Dynamic Sparse Training with Structured Sparsity

Mike Lasby[1], Anna Golubeva[2,3], Utku Evci[4], Mihai Nica[5,6], Yani A. Ioannou[1]

[1]University of Calgary [2]Massachusetts Institute of Technology [3]IAIFI [4]Google Research [5]University of Guelph [6]Vector Institute for AI

Motivation

Can we learn Sparse Neural Networks (SNNs) with high generalization performance that are easy to accelerate?

- Unstructured Dynamic Sparse Training (DST) matches the generalization of dense models with 85–95% fewer weights; however, accelerating unstructured SNNs is challenging.
- Structured sparsity is easy to accelerate, but results in worse generalization performance.

Method

1. Structured RigL (SRigL) is a novel sparse-to-sparse DST method that learns a SNN with constant fan-in sparsity.
2. SRigL ablates entire neurons to match the generalization performance of unstructured sparsity at high sparsities.
3. The sparse representation learned by SRigL is parameter- and memory-efficient and amenable to real-world acceleration.

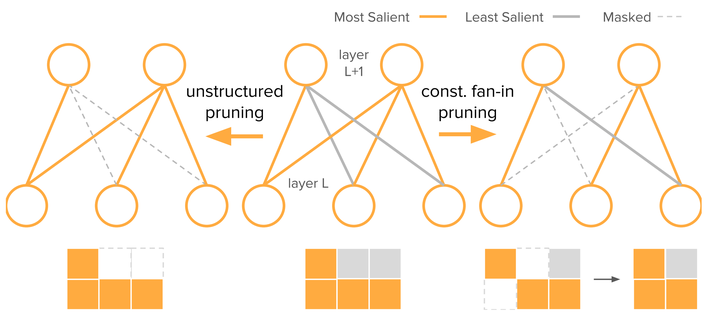

Figure 1. Constant fan-in pruning v.s. unstructured pruning.

Figure 2. Neuron ablation. At sparsity levels over 90%, RigL learns to completely mask (ablate) a large number of neurons within each layer, effectively reducing layer width. Allowing SRigL to ablate neurons restores RigL-level performance, even at high sparsities.

Constant fan-in theoretical analysis

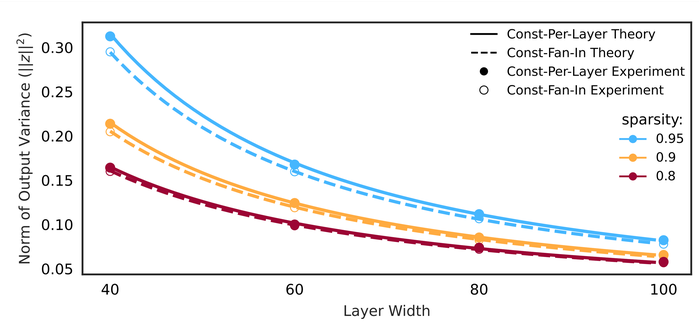

Figure 3. Output-norm variance analysis. Smaller output variance is indicative of training stability. Constant fan-in consistently yields smaller variance than other sparsity types.

