3D LUT

Tips: the project is based on Ubuntu 22.04.

Dependencies

Miniconda

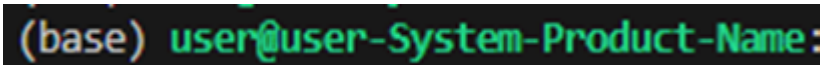
Use the following commands to install Miniconda:

```
mkdir -p ~/miniconda3
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh -O
~/miniconda3/miniconda.sh
bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3
rm -rf ~/miniconda3/miniconda.sh
```

After installing, use the following commands to initialize your newly-installed Miniconda:

```
~/miniconda3/bin/conda init bash
~/miniconda3/bin/conda init zsh
```

Restart a terminal. If Miniconda is installed successfully, '(base)' will appear in front of the username as shown below:



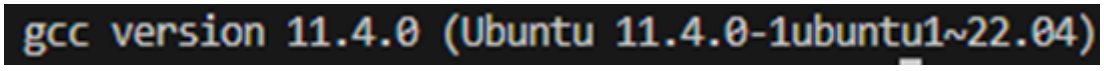
GCC/C++

Use the following commands to install GCC/C++:

```
sudo apt update
sudo apt install gcc g++
```

Use the following command to check if GCC is installed on your system. The last line of the result indicates the GCC version, as shown in the following figure:

```
gcc -v
```



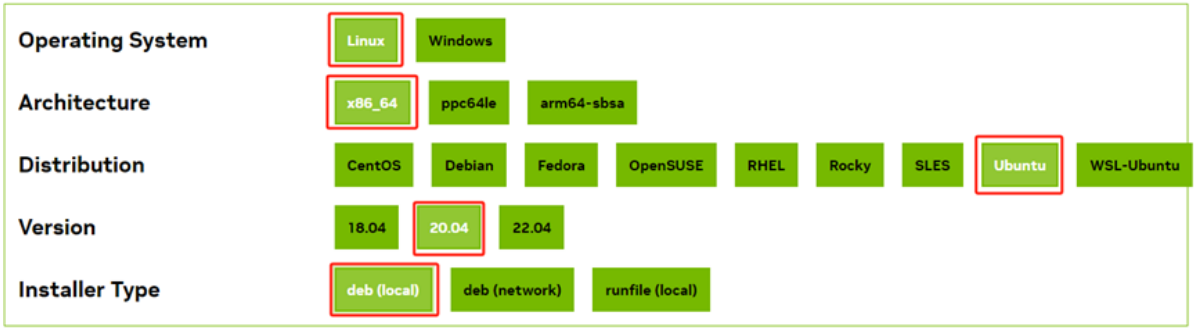
CUDA

Use the following command to check the highest version of CUDA supported by your system, as shown in the following figure:

```
nvidia-smi
```

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 535.161.07             Driver Version: 535.161.07   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  NVIDIA GeForce RTX 4090       Off | 00000000:31:00.0 Off |                  Off |
| 30%   49C    P2             179W / 450W |  24095MiB / 24564MiB |     15%      Default |
|                                         |                      |                  N/A |
```

Go to the CUDA official website (https://developer.nvidia.com/cuda-toolkit-archive) and choose the appropriate CUDA version (the CUDA version must be higher than the version of CUDA in the subsequently created virtual environment), then select multiple options as shown below:



After selection, follow the Installation Instructions given at the bottom of the website to install CUDA.

Use the following commands to add environment variables (Replace 'your_cuda_folder' with the actual CUDA folder name on your system):

```
export PATH=/usr/local/your_cuda_folder/bin${PATH:+:${PATH}}
export
LD_LIBRARY_PATH=/usr/local/your_cuda_folder/lib64${LD_LIBRARY_PATH:+:${LD_LIBRARY
_PATH}}
source ~/.bashrc
```

Use the following command to check if CUDA is installed on your system. The result indicates the GCC version, as shown in the following figure:

```
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2023 NVIDIA Corporation
Built on Tue_Feb__7_19:32:13_PST_2023
Cuda compilation tools, release 12.1, V12.1.66
Build cuda_12.1.r12.1/compiler.32415258_0
```

Environment

Create a virtual Environment

You should run the project in a suitable virtual environment. Use the following command to create a virtual environment with Python 3.9.0:

```
conda create -n LUT python=3.9
```

Use the following command to activate the created virtual environment:

```
conda activate LUT
```

Use the following command to install PyTorch 1.11.0 after activation:

```
pip install torch==1.11.0+cu113 torchvision==0.12.0+cu113 torchaudio==0.11.0 --
extra-index-url https://download.pytorch.org/whl/cu113
```

If you need to exit the virtual environment, use the following command:

```
conda deactivate
```

Install libraries

First, you should make sure the current directory is the root of the project.

Use the following command to install the required libraries for the project:

```
pip install -r requirements.txt
```

The proposed LUT Transform is implemented as a PyTorch CUDA extension. You should install the extension in the following way:

```
python models/LUT/ailut_transform/setup.py install
```

Usage

Datasets

You can download the LOL datasets from the baidu pan (code: vvdp).

If you want to use your own custom dataset, you should note that the dataset used for this model is required to have a folder organization similar to the following:

