CSC 591
REAL TIME AND HIGH-PERFORMANCE ML
UNIT PROJECT 1
sthakur5
xhui

(1) What pruning methods you have applied to which DNN model?

Model Used	VAE	
Pruning Method 1	L1 Norm Pruner	L1 norm pruner computes the l1 norm of the layer weight on the first dimension, then prunes the weight blocks on this dimension with smaller l1 norm values. i.e., compute the l1 norm of the filters in the convolution layer as metric values, and compute the l1 norm of the weight by rows in the linear layer as metric values.
Pruning Method 2	FPGM Pruner	FPGM pruner prunes the blocks of the weight on the first dimension with the smallest geometric median. FPGM chooses the weight blocks with the most replaceable contribution.
Pruning Method 3	L2 Norm Pruner	L2 norm pruner is a variant of L1 norm pruner. The only different between L2 norm pruner and L1 norm pruner is L2 norm pruner prunes the weight with the smallest L2 norm of the weights.L2

		norm pruner also supports dependency-aware mode.

(2) What configurations did you try on each pruning method, and what machine(s) you have used, including the CPU and GPU models of the machine(s).
(2) Configuration used for each pruner:

Sparsity	Layer Pruned	Excluded Layer
.2	Linear	Last layer is not pruned since we want the output dimension to be same as original.
.5		
.8		

System Configuration:

Cluster	Node	CPU Detail	GPU Detail
NCSU ARC Cluster	c25	**Architecture**: x86_64 **Thread(s) per core:** 2 **Core(s):** 16 Model name: **AMD EPYC 7302P** 16-Core Processor **CPU MHz:** 1496.343 **CPU max MHz:** 3000.0000 **CPU min MHz:** 1500.0000	**GPU make**:NVIDIA GeForce RTX 2060 SUPER, 90.06.45.00.01 **Memory**: 8GB

(3) What results you have obtained, including the output of "print(model)" of your original model and pruned model, their running speeds and accuracies;

(3.) <u>Model size:</u>

Pruner	Sparsity	"print(model)" output
None	N/A	VAE((fc1): Linear(in_features=784, out_features=400, bias=True) (fc21): Linear(in_features=400, out_features=20, bias=True) (fc22): Linear(in_features=400, out_features=20, bias=True) (fc3): Linear(in_features=20, out_features=400, bias=True) (fc4): Linear(in_features=400, out_features=784, bias=True))
L1 Norm	.8	VAE((fc1): Linear(in_features=784, out_features=80, bias=True) (fc21): Linear(in_features=80, out_features=8, bias=True) (fc22): Linear(in_features=80, out_features=8, bias=True) (fc3): Linear(in_features=8, out_features=80, bias=True) (fc4): Linear(in_features=80, out_features=784, bias=True)) is model
L1 Norm	.5	VAE((fc1): Linear(in_features=784, out_features=200, bias=True) (fc21): Linear(in_features=200, out_features=18, bias=True) (fc22): Linear(in_features=200, out_features=18, bias=True) (fc3): Linear(in_features=18, out_features=200, bias=True) (fc4): Linear(in_features=200, out_features=784, bias=True)) is model
L1 Norm	.2	VAE((fc1): Linear(in_features=784, out_features=320, bias=True) (fc21): Linear(in_features=320, out_features=20, bias=True) (fc22): Linear(in_features=320, out_features=20, bias=True) (fc3): Linear(in_features=20, out_features=320, bias=True) (fc4): Linear(in_features=320, out_features=784, bias=True)) is model
FPGM	.8	VAE((fc1): Linear(in_features=784, out_features=80, bias=True) (fc21): Linear(in_features=80, out_features=8, bias=True) (fc22): Linear(in_features=80, out_features=8, bias=True) (fc3): Linear(in_features=8, out_features=80, bias=True) (fc4): Linear(in_features=80, out_features=784, bias=True)) is model

FPGM	.5	VAE((fc1): Linear(in_features=784, out_features=200, bias=True) (fc21): Linear(in_features=200, out_features=17, bias=True) (fc22): Linear(in_features=200, out_features=17, bias=True) (fc3): Linear(in_features=17, out_features=200, bias=True) (fc4): Linear(in_features=200, out_features=784, bias=True)) is model
FPGM	.2	VAE((fc1): Linear(in_features=784, out_features=320, bias=True) (fc21): Linear(in_features=320, out_features=20, bias=True) (fc22): Linear(in_features=320, out_features=20, bias=True) (fc3): Linear(in_features=20, out_features=320, bias=True) (fc4): Linear(in_features=320, out_features=784, bias=True)) is model

Running Speed:

Since VAE is a generative model we calculate the running speed by simply measuring the training and evaluation times.

