Towards a Lossless Conversion for Spiking Neural Networks with Negative Spike Dynamics

This code can be used as the supplemental material for the paper: "Towards a Lossless Conversion for Spiking Neural Networks with Negative Spike Dynamics". (Submitted to *35th Conference on Neural Information Processing Systems (NeurIPS 2021)*, June, 2021) .

Citation:

To be completed.

Features:

- This supplemental material gives a reproduction function of ANN training, testing and converted SNN inference experiments in our paper. Besides, additional results for spiking MLP on FashionMNIST and VGG-Net (CNN 2) on CIFAR10 are provided.

File overview:

- `README.md` - this readme file.
- `MNIST` - the workspace folder for `LeNet` on MNIST.
- `FashionMNIST` - the workspace folder for `LeNet`/`MLP` on FashionMNIST.
- `CIFAR10` - the workspace folder for VGG-Net (`CNN 1` and `CNN 2`) on CIFAR10.

Requirements

Dependencies and Libraries:

- python 3.5 (https://www.python.org/ or https://www.anaconda.com/)
- tensorflow_gpu 1.2.1 (https://github.com/tensorflow)
- tensorlayer 1.8.5 (https://github.com/tensorlayer)
- CPU: Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz
- GPU: Tesla V100

Installation:

To install requirements,

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

Datasets:

- MNIST: dataset, preprocessing
- FashionMNIST: dataset, preprocessing

- CIFAR10: dataset, preprocessing

ANN Training

Before running:

- Please installing the required package Tensorflow and Tensorlayer (using our modified version)
- Please note your default dataset folder will be `workspace/data`, such as `supplemental material/CIFAR10/CNN_1/data`
- Select the index of GPU in the training scripts (0 by default)

Run the code:

for example (training, *k=0*, CNN1, CIFAR10):

```
$ cd CIFAR10/CNN_1
$ python Quant_CNN1_CIFAR10.py  --k 0 --resume False --learning_rate 0.001 --mode
'training'
```

ANN Inference

Run the code:

for example (inference, *k=0*, CNN1, CIFAR10):

```
$ python Quant_CNN1_CIFAR10.py --k 0 --resume True  --mode 'inference'
```

SNN inference

Run the code:

for example (inference, *k=0*, spiking CNN1, CIFAR10):

```
$ python Spiking_CNN1_CIFAR10.py --k 0
```

it will generate the corresponding log files including: `accuracy.txt`, `sop_num.txt`, `spike_collect.txt` and `spike_num.txt` in `./figs/k0/`.

Others

- We do not consider the synaptic operations in the input encoding layer and the spike output in the last classification layer (membrane potential accumulation) for both original ANN counterparts and converted SNNs.

- More instructions for running the code can be found in the respective workspace folder (`MNIST/README_MNIST.md`, `FashionMNIST/README_FashionMNIST.md`, `CIFAR10/README_CIAFR10.md`).

Results

Our proposed methods achieve the following performances on MNIST, FashionMNIST and CIFAR10 dataset:

MNIST:

Quantization Level	Network Size	Epochs	ANN	SNN	Time Steps
Full-precision	20C5-P2-50C5-P2-500	150	99.28%	N/A	N/A
k=1	20C5-P2-50C5-P2-500	150	99.32%	99.32%	13

FashionMNIST:

Quantization Level	Network Size	Epochs	ANN	SNN	Time Steps
Full-precision	400-400	150	89.83%	N/A	N/A
k=1	400-400	150	88.79%	88.79%	11
Full-precision	32C5-P2-64C5-P2-1024	100	90.01%	N/A	N/A
k=1	32C5-P2-64C5-P2-1024	100	89.99%	89.99%	17

CIFAR10:

Quantization Level	Network Size	Epochs	ANN	SNN	Time Steps
full-precision	96C3-256C3-P2-384C3-P2-384C3-256C3-P2-1024-1024	200	92.66%	N/A	N/A
k=1	96C3-256C3-P2-384C3-P2-384C3-256C3-P2-1024-1024	200	92.77%	92.77%	57
full-precision	128C3-256C3-P2-512C3-P2-1024C3-512C3-P2-1024-512	200	93.20%	N/A	N/A
k=1	128C3-256C3-P2-512C3-P2-1024C3-512C3-P2-1024-512	200	93.35%	93.35%	40

More question:

- There might be a little difference of results for multiple training repetitions, because of the randomization.
- Please feel free to reach out here or email: xxx@xxx, if you have any questions or difficulties. I'm happy to help guide you.