Team MWY report

Problem statement.

In this project we use Python 3.9 and Pytorch. We train our model locally on a RTX 3090. Code is tested under Ubuntu system.
First, we implemented a ResNet18 model on Cifar10 Dataset. By tuning the hyper-parameters (learning rate and epoch), we reach 93.21% test accuracy. Then we implement one-shot pruning and iterative pruning and compare the test accuracy and test accuracy drop. In this problem we do not consider the speed up(bonus points) part.

Implementation.

1. Overall Program design

In this project, we use Pytorch to implement two different CNN pruning algorithms, one shot pruning and iterative pruning. We build the ResNet 18 on our own and then we train ResNet18 on the given dataset Cifar 10. After about 40 epochs of training, the learning rate decaying from 0.1 to 0.001, we get a well-trained model, with about 93% test accuracy. Our second step is to prune the well-trained model into a predefined set of sparsity ratios by using some API in Pytorch. In this part, we remove some unimportant connections which convert our original connected network to a sparse network. Then we retrain the network to finetune the weights of the remaining connections. Up to this step, the one-shot pruning is done. Finally, we use one shot pruning progress and finetune repeatedly till the eventual sparsity ratio requirement is met to implement the iterative pruning.

2. Pruning algorithms

2.1 One shot pruning

We first iterate get each module in the model and find the convolution layer and fully connect layer in the module. Then, we get the weight and append it to a list. At last, we use the global_unstructured API in Torch, pruning method of which is L1Unstructured pruning. At last, we update the parameters which is masked by pruning method.

2.2 Iterative pruning

After finishing one shot pruning, iterative pruning is relatively easy. We use a loop to judge whether current sparsity satisfied our target. After each pruning, we finetune the model for 5 epochs to and record 2 data: the test accuracy after pruning and the accuracy drop after pruning (accuracy before pruning minus accuracy just after pruning).

Experimental results.

1. Evaluation mythology
   A) Sparsity
   We first compute # of zero parameters in each module and the total number of all parameters in each module. Then, we iterate all the convolution layer and Linear layer of the model to summarize the # zero parameters and # of total parameters of all module and compute the global sparsity, which equal the # of zero parameters in all the convolution layer and Linear layer divide by the total number of all parameters in all the convolution layer and Linear layer. The formula shows as:

$$sparsity = \frac{\#zero\ parameters}{\#parameter}$$

   B) Test accuracy and accuracy drop
   The method we compute test accuracy and accuracy drop as the same as the instruction in the project description.

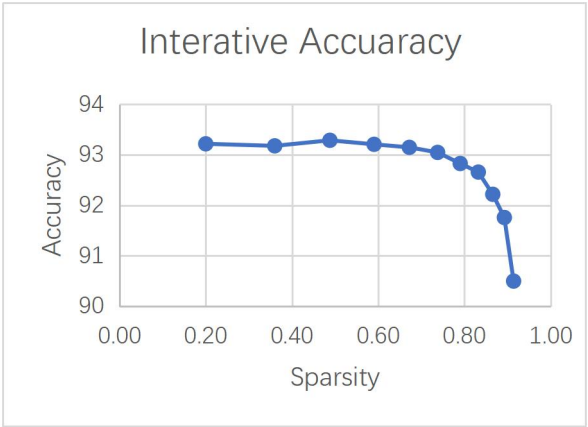
$$\text{Test Accuracy} = \frac{Correctly\ classified\ data\ points\ from\ test\ data}{total\ amount\ of\ data\ points\ from\ test\ data}$$

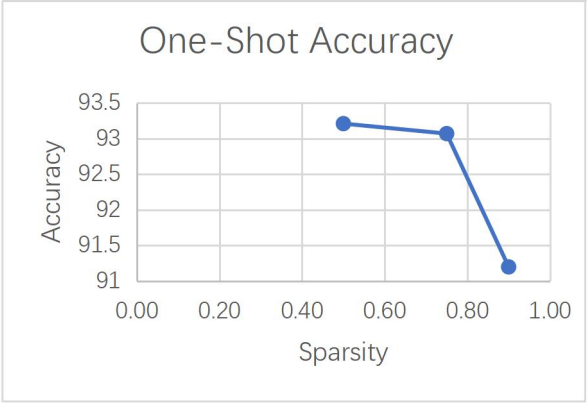
Accuracy drop = test accuracy without pruning - test accuracy after pruning

