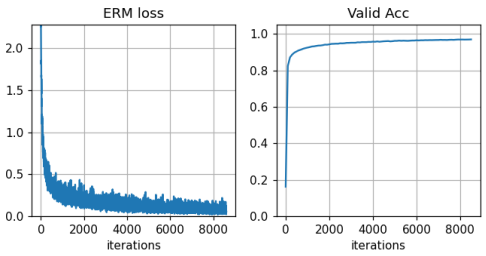
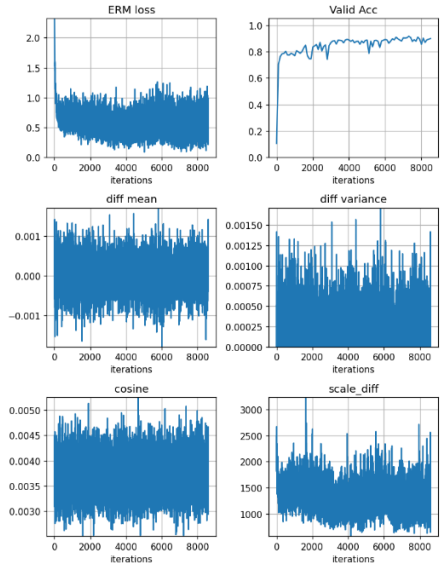
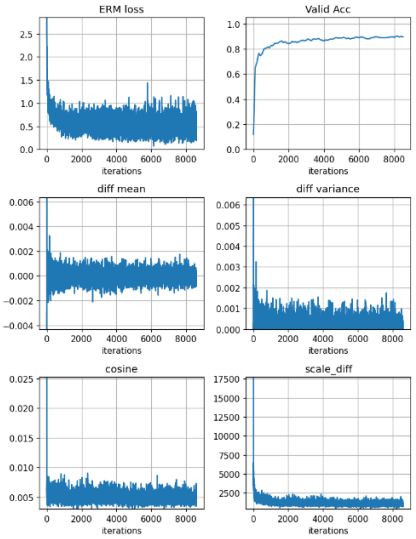
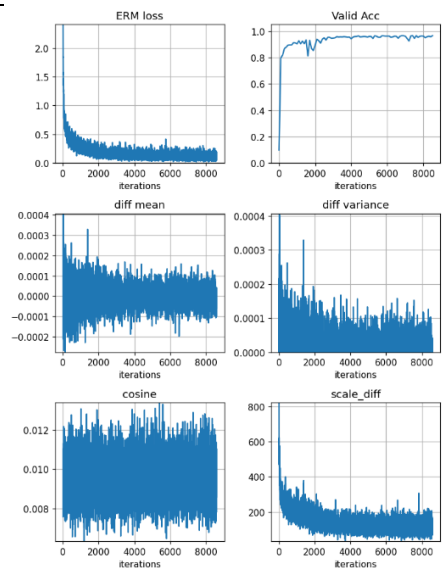
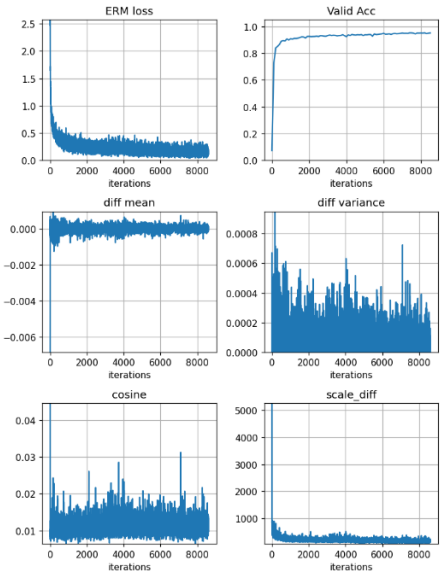
Projection method on Conv layer and FC Layer:

On the original CNN (6 milli)which FC layer is at dominant place Lr = 1e-3, num_dir = 20:

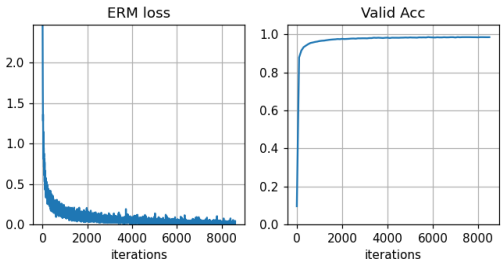
Backprop baseline:

