DEEP NEURAL NETWORK COMPRESSION USING HASHED NETS

Aditya Sharma

Mentors - Sergei V. Gleyzer, Lorenzo Moneta, Omar Zapata

Introduction

Motivation For HashedNets

- Datasets are growing at a MUCH faster pace than physical memory.
- The trend in deep learning is to grow models to absorb these ever-increasing data set sizes.
- These giant DNN models often don’t fit in the RAM, which leads to extensive memory swapping, in turn increasing the training time.

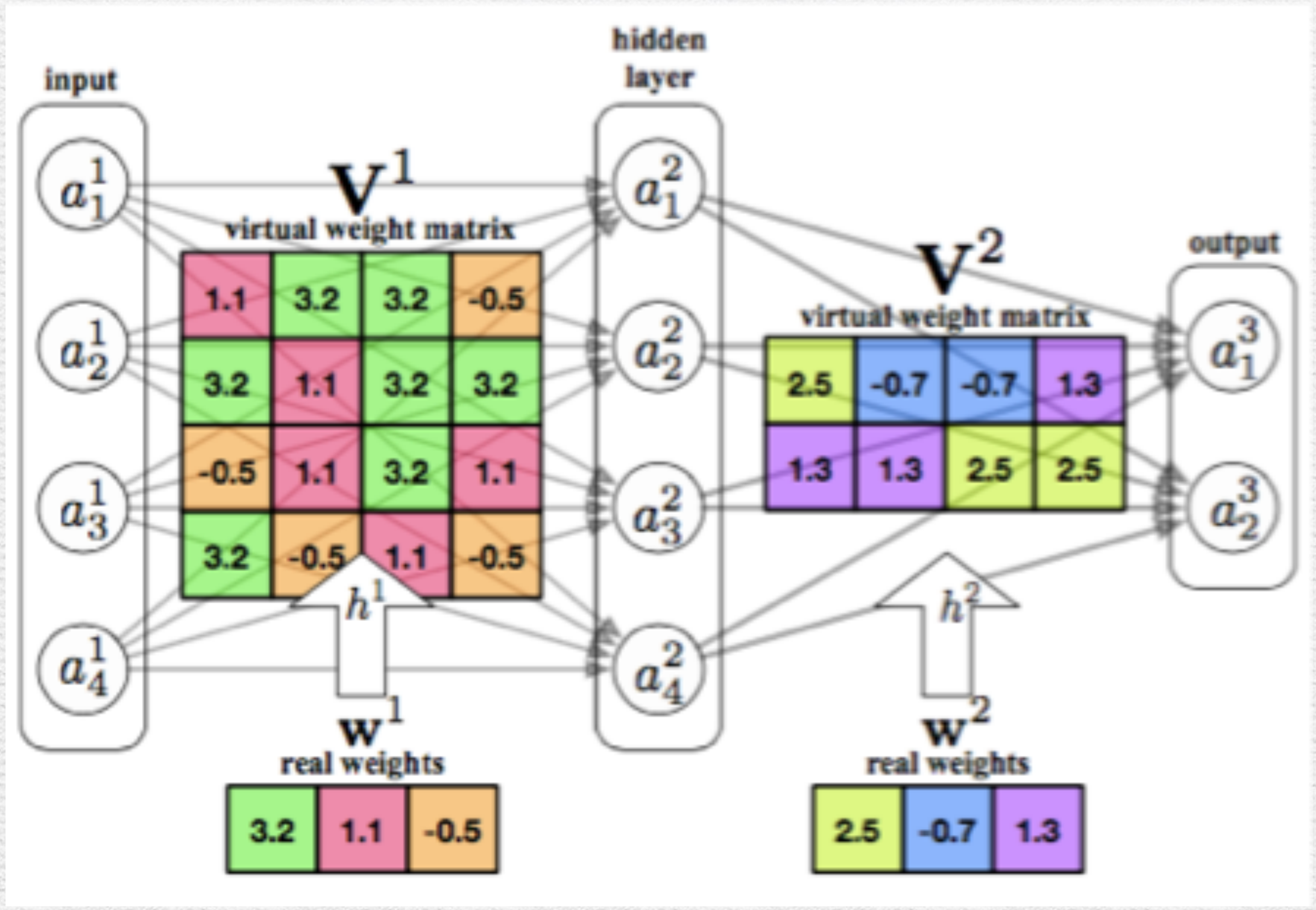
Motivation For HashedNets

- HashedNets[1] exploits inherent redundancy in neural networks to achieve drastic reductions in model sizes.
- It uses a low-cost hash function to randomly group connection weights into hash buckets, and all connections within the same hash bucket share a single parameter value.

