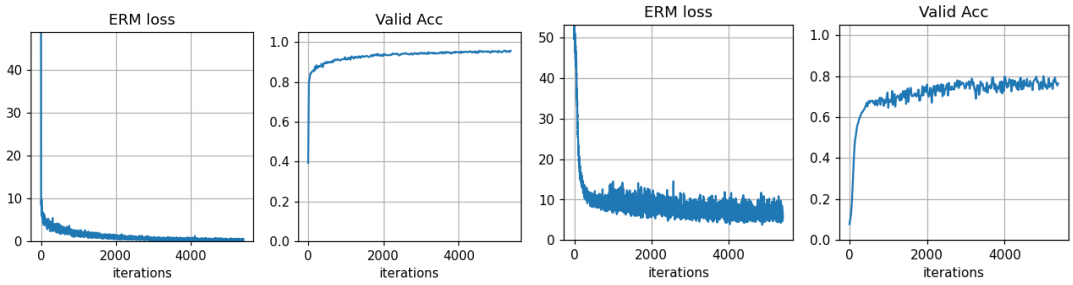
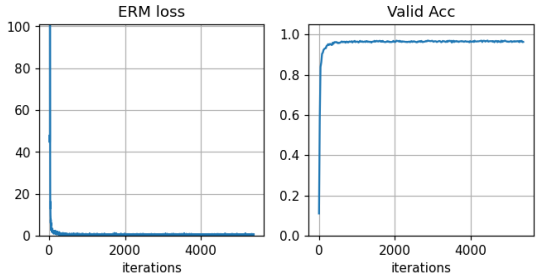
Something about softweight：

BCNN on MNist dataset (~5 million parameters)

Lr = 0.1, With/without softweight:

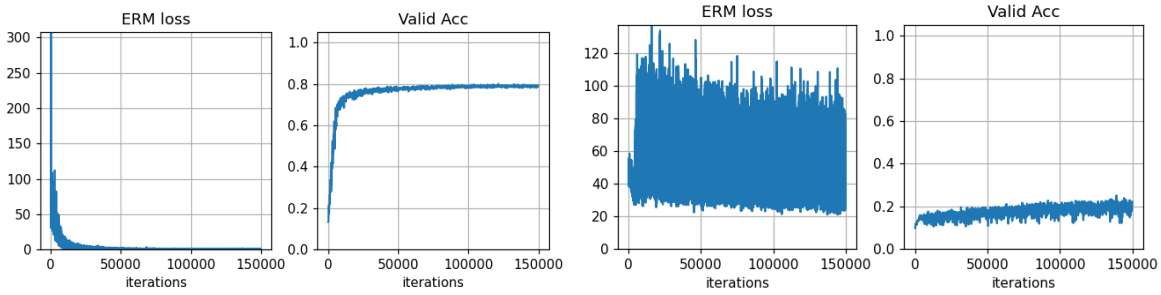


Lr = 0.1, without softweight, using the approximation of the softweight. (5 bits storage):

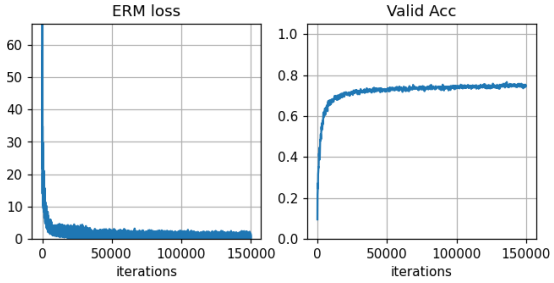


BVGG16 on Cifar-10 dataset (~12 million parameters)

Lr = 5e-3, Adam optimizer With/without softweight:

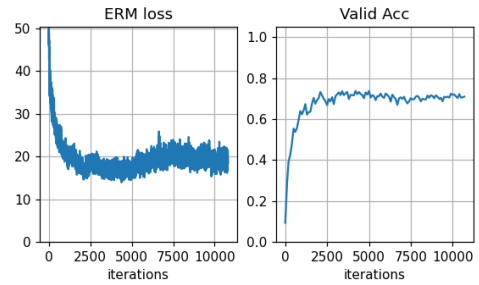
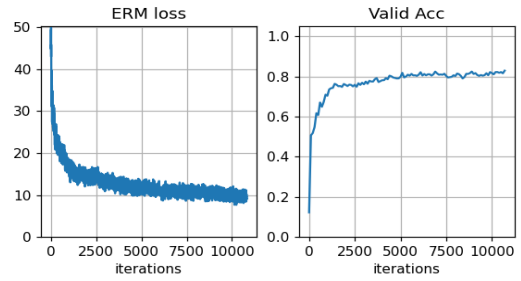
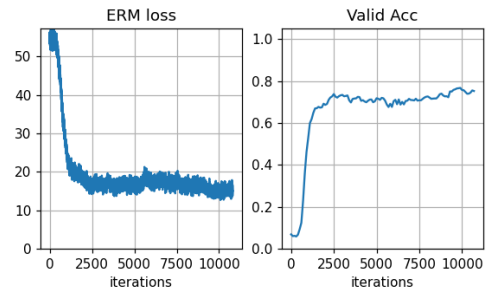
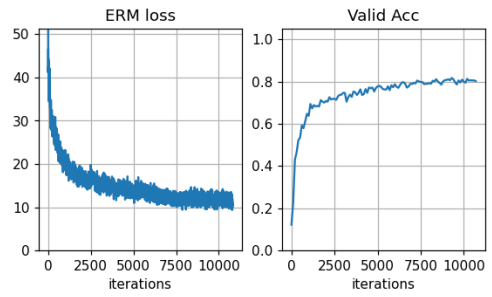


