Prashan sengo (psengo)
CSC791

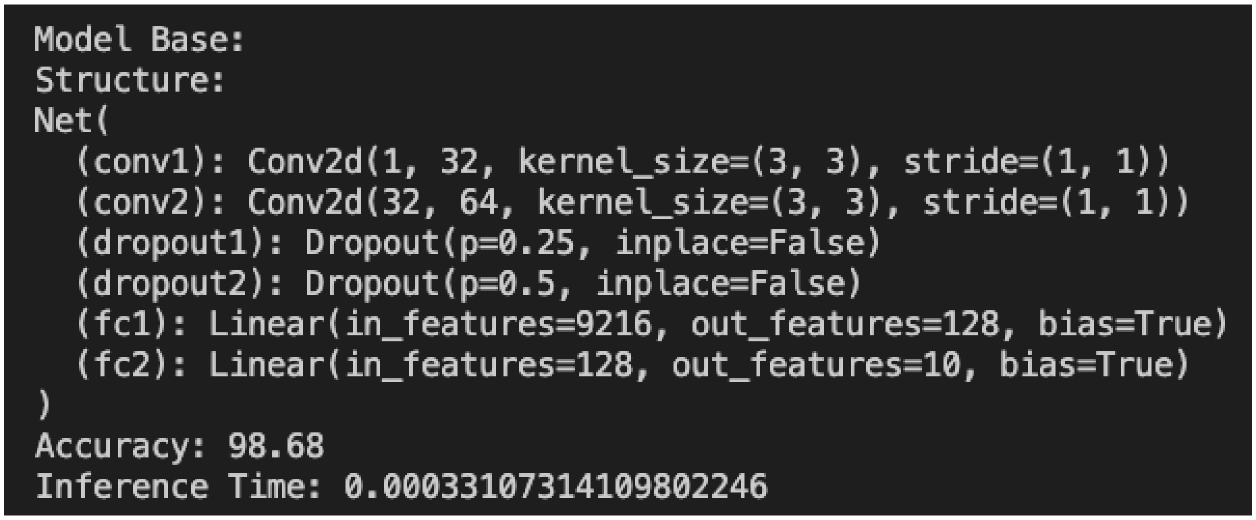
Report 1

Link to github repo: https://github.com/psengo7/csc791_P1

Pruning Methods/Results:

- Four models were created off of pruning a pre-trained MNIST model (Link for model here) using either L1NormPruner or FPGPruner with configurations of sparsity .2 or .7 on the convolution layers.
- The model structure, pruner, configurations, accuracy, and inference time(seconds) are outputted for each of the models below.
- The computing specs used for all models is the following::
    - In arc cluster using node C9
    - **CPU used:** Intel(R) Xeon(R) Silver 4110 CPU @ 2.10GHz
    - **GPU used:** NVIDIA Corporation GP104GL [Quadro P4000], ASPEED Technology, Inc. ASPEED Graphics Family

