Instructions for running MNIST experiments

File overview:

- `README_MNIST.md` - this readme file for MNIST.
- `spiking_ulils.py` - the functions of spiking convolution and linear.
- `figs` - visualization folder for SNN performance.
  - `accuracy_speed.py` - the accuracy versus speed script for `spiking LeNet` with different quantization precisions on MNIST
  - `sops.py` - the computing operations script for `spiking LeNet` with different quantization precisions on MNIST
  - `sparsity.py` - the spike sparsity script for `spiking LeNet` with different quantization precisions on MNIST
- `LeNet` - LeNet for MNIST.
  - `tensorlayer` - our provided tensorlayer package.
  - `Quant_LeNet_MNIST.py` - the training script for `LeNet` with optional quantization precision *k* on MNIST
  - `Spiking_LeNet_MNIST.py` - the evaluation script for `spiking LeNet` with optional quantization precision *k* on MNIST
  - `FP32_LeNet_MNIST.py` - the training script for `LeNet` with `full precision (float32)` on MNIST

ANN Training

Before running:

- Please note your default dataset folder will be `./data`

Run the code:

for example (training, *k=0*, LeNet, MNIST):

```
$ python Quant_LeNet_MNIST.py --k 0 --resume False --mode 'training'
```

finally, it will generate the corresponding model files including: `checkpoint`, `model_MNIST_advanced.ckpt.data-00000-of-00001`, `model_MNIST_advanced.ckpt.index`, `model_MNIST_advanced.ckpt.meta` and `model_MNIST.npz`.

ANN Inference

Run the code:

for example (inference, *k=0*, LeNet, MNIST):

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

Then, it will print the corresponding ANN test accuracy.

SNN inference

Run the code:

for example (inference, *k=0*, spiking LeNet, MNIST):

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

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

Visualization

Accuracy versus speed:

```
$ cd figs
$ python accuracy_speed.py
```

Firing sparsity:

```
$ python sparsity.py
```

Computing operations:

```
$ python sops.py
```

Results

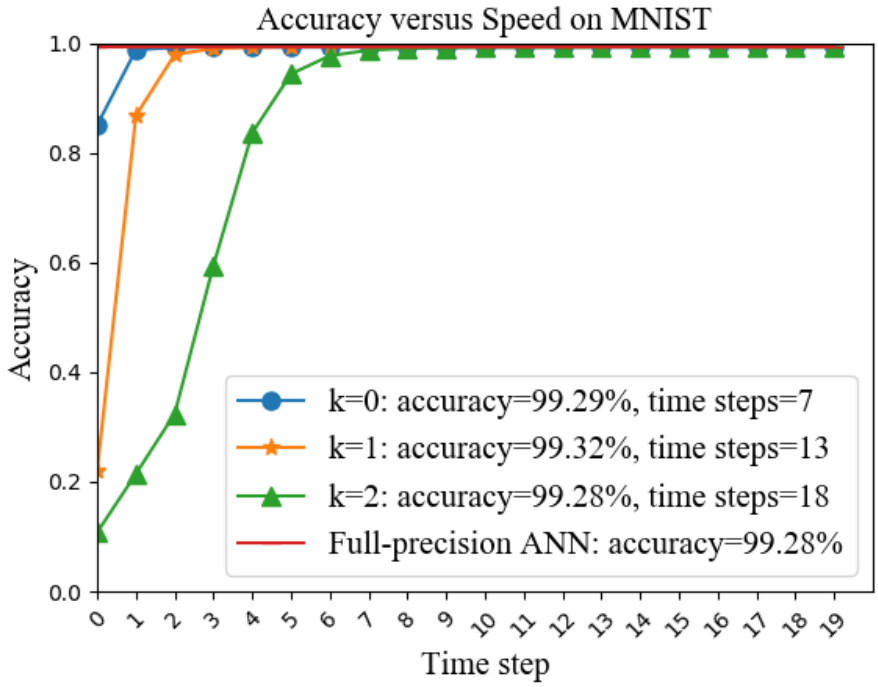
Our proposed method achieves the following performance on :

MNIST:

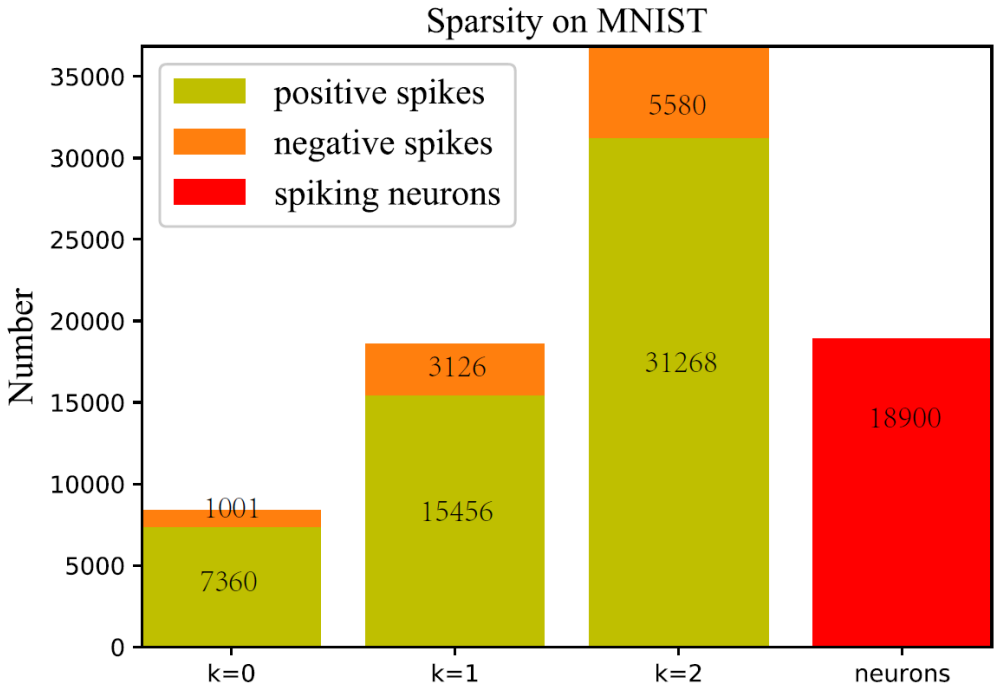
Quantization Level	Network Size	Epochs	ANN	SNN	Time Steps
Full-precision	20C5-P2-50C5-P2-500	150	99.28%	N/A	N/A
k=1	20C5-P2-50C5-P2-500	150	99.32%	99.32%	13

Accuracy versus speed:

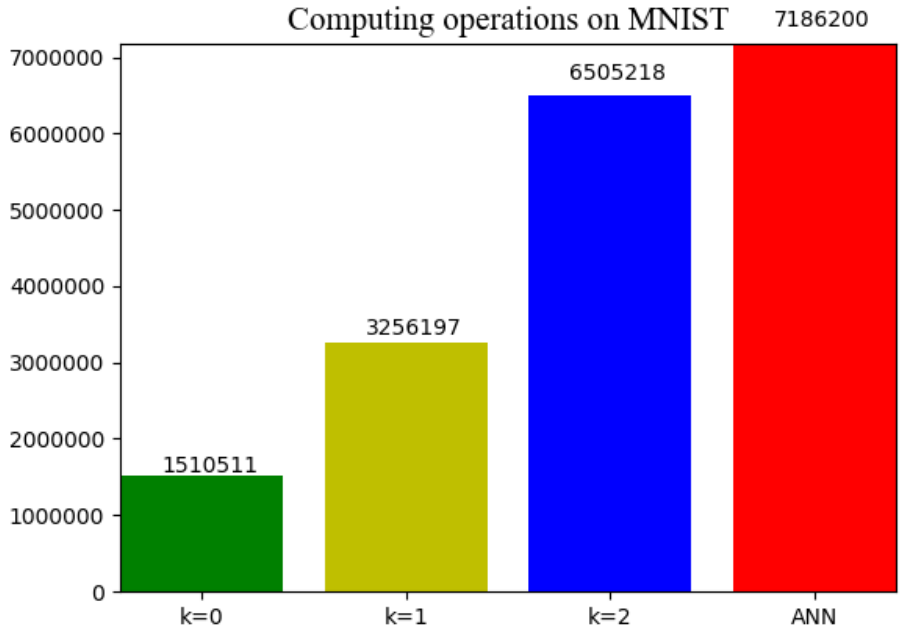
LeNet: 20C5-P2-50C5-P2-500



Firing sparsity:



Computing operations:



Notes

- We do not consider the synaptic operations in the input encoding layer and the spike outputs in the last classification layer (membrane potential accumulation instead) for both original ANN counterparts and converted SNNs.
- We also provide some scripts for visualization in ./figs, please move `SNN_accuracy.txt`, `sop_num.txt`, `spike_collect.txt` and `spike_num.txt` to this folder and directly run the scripts.

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.