Installation

We recommend installing all dependencies in a virtual environment.

```
conda create -n iccv python=3.8
conda install cudatoolkit=11.7 -c pytorch -c conda-forge
pip install torch==1.13.0 torchvision==0.14.0 torchaudio==0.13.0 spconv-cu117

# install dependencies
pip install -r requirements.txt
python setup.py develop

# build nms cuda version
python opencood/utils/setup.py build_ext --inplace
```

Dataset

1. Please refer to HEAL to prepare the dataset OPV2V and OPV2V_H
2. Prepare OPV2V_few_shot and OPV2V_Heter_few_shot dataset following the method below.

```
  # split by scenario.
  python opencood/tools/split.py --data_dir ${OPV2V_DIR} --out_dir ${OPV2V_few_shot}

  # link opv2v_heter_few_shot to opv2v_h
  ln -s ${OPV2V_Hetero} ${OPV2V_Hetero_few_shot}
```

3. Prepare another few-shot dataset OPV2V_few_shot_no_first, this dataset is the same as OPV2V_few_shot. The only difference is that we have removed the first folder in each scenario. Because this dataset is used for training the adapter, and we do not want to leak the data of m1 here. The deletion was done manually.
4. Make sure the `dataset` folder points to your data folder to ensure that the `dataset/OPV2V_few_shot`, `dataset/OPV2V_Hetero_few_shot` folder is valid.

Train and Evaluation

We have prepared the configuration files for reproducing the 5-shot experiment, there files are stored in the `opencood/logs` directory.

1. Train the base collaborative models for m1 and m3 separately.

```
# train m1 based model
python opencood/tools/train.py -y None --model_dir opencood/logs/m1

# train m3 based model
python opencood/tools/train.py -y None --model_dir opencood/logs/m3
```

2. Regard m1 as the ego and m3 as the agent. We merge the encoder of m3 into m1 base model.

```
python opencood/tools/heal_tools.py  merge_encoder_and_save_final opencood/logs/m1
opencood/logs/m3 opencood/logs/final
```

3. For each scenario, we can utilize the predictions of m3 as pseudo-labels, with the option to set different confidence scores. The default value is 0.5.

```
cp opencood/logs/m3/net_epochnet_at*.pth opencood/logs/final/fake_label
python opencood/tools/generate_fake_label.py --model_dir opencood/logs/final/fake_label --
score 0.5
cp opencood/logs/final/fake_label/fake_label.pkl opencood/logs/final/fsl_train
```

4. For each scenario, train the adapter of m1 separately using few-shot finetuning.

```
cp opencood/logs/m1/net_epochnet_at*.pth opencood/logs/final/fsl_train
python opencood/tools/few_shot_train.py --model_dir opencood/logs/final/fsl_train
```

5. For each scenario, merge the adapter into the m1 base model and perform collaborative perception inference together with m3.

```
python opencood/tools/batch_merge_and_infer.py --base_dir opencood/logs/final/ --vis
```

After execution is done, you can find the AP of all scenarios as well as the mSAP in `opencood/logs/final/log.txt`. Here is an example.

```
2021_08_23_12_58_19: 0.86, 0.83, 0.67
2021_08_21_09_28_12: 0.97, 0.96, 0.9
2021_08_23_16_06_26: 0.95, 0.95, 0.9
2021_08_20_21_10_24: 0.98, 0.98, 0.91
2021_08_23_17_22_47: 0.96, 0.96, 0.93
2021_08_22_09_08_29: 0.96, 0.96, 0.92
2021_08_24_11_37_54: 0.86, 0.86, 0.84
2021_08_22_21_41_24: 0.96, 0.95, 0.91
2021_08_23_21_07_10: 0.89, 0.89, 0.85
2021_08_23_21_47_19: 0.96, 0.96, 0.89
```

```
2021_08_24_20_49_54: 0.91, 0.89, 0.79
2021_08_23_15_19_19: 0.97, 0.97, 0.95
2021_08_22_07_52_02: 0.95, 0.94, 0.88
2021_08_18_19_48_05: 0.97, 0.96, 0.89
2021_08_24_07_45_41: 0.81, 0.81, 0.68
2021_08_24_20_09_18: 0.91, 0.91, 0.83
average: 0.929, 0.924, 0.859
```