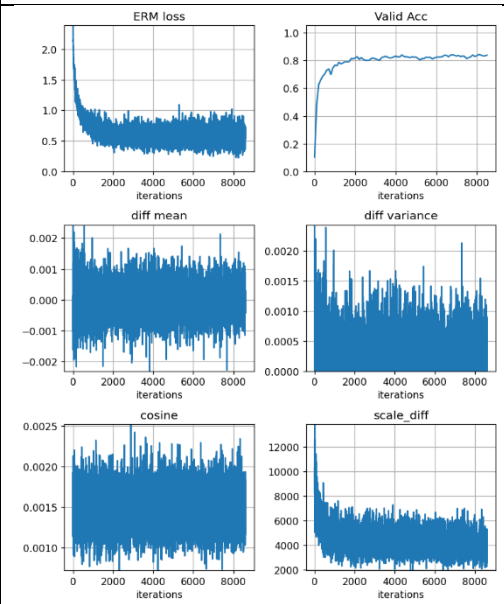
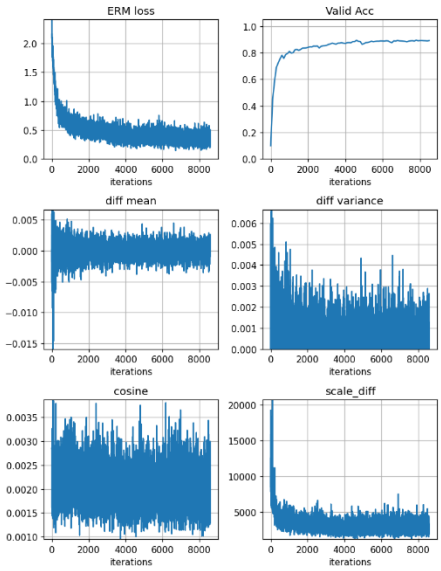
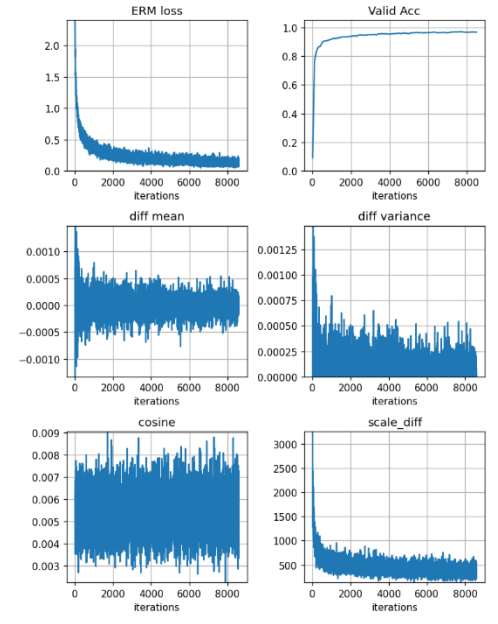
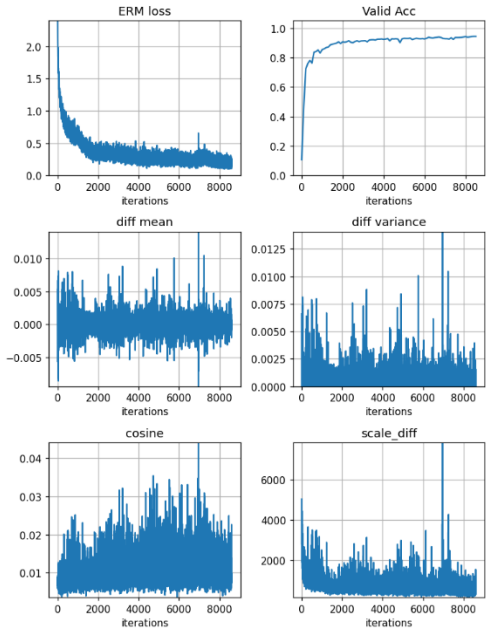


	normal conv	Projected conv
Normal fc		
Projected fc		

Modified version of CNN (9 milli), more portion for conv layer. Lr=1e-3, num_dir = 20

Backprop baseline:



	normal conv	Projected conv
Normal fc		
Projected fc		

Conclusion:

1. They both work.

2. The improvement for linear layer is very good, the improvement for conv layer is also observable, and it gets larger if the portion of conv layer's parameter amount increases.

3. This can increase the memory consumption because we need to store the input (or the projection matrix) to reconstruct the random vectors instead of only storing a random seed. But still an improvement regarding batch size.

4. Although both methods improve the performance when implemented separately, and the cosine similarity gets better and reaches the best when both are implemented at the same time. The performance is not the best when both methods are implemented at the same time.

Next step:

1. The proof for unbiased estimation of linear layer is done, so what's left is:
   (1) Unbiased estimation for conv layer
   (2) Variance analysis for both.