Results

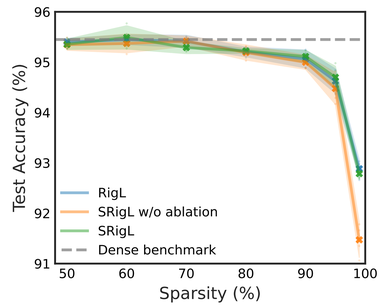

Figure 4. ResNet-18/CIFAR-10

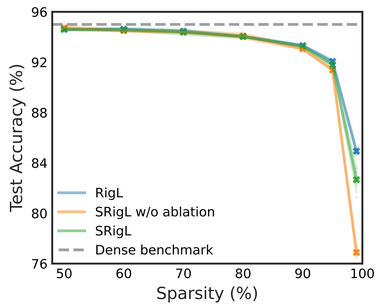

Figure 5. Wide ResNet-22/CIFAR-10

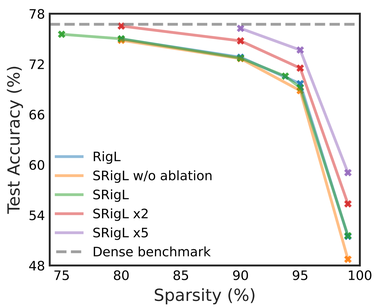

Figure 6. ResNet-50/ImageNet

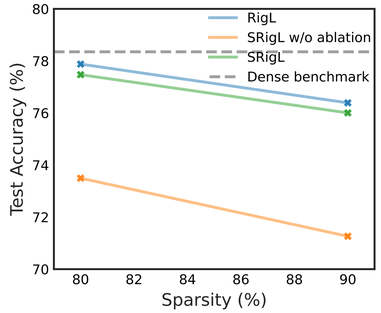

Figure 7. Vision Transformer (ViT-B/16)/ImageNet

Neuron ablation

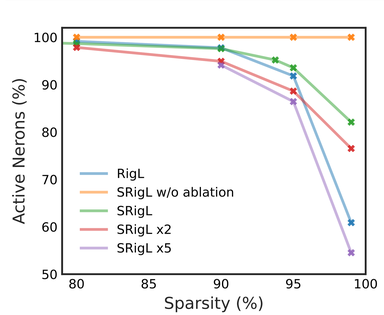

Figure 8. Percentage active neurons (i.e., not ablated) following RigL/SRigL training on ResNet-50/ImageNet.

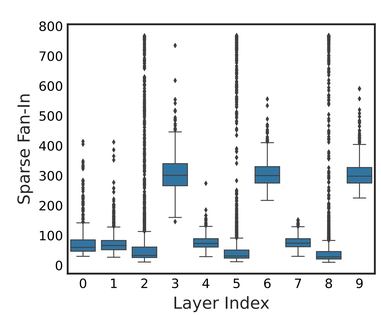

Figure 9. Sparse Fan-In vs. ViT-B/16 layer index at the end of training with RigL at 90% sparsity

Acceleration of constant fan-in sparsity

Algorithm 1 "Condensed" linear layer with constant fan-in sparsity forward pass

```
Input: x: the input matrix of shape (batch_size, num_features)
       w: the condensed weight matrix of shape (active_neurons, constant_fan_in)
       indx: indices of non-zero dense weights of shape (active_neurons, constant_fan_in)
output ← torch.zeros(size=(batch_size, neurons))
for b in range(batch_size) do                          ▷ For each sample in mini-batch
    for n in range(neurons) do                         ▷ For each active neuron in layer
        for k in range(constant_fan_in) do             ▷ For each non-zero weight
            source_idx ← idx[n, k]
            feature ← x[b, source_idx]
            output[b, n] += feature * w[n, k]
        end for
    end for
end for
return output
```

Benchmarks

- To benchmark our method, we extracted a linear layer from a ViT-B/16 model trained with SRigL and compared it to structured (i.e. the same layer accelerated using only the ablated neurons without exploiting the fine-grained sparsity), and unstructured (i.e. CSR) representations.
- The increased timings for the 95 & 99% sparse structured representations is due to SRigL ablating relatively fewer neurons at these sparsities compared to 80 and 90%.

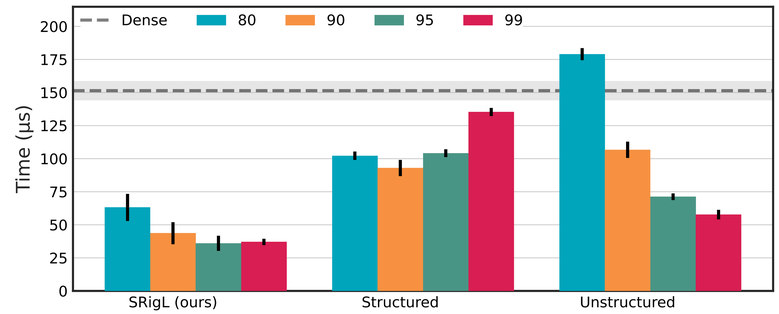

Figure 10. Online CPU inference on an Intel Xeon W-2145. For online (single input) inference, our condensed representation at 90% is *3.4× faster than dense* and *2.5 × faster than unstructured sparsity*

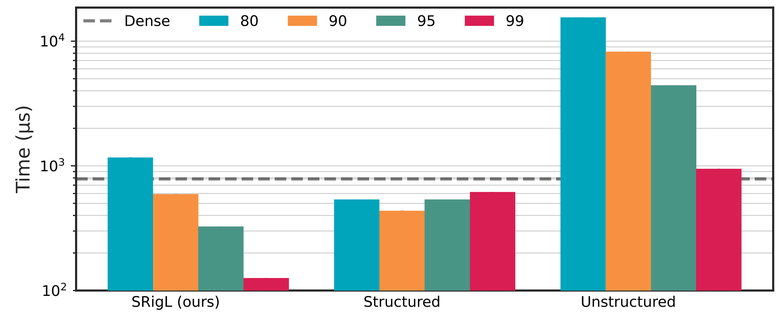

Figure 11. Batched GPU inference with batch size of 2048 on an NVIDIA Titan V. At 90% sparsity, our condensed representation is *1.7× faster than dense* and *13.0× faster than unstructured (CSR)* sparse layers. Note y-axis is log-scaled.