```
dataset_name/
    train/
        input/
            image0001.png # PNG is just an example, it does not mean that the
format has to be PNG.
            image0002.png
            ...
```

```
            image1234.png
        gt/
            image0001.png
            image0002.png
            ...
            image1234.png
    test/
        input/
            image0001.png
            image0002.png
            ...
            image0050.png
        gt/
            image0001.png
            image0002.png
            ...
            image0050.png
```

Settings

All configuration YML files of training are in `./options/train/`. The configuration file of training for the model has the following structure, and sections that can be customized are followed by comments:

```
name: LUT_lol_syn_rank3 # The name of this training
use_tb_logger: true
model: image_base
distortion: llie
gpu_ids: [0] # The id of the GPU used for this training. Multiple GPUs can be
used.

datasets:
  train:
    name: train
    mode: LOL
    interval_list: [1]
    random_reverse: false
    border_mode: false
    dataroot_GT: /home/user/Datasets/LOL-v2/Synthetic/Train/Normal # Path of the
ground truths of your training dataset
    dataroot_LQ: /home/user/Datasets/LOL-v2/Synthetic/Train/Low # Path of the
input images of your training dataset
    cache_keys: ~
    cache_data: true

    use_shuffle: true
    n_workers: 4
    batch_size: 8
    GT_size: 256
    LQ_size: 256
    use_flip: true
    use_rot: true
  val:
    name: val
    mode: LOL
```

```
    dataroot_GT: /home/user/Datasets/LOL-v2/Synthetic/Test/Normal # Path of the
ground truths of your validation dataset
    dataroot_LQ: /home/user/Datasets/LOL-v2/Synthetic/Test/Low # Path of the
input images of your validation dataset
    cache_data: true

network_G:
  which_model_G: LUT_LLIE
  if_train: false
  n_ranks: 3 # The number of basis LUTs in the model
  n_vertices_3d: 33 # The number of sampling points on each dimension of each LUT
in the model
  n_base_feats: 8
  smooth_factor: 0
  monotonicity_factor: 10

path:
  root: ./
  ## pretrain_model_G: experiments/LUT_lol_syn_rank3/models/xxxx_G.pth
  ## resume_state: experiments/LUT_lol_syn_rank3/training_state/xxxx.state
  strict_load: false

train:
  lr_G: !!float 1e-4
  lr_scheme: CosineAnnealingLR_Restart
  beta1: 0.9
  beta2: 0.999
  niter: 600000
  sparse_factor: 0.0001
  warmup_iter: -1
  T_period: [50000, 100000, 150000, 150000, 150000]
  restarts: [50000, 150000, 300000, 450000]
  restart_weights: [1, 1, 1, 1]
  eta_min: !!float 1e-7

  pixel_criterion: Charbonnier
  val_freq: !!float 5e3

  manual_seed: 100

logger:
  print_freq: 100
  save_checkpoint_freq: !!float 5000
```

All configuration YML files of testing are in `./options/test/`. The configuration file of testing for the model has the following structure, and sections that can be customized are followed by comments:

```
name: LUT_lol_syn_rank3 # The name of this test
model: image_base
distortion: llie
gpu_ids: [0] # The id of the GPU used for this test. Multiple GPUs can be used.

datasets:
  test:
```

```
    name: test
    mode: LOL
    dataroot_GT: /home/user/Datasets/LOL-v2/Synthetic/Test/Normal # Path of the
ground truths of your testing dataset
    dataroot_LQ: /home/user/Datasets/LOL-v2/Synthetic/Test/Low # Path of the
input images of your testing dataset
    cache_data: true

network_G:
  which_model_G: LUT_LLIE
  if_train: false
  n_ranks: 3 # The number of basis LUTs in the trained model.
  n_vertices_3d: 33 # The number of sampling points on each dimension of each LUT
in the trained model
  n_base_feats: 8
  smooth_factor: 0
  monotonicity_factor: 10

path:
  root: ./
  pretrain_model_G: pretrain/LUT3D_syn_rank3.pth # The path of the trained model
for the test
```

Training

Before starting the training, you must change the configuration file of training according to the path of dataset on your system.

The training on the synthetic dataset of LOL-v2:

```
python train.py -opt options/train/train_lol.yml
```

The training on the real dataset of LOL-v2:

```
python train.py -opt options/train/train_lol_real.yml
```

The training on other dataset (You should create new configuration files for other datasets):

```
python train.py -opt options/train/xxxx.yml
```

Testing

Before starting the evaluation, you must change the configuration file of testing according to the path of dataset on your system.

The test on the synthetic dataset of LOL-v2:

```
python quantitative_test.py -opt options/test/test_lol.yml
python qualitative_test.py -opt options/test/test_lol.yml
```

The test on the real dataset of LOL-v2:

```
python quantitative_test.py -opt options/test/test_lol_real.yml
python qualitative_test.py -opt options/test/test_lol_real.yml
```

The test on other dataset (You should create new configuration files for other datasets):

```
python quantitative_test.py -opt options/test/xxxx.yml
python qualitative_test.py -opt options/test/xxxx.yml
```

License