Effective Training Strategies for Deep Learning-based Precipitation Nowcasting and Estimation (Software User Guide)

Jihoon Ko*, Kyuhan Lee*, Hyunjin Hwang*, Seok-Geun Oh, Seok-woo Son, Kijung Shin

1 General Information

- Version: 1.0

2 Introduction

In this software, we provided the following implementations in PyTorch:

- Model and Dataset implementation (`model.py` and `dataset.py`): U-Net based model (1x1 resolution) for precipitation nowcasting and estimation, and the dataset for each target task.
- Codes for experiments: Codes for pre-training and fine-tuning the model, and evaluating the performance of the trained model.

Detailed information about the implementations is explained in the following paper:

- Jihoon Ko*, Kyuhan Lee*, Hyunjin Hwang*, Seok-Geun Oh, Seok-Woo Son, and Kijung Shin, "Effective training strategies for deep learning-based precipitation nowcasting and estimation", Under Review

3 Requirements

In order to execute the codes, you need to install following packages.

- NumPy
- PyTorch
- tqdm
- pyProj

4 Input File Format

To run the provided code, you need to prepare radar images and ground-truth rainfall data and then designate each dataset's path.

4.1 Radar Images

The only supported type is *.bin.gz containing an array of 16bit integers. The dBZ of the i-th pixel multiplied by 100 should be stored in the $2i-1$ to $2i$-th bytes of the file. For running experiments, you need to specify `center`, `radius`, and `original_radar_size` of the datasets, which indicates center coordinates of the cropped radar image, half of the size of the radar image, and the original image size before cropping. See `dataset.py` for more details. Our implementation assumes the input image has a size of 1468x1468. You should change the architecture if you want to use the model with different input sizes of images.

4.2 Ground-truth Rainfall Data

The only supported type is *.pkl containing a serialized list of ground-truth rainfall information. Each element in the list should be a timestamp pair with format YYYYMMDDhhmm and the corresponding rainfall information. The rainfall information is a list of tuples consist of coordinates (y-axis and x-axis) of pixels consistent with the coordinates of radar images and the corresponding rainfall information. Here is an example of the format of the list.

[['202006010000', [(Y_1, X_1, V_1), (Y_2, X_2, V_2), $\cdots$], ['202006010100', [(Y_3, X_3, V_3), (Y_4, X_4, V_4), $\cdots$], $\cdots$]

After preparing the data, the last step of processing the dataset is to serialize the data. `pickle.dump(open("sampled.pkl", "wb"), list)` serializes the list and stores the serialized data in `sampled.pkl`.

5 How to Execute

5.1 Pretraining (`pretrain.py`)

- **-h, --help**: Show help message
- **--lr LR**: Set learning rate as LR
- **--n-steps N_STEPS**: Set number of training steps as N_STEPS
- **--batch-size BATCH_SIZE**: Set batch size as BATCH_SIZE
- **--data-path DATA_PATH**: Set path of the radar input images as DATA_PATH
- **--gpus GPUS**: Set which GPUs to use. GPUS should be comma-separated string or single GPU id. If GPUS is not specified, only CPU is used during execution.

5.2 Finetuning (`finetuning-[CE|focal|ours].py` for Nowcasting / `finetuning.py` for Estimation)

- **-h, --help**: Show help message
- **--lr**, **--n-steps**, **--batch-size**, **--data-path**, **--gpus**: See Section 5.1
- **--sampled-path SAMPLED_PATH**: Set path of the sampled rainfall data as SAMPLED_PATH
- **--pretrained-weights-path WEIGHTS_PATH** : Set path of pretrained weights as WEIGHTS_PATH

5.3 Evaluation (`evaluation.py`)

- **-h, --help**: Show help message
- **--batch-size**, **--data-path**, **--gpus**: See Section 5.1
- **--sampled-path**: See Section 5.2
- **--finetuned-weights-path WEIGHTS_PATH** : Set path of finetuned weights as WEIGHTS_PATH

5.4 Inference (`inference.py`)

- **-h, --help**: Show help message
- **latitude**: Latitude of target point (as a float value)
- **longitude**: Longitude of target point (as a float value)
- **time**: Timestamp of the last input radar image (Format: YYYYMMDDHH)
- **gap**: Gap between given images and target time (hour), the value should be 1, 2, 3, 4, 5, or 6
- **--data-path**, **--finetuned-weights-path**, **--gpus** : See Section 5.1 and 5.3

6 Examples

We prepared sample radar images and sample data for running the codes in `example_data` folder.

1. Pretraining example:
```
python pretrain.py --data-path ../example_data --gpus 0,1,2,3
```

2. Finetuning example:

```
python finetuning-ours.py --data-path ../example_data --sampled-path ../exam-
ple_data/sampled.pkl --pretrained-weights-path checkpoints/pretraining_10.pkt
--gpus 0,1,2,3
```

3. Evaluation example:

```
python evaluation.py --sampled-path ../example_data/sampled.pkl --data-path
../example_data --finetuned-weights-path example_checkpoints/finetuned.pkt --
gpus 0,1,2,3
```

4. Inference example:

```
python inference.py 35.0 126.0 2019060101 1 --data-path ../example_data --
finetuned-weights-path checkpoints/finetuned.pkt --gpus 0,1,2,3
```