mmlab_extension

Release latest

windzu

Sep 03, 2022

GET STARTED

1 Introduction 1

2 Open-mmlab Introduction 3

3 Environment configuration 5

4 Installation 7

5 Installation 9

6 Installation 11

CHAPTER

ONE

INTRODUCTION

the automatic driving direction will be given priority when the function is implemented.

The `closed loop` of a deep learning task can be simply divided into the following three parts

- Data : Collection, Label, Data Management
- Model : Build Model, Training, Testing, Evaluation
- Deploy : Model Deployment

As the complexity of tasks increases, it becomes more difficult to rely on a single person to complete the whole process of work, and if you rely on the cooperation of multiple people, there will often be "wrangling" between collaborators due to level mismatch and conflicting work content. In response to this situation, the industry proposed the concept of `MLOps`

But at present, according to my research, there is no mature `MLOps` solution, and many open source tools have many restrictions, so before there are no mature and useful tools, you can only use open source tools to simply piece together one and use it. ~

This project relies on a series of tools in open-mmlab to complete the modeling and deployment work in the MLOps task, because the main used mmdeploy and mmdetection3d are still there Quick updates, so engineers who want to use this project, please read the relevant documents and update comments, and hope that everyone can cooperate and communicate, mention more issues and pr, and jointly create a work that is easy to use, and set aside some time. fish

CHAPTER

TWO

OPEN-MMLAB INTRODUCTION

> Because this project is built on the basis of openmmlab with some minor modifications, the premise of using this project is to have a certain understanding of the related work of open-mmlab. The following is the relevant introduction

The series of frameworks in open-mmlab have dependencies, and their essence is the inheritance performance of OO (object-oriented).

If you want to quickly understand open-mmlab, you must first understand the `factory design mode`, which is implemented through registry in open-mmlab. For details, please refer to: registry details or is the corresponding code (the code is very short, it is recommended to look at it)

basic introduction

- mmcvBase library in open-mmlab

  The basic functions in the other frameworks of open-mmlab are derived from it or inherited from it

- mmdetectionAn integrated framework for 2D detection and segmentation

  Provide landmark model support in this field, as well as other common model support with good reputation, such as RCNN series, yolo series, etc. The update speed is generally about half a year later than that of academia

- mmdetection3dAn integrated framework for 3D detection and segmentation

  It is modified on the basis of mmdetection. After all, many 3d detection methods are actually 2d methods. The update speed is generally about half a year later than that of academia, but because of the popularity of automatic driving and the 2D track rolling Everyone is turning to 3D, so its recent update is relatively "rapid"

- mmdeploya deployment framework

  The purpose of its appearance is to get through the last step of the project landing in open-mmlab, and its update speed is the slowest, generally a very mature model will be supported (in other words, if the support speed is fast, then everyone will be unemployed.) Each supported model generally supports a variety of backends, at least onnx and tensorrt have not yet reached the official version of v1.0. For the supported models, some will provide related sdk intimately, so that the front and back are connected. You don't need to write it yourself.

CHAPTER

THREE

ENVIRONMENT CONFIGURATION

For the convenience of subsequent use, it is best to prepare the necessary configuration of checkpoints, data and environment variables according to the following requirements

3.1 Prepare Data

Prepare the data required for training and testing, pre-trained checkpoints, and place them according to the specified structure for subsequent configuration data symbolic links

3.1.1 Checkpoints

The file structure is as follows

```
mmlab_checkpoints
├── mmdetection
│   └── checkpoints
└── mmdetection3d
    └── checkpoints
```

3.1.2 Data

The file structure is as follows

```
mmlab_dataset
├── mmdetection
│   └── data
└── mmdetection3d
    └── data
```

3.2 Pull Project

In order to reduce the amount of data, generally only pull the latest commit

```
git clone https://github.com/windzu/mmlab_extension.git --depth 1 && \
cd mmlab_extension && \
git submodule update --init --recursive
```

3.3 Add Environment Variables

based self shell to choose .bashrc or .zshrc

```
cd mmlab_extension && \
echo "export MMLAB_EXTENSION_PATH=$(pwd)" >> ~/.bashrc

cd mmlab_checkpoints && \
echo "export MMLAB_CHECKPOINTS_PATH=$(pwd)" >> ~/.bashrc

cd mmlab_dataset && \
echo "export MMLAB_DATASET_PATH=$(pwd)" >> ~/.bashrc
```