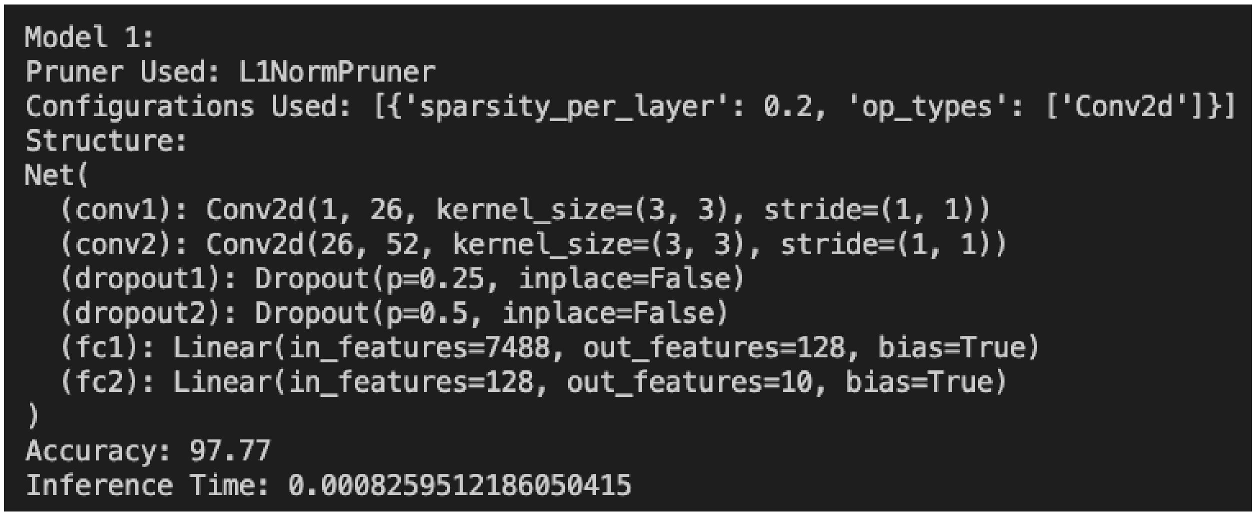
Pre-trained MNIST model with no pruning:



```
Model Base:
Structure:
Net(
  (conv1): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1))
  (conv2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1))
  (dropout1): Dropout(p=0.25, inplace=False)
  (dropout2): Dropout(p=0.5, inplace=False)
  (fc1): Linear(in_features=9216, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=10, bias=True)
)
Accuracy: 98.68
Inference Time: 0.00033107314109802246
```

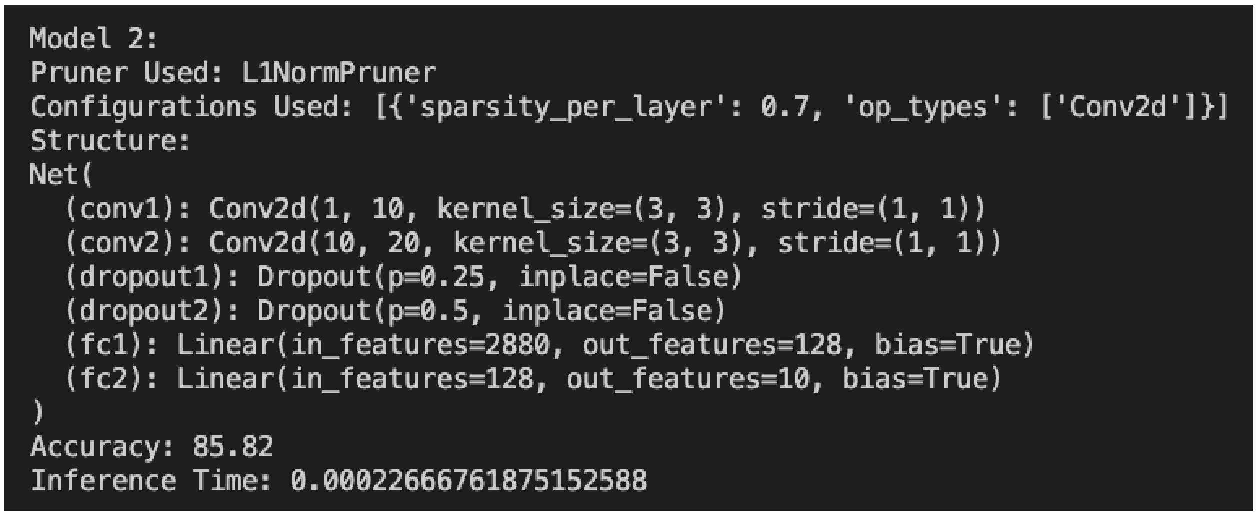
Prashan sengo (psengo)
CSC791

Model 1:



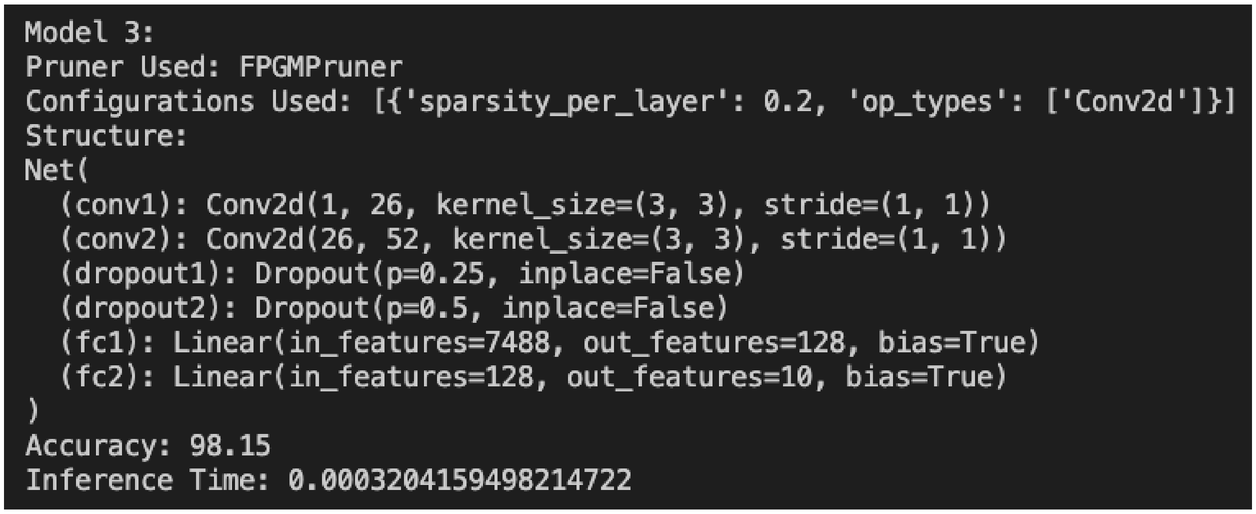
```
Model 1:
Pruner Used: L1NormPruner
Configurations Used: [{'sparsity_per_layer': 0.2, 'op_types': ['Conv2d']}]
Structure:
Net(
  (conv1): Conv2d(1, 26, kernel_size=(3, 3), stride=(1, 1))
  (conv2): Conv2d(26, 52, kernel_size=(3, 3), stride=(1, 1))
  (dropout1): Dropout(p=0.25, inplace=False)
  (dropout2): Dropout(p=0.5, inplace=False)
  (fc1): Linear(in_features=7488, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=10, bias=True)
)
Accuracy: 97.77
Inference Time: 0.0008259512186050415
```

Model 2:



