Moving Haar Learning Rate Scheduler for Large-scale Face Recognition Training with One GPU

Moving Haar Learning Rate (MHLR) scheduler is a learning rate scheduler dedicated to large-scale face recognition training with 1xGPU. It is able to accelerate the model to 1/4 of its original training time without sacrificing more than 1% accuracy. For example, on the dataset WebFace12M containing more than 12M face images with 0.6M identities, MHLR only requires 30 hours to train the model ResNet100. With MHLR, scholars lacking expensive hardware resources and massive computing power can also do large-scale face recognition research now.

Acknowledgement

We would like to express our sincere thanks to InsightFace[1]. MHLR is modified and developed based on arcface_pytorch[2] in InsightFace. Hereby, we advocate more people to develop such reproducible source code in order to stimulate more future works in the face recognition direction.

[1] Jia Guo, Jiankang Deng, Xiang An, Jack Yu, and Baris Gecer, InsightFace: https://github.com/deepinsight/insightface, 2023.

[2] Arcface_repository: https://github.com/deepinsight/insightface/tree/master/recognition/arcface_torch, 2023.

Some differences corresponding to arcface_pytorch

- Arface_pytorch needs to install MXNet in order to read datasets from ".rec" files. The way to install the environment is in the document `./utils/INSTALL_MXNET.md`. However, it took us a long time to configure the developing environment. Thus, for simplicity and reproducibility, we converted the ".rec" files into ".jpg" images, though the huge amount of images will lead to that too many small files on the hard disk. In addition, the dataset WebFace42M contains ".jpg" images instead of ".rec" files. Therefore, we select ".jpg" images as our training data.
- The implementation of arcface_pytorch aims at distributed training, while MHLR aims at single GPU training. Therefore, we mainly revised the multi-nodes with multi-GPUs code in arcface_pytorch to single GPU version in MHLR. However, Partial FC (PFC) have not successfully modified yet. Hence, training models on WebFace42M is our future work, since the 1xGPU memory space without PFC is not enough for it.

Requirements

- NVidia Geforce RTX 4090.
  - We also tried to run on NVidia Geforce RTX 3090, which is a little slower yet feasible for large-scale face recognition training
  - We also tried to run on NVidia Tesla A800, which is a little faster yet no big difference
  - To the best of our knowledge, 3090 is not the very basic requirement. Although we did not try it yet, we can confidently claim that training ResNet100 on MS1MV3 in 1 day with even lower level GPU must be possible.
- The operating system could be either Windows, or Linux (Ubuntu and CentOS).

- (Optional) Install Anaconda, we employ Anaconda to manage the developing environment.
- Install PyTorch.
- Install the depending packages using the command: `pip install -r requirement.txt`.

Datasets

Since we are not very clear with the copyright regulations, thus we do not publish any dataset by ourselves. Instead, the links where we download our datasets are given below:

- MS1MV2 (87k IDs, 5.8M images)
  - Need to convert from ".rec" files to ".jpg" images. We put our converting file in `./utils/rec2img.py`. Yet, it requires installing MXNet to read ".rec" files and we believe it will take some time for the deployment. The way to install MXNet is in the document `./utils/INSTALL_MXNET.md`
  - This is employed as a training dataset. Yet, comparing to MS1MV3, it has more images yet a lower accuracy of trained models. We recommend to train models on MS1MV3 instead.
- MS1MV3 (93k IDs, 5.2M images)
  - Need to convert from ".rec" files to ".jpg" images. We put our converting file in `./utils/rec2img.py`. Yet, it requires installing MXNet to read ".rec" files and we believe it will take some time for the deployment. The way to install MXNet is in the document `./utils/INSTALL_MXNET.md`
  - This is employed as a training dataset. To the best of our knowledge, MS1MV3 is a good training dataset if lacking of enough hardware resources, since comparing to WebFace42M, it has relatively less images while a satisfied accuracy of trained models.
- WebFace42M (2M IDs, 42.5M images)
  - Need to sign a form and send an email asking for the download link.
  - We sampled 10%, 20%, and 30% of WebFace42M as WebFace4M, WebFace8M, and WebFace12M. Those three datasets are employed as training datasets. More specifically, we unzipped files from "0_0.zip" to "0_6.zip" to get WebFace4M. Moreover, We unzipped extra files from "1_0.zip" to "1_6.zip" to get WebFace8M. Finally, we unzipped extra files from "2_0.zip" to "2_6.zip" to get WebFace12M.
- LFW, CFP-FP, AgeDB-30
  - These three datasets are employed as test datasets. It is downloaded along with MS1MV3.
  - Need to convert from ".bin" files to ".jpg" images. We put our converting file in `./utils/bin2img.py`. Yet, it requires installing MXNet to read ".bin" files and we believe it will take some time for the deployment. The way to install MXNet is in the document `./utils/INSTALL_MXNET.md`
- IJB-B, IJB-C
  - These two datasets are employed as test datasets.