AIM:

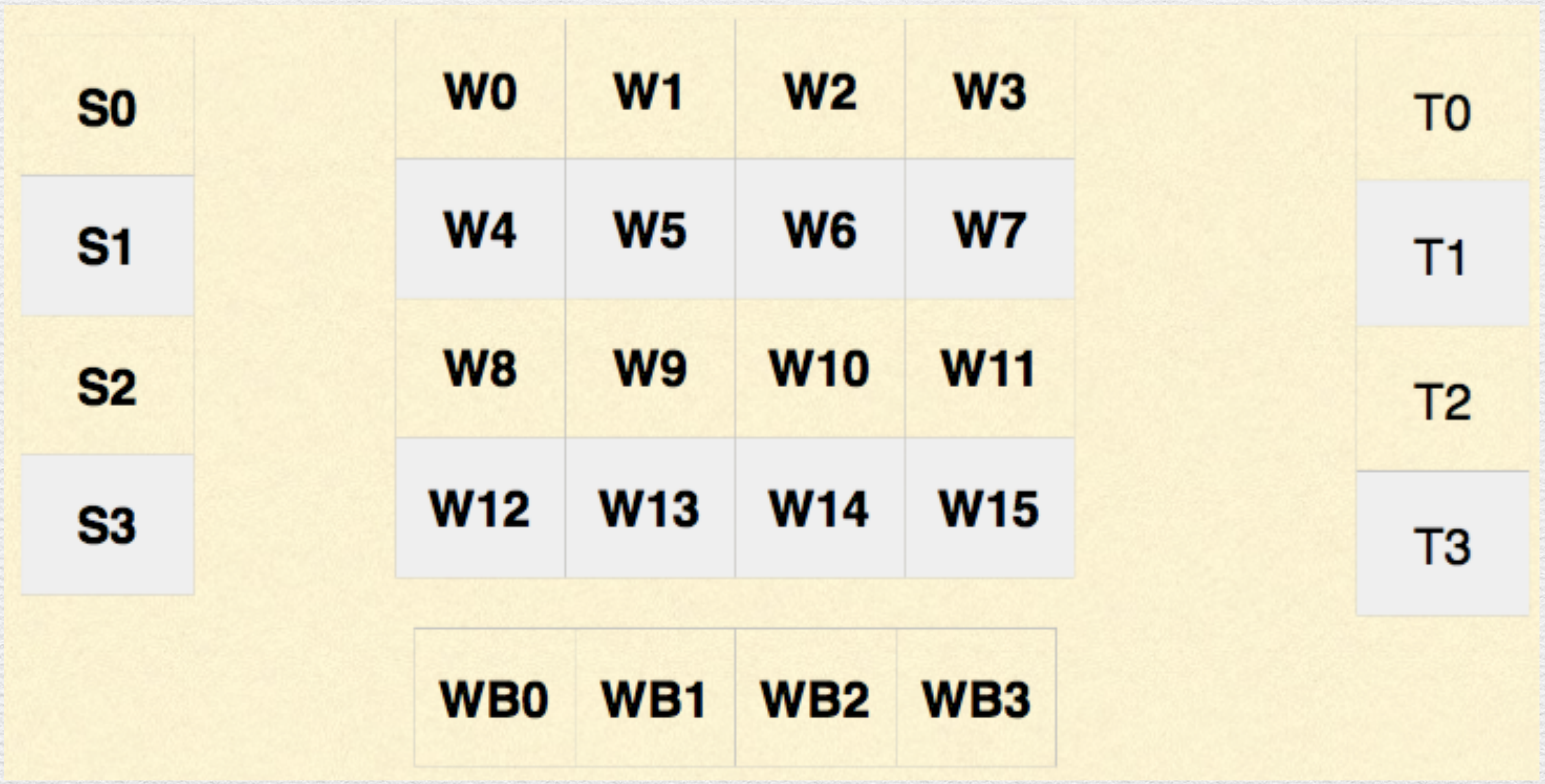
Provide an efficient and easy-to-use implementation of deep neural network model compression based on HashedNets.

HashedNets - How does it work?



- Exploits the inherent redundancy in Deep Neural Networks.
- Basically, group the weights of the DNN into hashed buckets.
- Here, the parameters in one bucket share the same value.

HashedNets - How does it work?



- The source, the target and the weightBucket vectors are real vectors, while the weight matrix is virtual.
- In the above figure, the compression ratio is 4:1.

Example - HashedNets

If an actual Deep Neural Network has the following structure with a total of 8410 Synapses:

Four layers with 100, 50, 10 and 1 nodes respectively.

Then, with Hash Nets using a Bucket Size of 8, only 32 Weight Buckets will be stored in physical memory.

Example Results - HashedNets

- The above example was trained over the entire HIGGS dataset with 11 million instances on a machine with quad core intel i3 processor and 16 GB RAM.
- The max memory consumed was 11.4 GB. Out of which dataset consumed 9.5 GB. Therefore, the model consumed roughly 1.9 GB.
- In comparison, regular DNN consumes all of the available 16 GB RAM with the same parameters, and goes beyond that as seen from huge increase in memory swaps. This causes it to run roughly a minute slower than HashedNets.
- Memory saving is significant, but the speedup is not very high here. I'm guessing the speedup will be a LOT more on GPUs, since in GPUs RAM misses are much more costly in terms of performance.

Next Step After GSoC

- Current implementation only has support for CPUs.
- Next step is creating a GPU implementation built over the GPU capable DNN created by Simon.