   These mathematic analysis will definitely help us to overcome the fourth point in the "Conclusion" section.

2. Implement this method on Binary Neural Network.

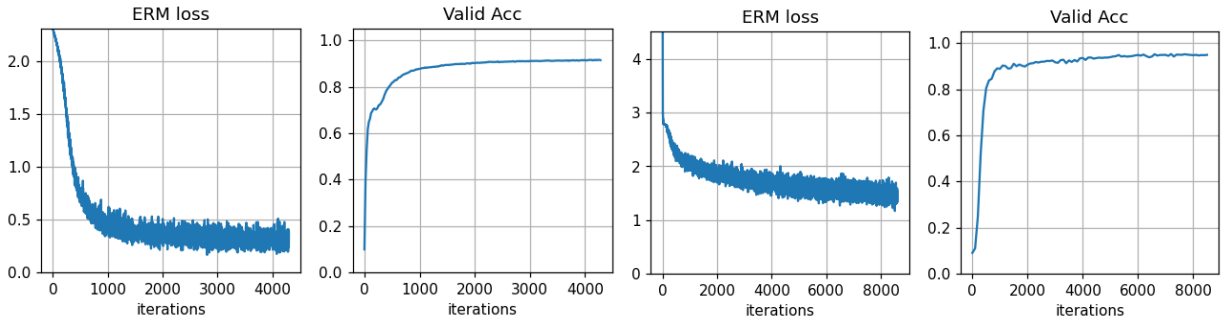
Forward-forward Algorithm:

Motivation:

1. Quantizer gap.
2. Binarization act like the normalization step.
3. Cheap for on-device training
4. Lower the expense for FFA inference

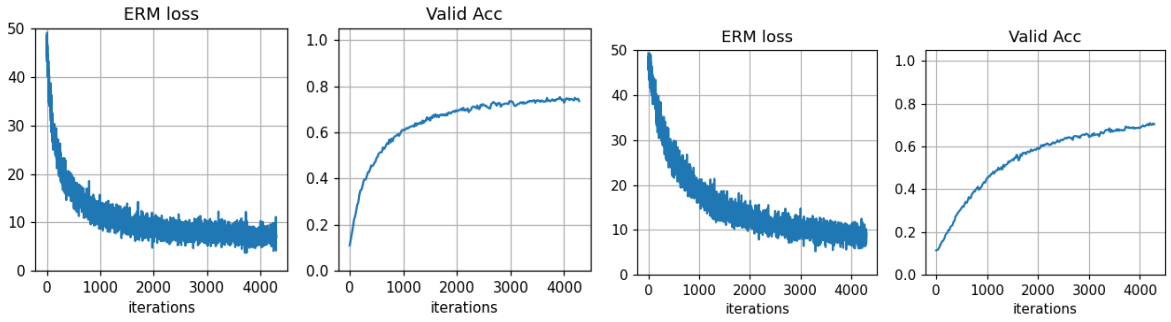
MLP on MNist: (All are ran for 20 epochs, the difference of x-axis is from batchsize)

Backprop/FFA:

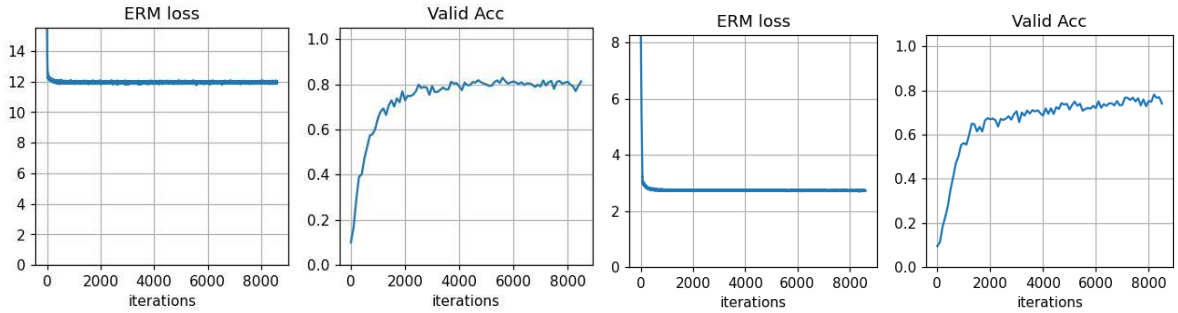


BMLP on MNist:

Backprop baseline (without bn layers) SGD/Adam



FFA SGD/Adam: (Tuned separately, slightly different loss function)



Next step:

1. explore if there are similar approaches for other architecture
2. design the loss function to reach better performance.
3. Try to develop a symmetric training method to utilize the information of the entire neural network.