key and the Cuda home cannot be found. please check HERE.

1.2 Before Preparation Dataset

You can download ZoD 3D detection data here, and unzip all zip files.

Like the general way to prepare dataset, it is recommended to symlink the dataset root to $zod-mae/data$. The folder structure should be organized as follows before processing.

```
zod-mae
├── mmdet3d
├── tools
├── configs
├── data
│   ├── zod
│   │   ├── single_frames
│   │   ├── trainval_frames_mini.json
│   │   ├── trainval_frames_full.json
```

Figure 1: The folder structure before processing

From the mmcv registry documentation. we need to add ZoD dataset to the MMCV dataset registry. So we need to:

- Create a build method.
- Create a registry.
- Use this registry to manage the modules.

Note that you can easily make all necessary changes by using the development kit(pip install zod) from here.

1.3 After Preparation Dataset

We typically need to organize the useful data information with a .pkl or .json file in a specific style, e.g., coco-style for organizing images and their annotations. The folder structure after processing for the ZoD should be as below.

```
zod_mae
├── mmdet3d
├── tools
├── configs
├── data
│   ├── zod
│   │   ├── single_frames
│   │   ├── trainval_frames_mini.json
│   │   ├── trainval_frames_full.json
│   │   ├── zod_gt_database
│   │   ├── zod_infos_train.pkl
│   │   ├── zod_infos_val.pkl
│   │   ├── zod_db_infos_train.pkl
│   │   ├── zod_infos_train_mono3d.coco.json
│   │   ├── zod_infos_val_mono3d.coco.json
```

Figure 2: The folder structure after processing

After finishing this stage. To be sure the preparation was successful, simply run this line of code.

```
python tools/misc/browse_dataset.py configs/_base_/datasets/zod_3d.py --task det
    --output-dir ./data/vis
```

The ZoD compared to the NuScence has no information related to the number of LiDAR points, number of radar points, and ground truth velocity.

1.4 Code Modifications

1.4.1 datasets

In the path mmdet3d/datasets/builder.py. We need to create a builder to interact with data through the config file. Pipeline and object sampler is also defined in this section.

In the path mmdet3d/datasets. We need to create an API class both for 3D and 2D on ZOD frames. You can find them as zod-frames.py, and zod-frames-mono.py.

In the path mmdet3d/datasets/pipelines. We need a small modification on compose file in order to use the pipeline from mmdet3d.

In the path mmdet3d/datasets/pipelines. We need to change the loading file based on the new pipeline registry from ZOD.

1.4.2 Post-processing stage

The box3d-multiclass-nms function performs non-maximum suppression for the input bounding boxes considering the per-box score and overlaps. It returns the indices of the selected boxes. All the adjustments can be found in path mmdet3d/core/post-processing.py

1.4.3 Wrapper function

The wrapper function for data type agnostic processing needs a small modification to run perfectly on the new dataset ZOD.

1.4.4 Anchor Generator Range

Modify the 3D Anchor Generator by range. The anchor generates based on the given range in different feature levels. Different anchor sizes are related to different ranges for different categories. However, we find that sitting does not affect performance much.

1.4.5 Anchor Generator Range

Modify the 3D Anchor Generator by range. The anchor generates based on the given range in different feature levels. Different anchor sizes are related to different ranges for different categories. However, we find that sitting does not affect performance much.

1.4.6 Dimension of boxes in Reconstruction Head

Modify the dimension of boxes to be encoded in box regression transformation deltas that can be used to transform the “src-boxes” into the “target-boxes”. all the changes could be found in sst-base.py.

1.4.7 Test Inference

for getting the prediction and ground truth result in order to visualize the performance you need to modify the test.py in path mmdet3d/apis.