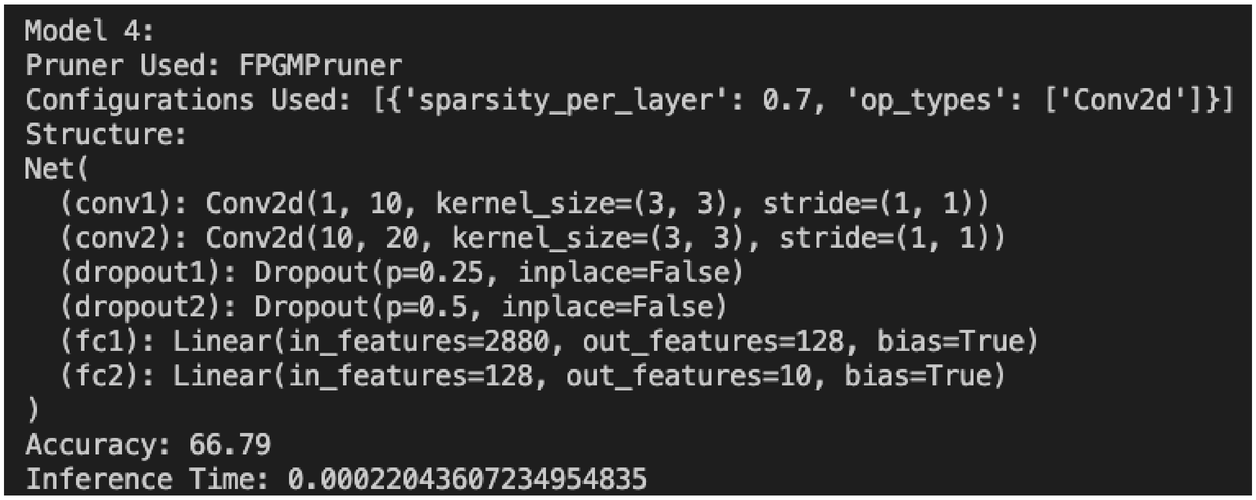
```
Model 2:
Pruner Used: L1NormPruner
Configurations Used: [{'sparsity_per_layer': 0.7, 'op_types': ['Conv2d']}]
Structure:
Net(
  (conv1): Conv2d(1, 10, kernel_size=(3, 3), stride=(1, 1))
  (conv2): Conv2d(10, 20, kernel_size=(3, 3), stride=(1, 1))
  (dropout1): Dropout(p=0.25, inplace=False)
  (dropout2): Dropout(p=0.5, inplace=False)
  (fc1): Linear(in_features=2880, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=10, bias=True)
)
Accuracy: 85.82
Inference Time: 0.00022666761875152588
```

Model 3:



```
Model 3:
Pruner Used: FPGMPruner
Configurations Used: [{'sparsity_per_layer': 0.2, 'op_types': ['Conv2d']}]
Structure:
Net(
  (conv1): Conv2d(1, 26, kernel_size=(3, 3), stride=(1, 1))
  (conv2): Conv2d(26, 52, kernel_size=(3, 3), stride=(1, 1))
  (dropout1): Dropout(p=0.25, inplace=False)
  (dropout2): Dropout(p=0.5, inplace=False)
  (fc1): Linear(in_features=7488, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=10, bias=True)
)
Accuracy: 98.15
Inference Time: 0.0003204159498214722
```

Model 4:



```
Model 4:
Pruner Used: FPGMPruner
Configurations Used: [{'sparsity_per_layer': 0.7, 'op_types': ['Conv2d']}]
Structure:
Net(
  (conv1): Conv2d(1, 10, kernel_size=(3, 3), stride=(1, 1))
  (conv2): Conv2d(10, 20, kernel_size=(3, 3), stride=(1, 1))
  (dropout1): Dropout(p=0.25, inplace=False)
  (dropout2): Dropout(p=0.5, inplace=False)
  (fc1): Linear(in_features=2880, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=10, bias=True)
)
Accuracy: 66.79
Inference Time: 0.00022043607234954835
```

Lessons Learned:

- Generally pruning resulted in a higher inference time and lower accuracy with the best pruning (of the options above) being L1NormPruner using sparsity = .7 on convolutional layers which resulted in only a 12.86 decrease in accuracy and a 33% increase in speed for inference compared to the base model.
- Pruning for .2 sparsity on convolutional layers with L1NormPruner actually resulted in a significant increase in inference time with decrease in accuracy which is the opposite of the general pattern of decrease in inference time we would expect. Some reasons for this could be that not setting the dependency aware option in the L1NormPruner as true could have increased the time due to having to pad the pruned filters with 0(more information found here under the dependency aware mode for output pruning channels section.)

- Pruning for Linear FC layers was omitted as it was in the last layer and led to very little inference time improvement at the cost of very low accuracy drop(accuracy of model after pruning Linear FC layers was typically at 2-5 percent). It seems that a majority of the information/patterns that are used to identify numbers in the mnist seem to be located in the final 2 layers as such minimal pruning should be made in those layers to prevent significant accuracy drops.