Training

In the folder `./configs`, there are configuration files. Modify the attribute `rec` in any file to your training dataset address. For example, my dataset is located at `/home/user001/Data/WebFace12M`, and then I set `rec="/home/user001/Data/WebFace12M"` in the file `./configs/webface12m.py`. After that, execute the following statement to train the model.

```
python train.py --config webface12m
```

- The package tensorboard should be installed before training

The folder of your training dataset should be organized like the follows:

```
/Dataset_name
├── ID1
│   ├── Image1.jpg
│   ├── Image2.jpg
│   ├── Image3.jpg
│   ├── Image4.jpg
│   └── ...
├── ID2
│   ├── Image1.jpg
│   ├── Image2.jpg
│   └── ...
├── ID3
│   ├── Image1.jpg
│   ├── Image2.jpg
│   ├── Image3.jpg
│   └── ...
└── ...
```

Testing

Suppose we trained a model by the configuration file `webface12m.py`. Modify the attribute `val` in `webface12m.py` to your test dataset address. For example, my dataset is located at `/home/user001/Data/val`, and then I set `val="/home/user001/Data/val"` in the file `./configs/webface12m.py`. After that, execute the following statement to test the model.

Please note that we only did a small amount of revisions to the test files from arcface_pytorch. Therefore, it still exists some bugs and we have given the fixing ways.

For LFW, CFP-FP, and AgeDB-30, execute the following statement:

```
python eval_veri.py --config webface12m
```

- The new version of scipy, for example, the version "1.11.1", will cause a bug reporting "ValueError: Expect x to not have duplicates". Thus, we need to downgrade the version of scipy to the prior version, for instance, the version "1.9.2".
    - The command: `pip install scipy==1.9.2` can downgrade scipy. We plan to fix the bug once for all in the future.
- If you failed to downgrade scipy, we still have another alternative, which is modifying the source code of scipy:

    - At the line 1295 and 1296 in the file `"anaconda3/envs/arcface/lib/python3.11/site-packages/scipy/interpolate/_bsplines.py"`. Comment the statement `if np.any(x[1:] == x[:-1]):` and `raise ValueError("Expect x to not have duplicates")`

For IJB-B, execute the following statement:

```
python eval_ijbb.py --config webface12m
```

- The package opencv-python should be installed.
- The package prettytable should be installed.
- The package menpo should be installed. However, it has a bug reporting "AttributeError: module 'numpy' has no attribute 'float'. Thus, we need to modify the source code of menpo, which is shown as follows:
    - At the line 292 in the file `"anaconda3/lib/python3.11/site-packages/menpo/image/base.py"`. Revise `dtype=np.float` to `dtype=np.float32` in the statement `def init_blank(cls, shape, n_channels=1, fill=0, dtype=np.float):`
    - At the line 384 in the file `"anaconda3/lib/python3.11/site-packages/menpo/image/base.py"`. Revise `dtype=np.float` to `dtype=np.float32` in the statement `dtype=np.float,`
    - At the line 84 in the file `"anaconda3/lib/python3.11/site-packages/menpo/image/masked.py"`. Revise `dtype=np.float` to `dtype=np.float32` in the statement `def init_blank(cls, shape, n_channels=1, fill=0, dtype=np.float, mask=None):`
    - At the line 165 in the file `"anaconda3/lib/python3.11/site-packages/menpo/image/masked.py"`. Revise `dtype=np.float` to `dtype=np.float32` in the statement `dtype=np.float,`
    - The bug is caused by that `np.float` was a deprecated alias for the builtin `float`. To avoid this error in existing code, use `float` by itself. Doing this will not modify any behavior and is safe. If you specifically wanted the numpy scalar type, use `np.float64` here. The aliases was originally deprecated in NumPy 1.20;

For IJB-C, execute the following statement:

```
python eval_ijbc.py --config webface12m
```

- The package opencv-python should be installed.
- The package prettytable should be installed.
- The package menpo should be installed. Please manually fix the bug of menpo as mentioned above. We plan to fix the bug once for all in the future.

The folder of your test datasets should be organized like the follows:

```
/test
├── lfw
```

```
│   ├── imgs
│   │   ├──Image1.jpg
│   │   ├──Image2.jpg
│   │   └──...
│   └── pair_label.npy
├── cfp_fp
│   ├── imgs
│   │   ├──Image1.jpg
│   │   ├──Image2.jpg
│   │   └──...
│   └── pair_label.npy
├── agedb_30
│   ├── imgs
│   │   ├──Image1.jpg
│   │   ├──Image2.jpg
│   │   └──...
│   └── pair_label.npy
├── ijb
│   ├── IJBB
│   │   ├──loose_crop
│   │   │   ├──Image1.jpg
│   │   │   ├──Image2.jpg
│   │   │   └──...
│   │   ├──meta
│   │   └──result
│   ├── IJBC
│   │   ├──loose_crop
│   │   │   ├──Image1.jpg
│   │   │   ├──Image2.jpg
│   │   │   └──...
│   │   ├──meta
└──  └──  └──result
```

Pretrained models

Too big to put it in supplementary materials. Instead, we put our training logs in the Performance section.

Performance

Datasets	Backbone	LFW	CFP-FP	AgeDB-30	IJB-B(1e-4)	IJB-B(1e-4)	log
MS1MV3	ResNet100	99.80	98.53	98.12	95.03	96.41	click me
WebFace4M	ResNet100	99.65	98.63	97.57	94.36	96.18	click me
WebFace8M	ResNet100	99.85	99.24	97.90	95.58	97.09	click me
WebFace12M	ResNet50	99.80	99.09	97.92	95.25	96.85	click me
WebFace12M	ResNet100	99.83	99.17	98.02	95.68	97.20	click me

Future Work

- Implement PFC for training models on WebFace42M with 1×GPU.
- Conduct experiments on Masked Face Recognition (MFR)[1].
- Fix bugs in the test files once for all.

[1] ICCV-2021 Masked Face Recognition Challenge & Workshop(MFR): http://iccv21-mfr.com/, 2023.