Pruner	Sparsity	Training Time(seconds)	Evaluation Time(seconds)	Pruned Model Training Time(seconds)	Pruned Model Evaluation(seconds)
L1	0.2	5.1782	0.892	5.2279	0.8798
L1	0.5	5.078	0.8802	5.082	0.885
L1	0.8	5.1023	0.8825	4.981	0.8859
FPGM	0.2	5.1311	0.8878	5.089	0.873
FPGM	0.5	5.0986	0.8986	5.133	0.8923
FPGM	0.8	5.0616	0.8701	4.963	0.883

Observations:

There is no improvement in training time and evaluation time before and after pruning the model.

Reason:

According to papers [3],[4], L1 norm pruner and FPGM pruner are designed for Convolutional Neural Networks. They can get better performance for the data-dependency mode “Conv2d”. However, the VAE model is an auto-generative model and is composed of an encoder and a decoder. In our experiment, we just use the “linear” configuration, thus, we cannot get a good speed up from these two pruners.

Accuracy comparison:

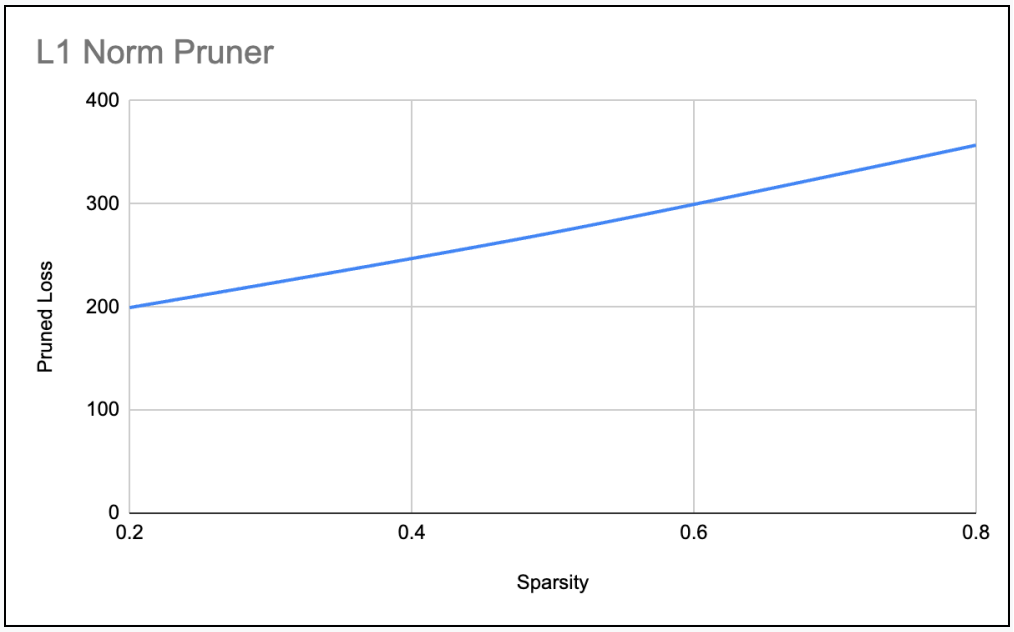
Since VAE is an auto-generative model and we are not predicting anything we need to figure out the metrics necessary to compare the performance of the pruned vs unpruned model. These are:

1. Loss
2. KLD Loss - Kullback-Leibler divergence. The purpose of the KL divergence term in the loss function is to make the distribution of the encoder output as close as possible to a standard multivariate normal distribution
3. Reconstruction loss - measures the quality of the autoencoding, i.e. the error between the original sample and its reconstruction.