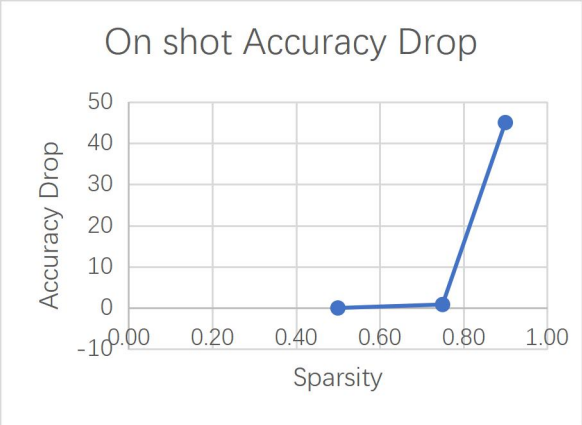
2. The results and explanation

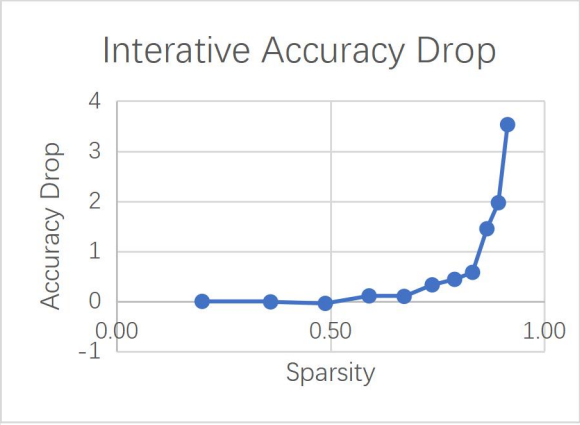
Here is our test result:

type	sparsity	accuracy	drop
iterative	0.20	93.22	0
iterative	0.36	93.18	-0.01
iterative	0.49	93.29	-0.04
iterative	0.59	93.21	0.11
iterative	0.67	93.15	0.1
iterative	0.74	93.05	0.33
iterative	0.79	92.83	0.44
iterative	0.83	92.66	0.58
iterative	0.87	92.22	1.45
iterative	0.89	91.76	1.97
iterative	0.91	90.5	3.53
One-shot	0.50	93.21	-0.03
One-shot	0.75	93.07	0.81
One-shot	0.90	91.2	45.01









The chart and table vividly show that, the larger the sparsity, the greater the impact on accuracy. In one-shot pruning, the accuracy drops when sparsity = 0.5 and 0.75 almost equals to 0, but when the sparsity =0. 90, accuracy drop quickly rises to 45%, which is relative a huge number. The same case shows in the iterative pruning. When sparsity falls in the range of 80 to 100, the accuracy drop rises sharply. Overall, after finetuning, the accuracy is still acceptable. Although the sparsity is greater than 90%, the test accuracy after finetuning is still greater than 90%.

Summary and Takeaway.

In this project, we build the ResNet 18 on our own and train it to get the original model. Then, one shot global pruning and iterative pruning are implemented by Pytorch. We prune the model by the two methods, finetune them and get the final model after finetuning.

The result suggests that in most of the neural network models, we might save a lot of space by pruning most of the parameters, but we can achieve similar performance compared with the model before pruning. The difference of the accuracy between the original model and pruned model is much smaller than we expected. However, the most important step is that we should finetune the model after pruning, especially for iterative pruning we should finetune after each one-shot pruning. Otherwise, the accuracy will be affected a lot by pruning.

Team Contribution.

Weigeng Li: Implement ResNet18 model and build tables and charts of result.
Haoyang Ma: Design and implement one-shot pruning. Design the training progress.
Sijing Yu: Design and implement iterative pruning. Design the training progress.

We write the report together.

Artifact evaluation.

Just run the prune.py and follow the instruction in the console. The output of 3 dictionary will just show in the console. The plots are draw all by Excel. We input s=0.2, which is the sparsity per iteration, to get all our results.