SPANDEX Write Up

Spiking Audio Neural DEnoising eXperiment – Georgia Tech Research Institute

Author: Michael Jurado (Michael.Jurado@gtri.gatech.edu)

Overview

The SPANDEX team created an ultra-lightweight Spiking Neural Network (SNN) audio denoiser through the use of unstructured synaptic pruning. We achieved 50% and 75% weight sparsity through Gradual Magnitude Pruning (GMP) applied during a SLAYER based fine tuning step. Our submission also includes an enhancement to the `lava.lib.dl.`netx codebase which enables sparse lava-dl synapses to be translated to lava as `Sparse` synapse types. We believe other SNNs trained in lava-dl can use our weight sparsification pipeline to dramatically reduce model size and power requirements without sacrificing performance.

Previous Work and Research Background

This research builds on a previous experiment where we mapped a Spiking Neural Network (SNN) variant of vgg16 applied to the cifar10 dataset onto the Loihi1 chip using Nengo. The significant challenge we faced was the large synaptic fanout in vgg16's convolutional layers. To reduce the synaptic fanout, we decided to employ both structured and unstructured pruning techniques. This latter technique allowed us to achieve a 95% synaptic sparsity for vgg16 and to successfully port vgg16 to Loihi1.

Technical Approach

After measuring the time taken for a single epoch of the baseline SDNN solution, we realized that, due to time constraints, our SNNs could only be trained for a total of 5 epochs. Therefore, the decision was made to prune the baseline solution, which already represents one of the smallest SNN parameterizations in the competition. Any large amount of pruning applied to the model would result in a remarkably small and low-power audio denoiser.

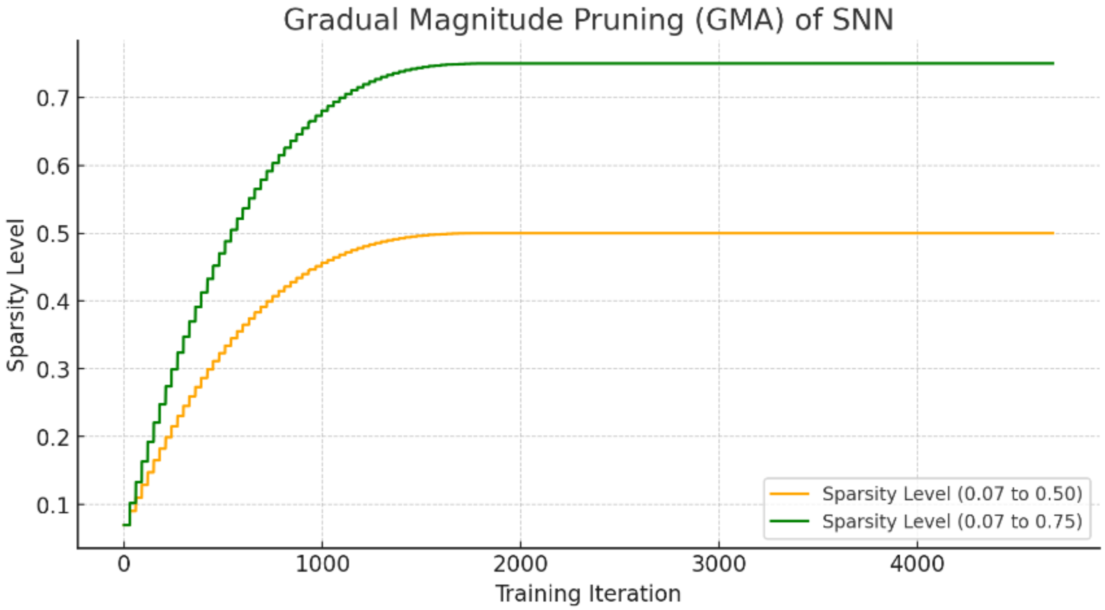
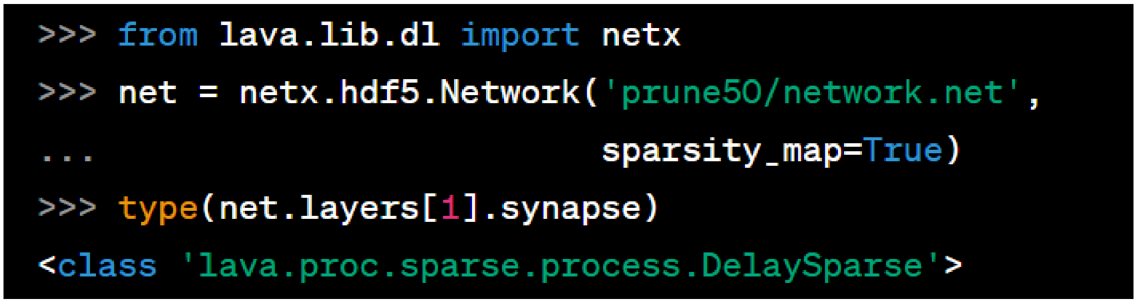