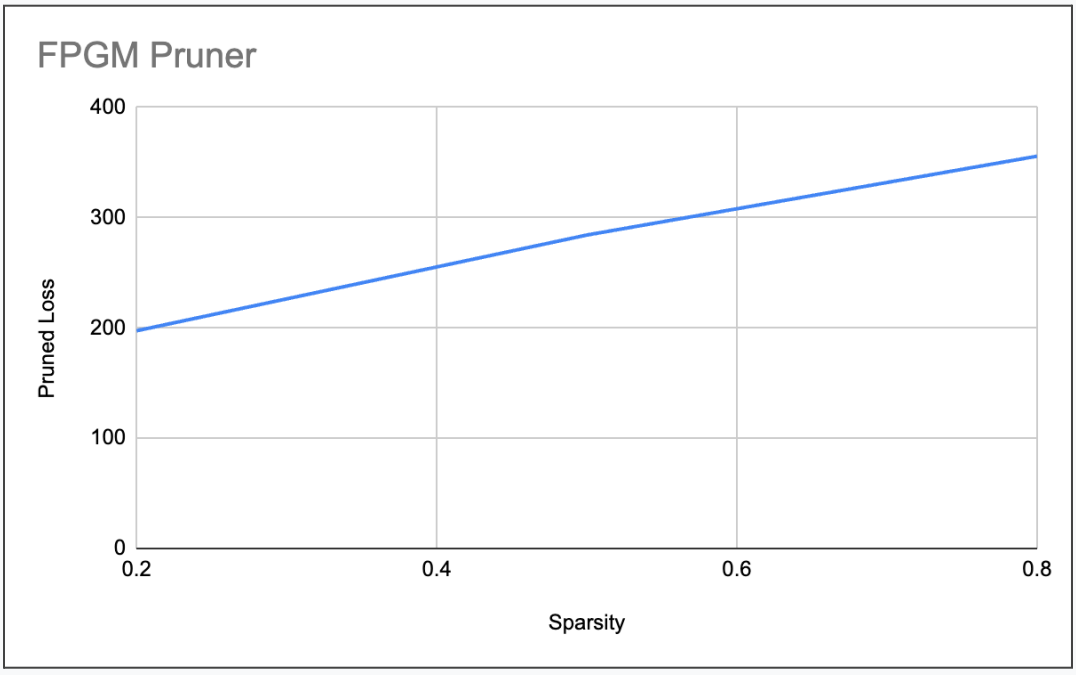
Note: Loss= KLD loss + Reconstruction loss

Total Loss for all configurations:

Pruner	Sparsity	Unpruned Loss	Pruned Loss	Unpruned KLD Loss	Pruned KLD Loss	Reconstruction Loss (Unpruned)	Reconstruction Loss (pruned)
L1	0.2	105.4703	199.3805	79.8444	177.5212	25.6259	21.8593
L1	0.5	105.6899	272.1167	79.5617	263.8532	26.1282	8.2635
L1	0.8	105.6894	356.9524	80.3431	356.9524	25.3462	1.0618



Pruner	Sparsity	Unpruned Loss	Pruned Loss	Unpruned KLD Loss	Pruned KLD Loss	Reconstruction Loss (Unpruned)	Reconstruction Loss (pruned)
FPGM	0.2	105.9588	197.3874	80.7086	176.5501	25.2502	20.8373
FPGM	0.5	105.7299	284.2211	80.5747	278.6445	25.1551	5.5766
FPGM	0.8	106.0563	355.7219	80.2888	355.0523	25.7675	0.6696



Observations:
We can see L1NormPruner and FPGM Pruner are both performing similarly for our use case. As stated here VAE suffers from over-pruning in the original model itself. Hence, a number of stochastic factors fail to learn anything because of inactivity. Therefore, we can say that pruning it further when it has an issue with over pruning in its original form is expected to give a higher loss and worse results.

Theoretical Inference Time Improvement:

How is FLOPS calculated?

Our VAE model consists of only linear layers. This simplifies the calculation as for a layer with m inputs to n outputs number of FLOPS = m*n
Hence, we can get the number of FLOPS by summing up the layerwise input units*output units.

Sparsity	**FLOPS**	**Improvement**
0	651200	1
.2	520960	1.25
.5	329152	1.978
.8	127360	5.11

Model Size:
Unpruned model size: 2614151 bytes

Sparsity	Pruned Model Size(bytes)
0.2	2092551
0.5	1290887
0.8	513223

Observation: Model size is changing appropriately in proportion to the sparsity.

(4) What lessons you have learned through the project;

1. Pruning can actually prune the model according to its sparsity. The sparse model can get significant shrink via pruning while the dense model gets trivial shrink.
2. Not every pruner can perform well for every model. Some pruners can only perform well for some specific models. We should get the model characteristics and find the appropriate pruner.
3. Over-pruning can produce bad results and high losses. Even though pruning can make the model small, we still need to be careful about the accuracy and loss when pruning.

(5) A link to your github repository that contains all the scripts used in this project and a README describing the structure of the repository and how the folders in the repository correspond to the items in the report. The report must be in PDF, with the five required parts clearly marked

(5) https://github.com/Sakshamrzt/CSC-591-P1

References:

1.https://www.vincent-lunot.com/post/on-the-use-of-the-kullback-leibler-divergence-in-variational-autoencoders/

2.https://olegpolivin.medium.com/experiments-in-neural-network-pruning-in-pytorch-c18d5b771d6d

3. Li, Hao, et al. "Pruning filters for efficient convnets." arXiv preprint arXiv:1608.08710 (2016).

4. He, Yang, et al. "Filter pruning via geometric median for deep convolutional neural networks acceleration." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019.