3.4 Configure Data Symbolic Link

```
# checkpoints
rm -rf $MMLAB_EXTENSION_PATH/mmdetection/checkpoints && \
ln -s $MMLAB_CHECKPOINTS_PATH/mmdetection/checkpoints $MMLAB_EXTENSION_PATH/mmdetection/
↪checkpoints && \
rm -rf $MMLAB_EXTENSION_PATH/mmdetection3d/checkpoints && \
ln -s $MMLAB_CHECKPOINTS_PATH/mmdetection3d/checkpoints $MMLAB_EXTENSION_PATH/
↪mmdetection3d/checkpoints

# dataset
rm -rf $MMLAB_EXTENSION_PATH/mmdetection/data  && \
ln -s $MMLAB_DATASET_PATH/mmdetection/data $MMLAB_EXTENSION_PATH/mmdetection/data && \
rm -rf $MMLAB_EXTENSION_PATH/mmdetection3d/data  && \
ln -s $MMLAB_DATASET_PATH/mmdetection3d/data $MMLAB_EXTENSION_PATH/mmdetection3d/data
```

CHAPTER

FOUR

INSTALLATION

At present, only the installation method of conda is provided

The cuda operator compilation in mmcv depends on the cuda installed by the host, so the installation is slightly different under different cuda versions. The following provides examples of ubuntu18.04+cuda10.2 and ubuntu20.04+cuda11.3

4.1 ubuntu18.04 cuda10.2

```
# To be added
```

4.2 ubuntu20.04 cuda11.3

```
export CONDA_ENV_NAME=mmdet && \
export PYTHON_VERSION=3.8 && \
export CUDA_VERSION=11.3 && \
export MMCV_VERSION=1.6.0 && \
export MMCV_CUDA_VERSION=113 && \
export TORCH_VERSION=1.12.0 && \
conda create -n $CONDA_ENV_NAME python=$PYTHON_VERSION -y && \
conda activate $CONDA_ENV_NAME && \
conda install pytorch=$TORCH_VERSION torchvision torchaudio cudatoolkit=$CUDA_VERSION -c␣
↪pytorch -y && \
pip install openmim && \
mim install mmcv-full==$MMCV_VERSION && \
cd $MMLAB_EXTENSION_PATH/mmdetection && \
pip install -e . && \
cd $MMLAB_EXTENSION_PATH/mmdetection_extension && \
pip install -e .
```

CHAPTER

FIVE

INSTALLATION

At present, only the installation method of conda is provided

The cuda operator compilation in mmcv depends on the cuda installed by the host, so the installation is slightly different under different cuda versions. The following provides examples of ubuntu18.04+cuda10.2 and ubuntu20.04+cuda11.3

5.1 ubuntu18.04 cuda10.2

```
export CONDA_ENV_NAME=mmdet3d && \
export PYTHON_VERSION=3.8 && \
export TORCH_VERSION=1.12.0 && \
export CUDA_VERSION=10.2 && \
export MMCV_CUDA_VERSION=102 && \
export MMCV_VERSION=1.6.0 && \
export MMDET_VERSION=2.25.0 && \
conda create -n $CONDA_ENV_NAME python=$PYTHON_VERSION -y && \
conda activate $CONDA_ENV_NAME && \
conda install pytorch=$TORCH_VERSION torchvision torchaudio cudatoolkit=$CUDA_VERSION -c␣
↪pytorch -y && \
cd $MMLAB_EXTENSION_PATH/mmcv && git checkout v$MMCV_VERSION && \
MMCV_WITH_OPS=1 pip install -e . && \
cd $MMLAB_EXTENSION_PATH/mmdetection && git checkout v$MMDET_VERSION && \
pip install -e . && \
pip install openmim && \
mim install mmsegmentation && \
cd $MMLAB_EXTENSION_PATH/mmdetection3d && \
pip install -e . && \
cd $MMLAB_EXTENSION_PATH/mmdetection3d_extension && \
pip install -e . && \
python3 -m pip install --user git+https://gitee.com/windzu/pypcd.git
```

5.2 ubuntu20.04 cuda11.3

```
export CONDA_ENV_NAME=mmdet3d && \
export PYTHON_VERSION=3.8 && \
export CUDA_VERSION=11.3 && \
export MMCV_VERSION=1.6.0 && \
export MMDET_VERSION=2.25.0 && \
export MMCV_CUDA_VERSION=113 && \
export TORCH_VERSION=1.12.0 && \
conda create -n $CONDA_ENV_NAME python=$PYTHON_VERSION -y && \
conda activate $CONDA_ENV_NAME && \
conda install pytorch=$TORCH_VERSION torchvision torchaudio cudatoolkit=$CUDA_VERSION -c␣
↪pytorch -y && \
cd $MMLAB_EXTENSION_PATH/mmcv && git checkout v$MMCV_VERSION && \
MMCV_WITH_OPS=1 pip install -e . && \
cd $MMLAB_EXTENSION_PATH/mmdetection && git checkout v$MMDET_VERSION && \
pip install -e . && \
pip install openmim && \
mim install mmsegmentation && \
cd $MMLAB_EXTENSION_PATH/mmdetection3d && \
pip install -e . && \
cd $MMLAB_EXTENSION_PATH/mmdetection3d_extension && \
pip install -e . && \
python3 -m pip install --user git+https://gitee.com/windzu/pypcd.git
```

CHAPTER

SIX

INSTALLATION

To be added