Figure 1: SDNN weight sparsity over the pruning and fine-tuning epochs. Gradual Magnitude Pruning (GMP) employs a cubic scheduler

Given our limited time budget, we decided that the iterative unstructured pruning approach previously employed for cifar10 was infeasible since it requires alternating between pruning and training epochs. Thus, a linear pruning schedule was initially adopted. This meant that every training batch of the neural network would be accompanied by a small pruning step, in which the smallest absolute magnitude weights would be set to zero. After running a single epoch of this approach, it was observed that SI-SNR was steadily decreasing over the iterations. One explanation for this is that the SNN was unable to recover lost performance due to pruning since its parameters were constantly being changed. In response, we adopted the Gradual Magnitude Pruning (GMP) technique. GMP aggressively prunes the NN early in the training and slowly transitions to the fine-tuning stage. Moreover, this technique calls for a pruning frequency parameter that allows the NN to recover performance between every pruning step.

Code Infrastructure Changes

While the `lava.lib.dl.netx` package effectively converts lava-dl SNNs to lava, it doesn't utilize sparse or sparse-delay synapses. To address this, we introduced a ‘`sparsity_map`’ parameter. This allows users to mark specific layers as sparse or to designate all synapses as sparse as shown below:



```
>>> from lava.lib.dl import netx
>>> net = netx.hdf5.Network('prune50/network.net',
...                         sparsity_map=True)
>>> type(net.layers[1].synapse)
<class 'lava.proc.sparse.process.DelaySparse'>
```

Spandex Results

Entry	SI-SNR (dB)	SI-SNRi data (dB)	MOS (ovrl)	MOS (sig)	MOS (bak)	Latency Enc+dec (ms)	Latency total (ms)	Power proxy (M-Ops/s)	PDP Proxy (M-Ops)	Params ($\times 10^2$)	Size (KB)
50% Sparse SDNN	12.16	4.80	2.69	3.19	3.45	.0056	32.006	9.32	.30	2,151	305
75% Sparse SDNN	11.72	4.36	2.68	3.24	3.29	.0060	32.006	6.06	.19	1,076	154

Replication Instructions

1. Follow standard install instructions outlined in README.md
2. pip install git+https://github.com/Michaeljurado42/lava-dl.git@sparsity_netx
3. cd baseline_solution/sdnn_delays
4. python train_sdnn_custom.py --sparsity_target .50 --epochs 5 --pruning_epochs 2 --pretrained_weights_folder Trained --trained_folder prune50
5. python train_sdnn_custom.py --sparsity_target .75 --epochs 5 --pruning_epochs 2 --pretrained_weights_folder Trained --trained_folder prune75
6. python get_model_size.py prune50 & python get_model_size.py prune75
7. python prune_evaluate.py --trained_folder prune50 --testing_set
8. python prune_evaluate.py --trained_folder prune75 --testing_set
9. python prune_inference.py --trained_folder prune50 --sparsity_map --testing_set
10. python prune_inference.py --trained_folder prune75 --sparsity_map --testing_set

This research was produced by the SPANDEX Hive IRAD at the Georgia Tech Research Institute (GTRI)