Lr = 1e-2, SGD optimizer, using the approximation of the softweight. (from 5 bits to 8 bits):



Comparison of different strategies to make forward-mode cheaper :

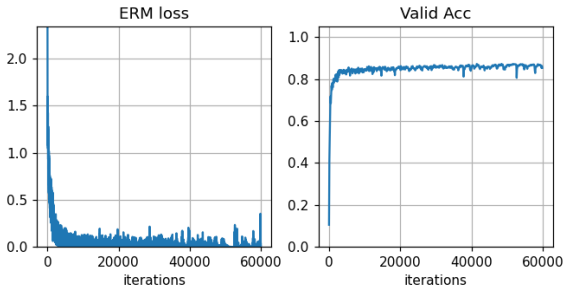
BCNN on MNIST with 10 different directions:

	Normal directional derivative; lr = 1e-6	Binarize directional derivative; lr = 1e-2
With SW		
No SW (5 bits storage)		

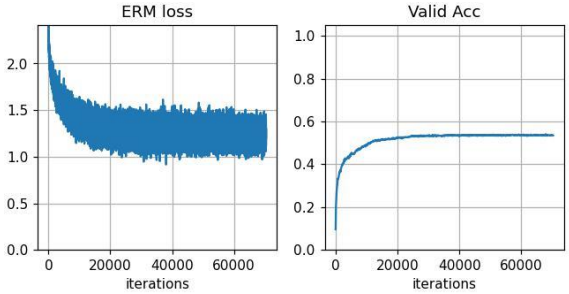
Training VGG16 (Not binary version) on Cifar-10:

Using backpropagation method:

Lr = 1e-3:



Use naïve forward mode method with lr = 1e-4, num_dir = 20:



Orthogonal trick:

Assumption:

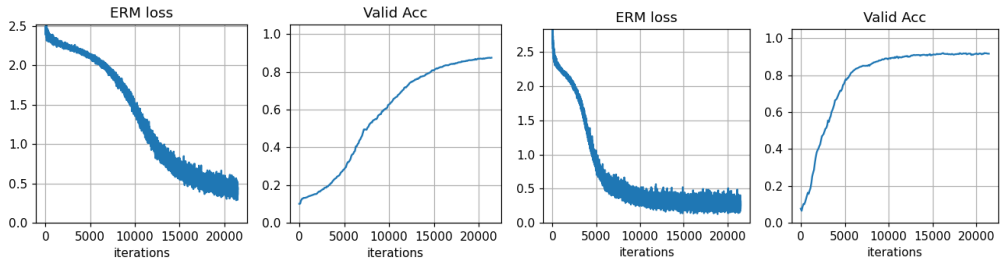
1. the directions of gradients between 2 adjacent iterations remain similar.
2. The distribution of gradients is zero-centered.

Then we make selection of the random directions of the t+1 iteration orthogonal to the t iteration ones.

This will ensure the directions choose at the t+1 iteration is no worse than the directions at the t iteration:

With simple MLP on MNist dataset:

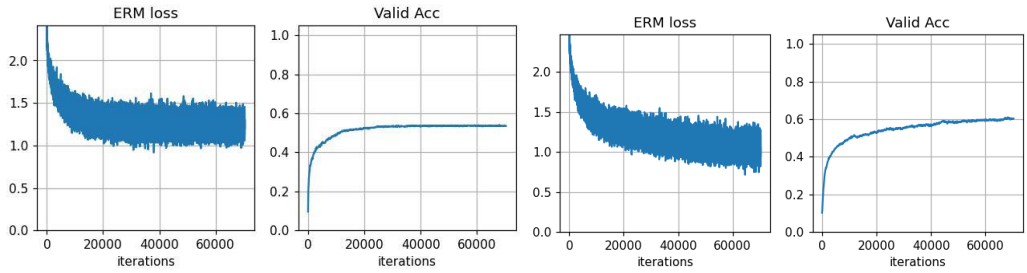
lr = 1e-4, num_dir = 1:



Very similar to when using the memory augmented optimizer!

With VGG16 on Cifar-10:

Lr = 1e-4, num_dir = 50/25:



Small improvement but still not enough.

Problem:

1. The forward-mode method is not doing good when the scale of model get larger.

2. The softweight approximation trick can reduce memory consumption but is we use optimizers like Adam then the memory cost goes up again.

3. The tradeoff between self-implemented forward-mode autodiff and Beta-version pytorch's API.

4. The gap of performance between the normal VGG and the Binary VGG on Cifar-10

5. Warning: Deallocating Tensor that still has live PyObject references. This probably happened because you took out a weak reference to Tensor and didn't call _fix_weakref() after dereferencing it. Subsequent accesses to this tensor via the PyObject will now fail.

Next step:

1. Try to make the forward mode method work better on large scale model. More investigation on how to utilize the historical information to get a better directional directive. (Most of them wouldn't allow the use of binary forward propagation).