Mini-Project: Pruning

In our project, we implemented three different methods for neural network pruning from scratch: L1-norm based filter pruning, activation-based pruning, and sparsity-informed adaptive pruning.

Methodology

L1-norm based filter pruning:

In L1-norm based filter pruning, we implement L1-norm structured pruning with iterative fine-tuning. The approach first removes less important filters/neurons by zeroing weights with the smallest L1 magnitudes (sum of absolute values). For convolutional layers, norms are computed across all input channels and spatial dimensions, while linear layers use input dimension sums.

The pruning occurs gradually through multiple cycles, each removing 10% of remaining weights followed by fine-tuning to recover accuracy. This iterative process continues until reaching the target sparsity (30%) or maximum iterations.

A comprehensive evaluation tests pruning rates from 3-40%, keeping only models that maintain >60% accuracy while achieving a favorable accuracy-sparsity balance (score >0.36). The method provides practical compressed models while systematically analyzing the sparsity-accuracy tradeoff.

Activation based pruning:

Activation-based pruning removes neurons or filters that rarely activate during training, under the assumption they contribute little to model predictions.

We applied this to both linear and convolutional layers by computing the average activation over a training subset and pruning units below a threshold by zeroing their weights and biases. After pruning, fine-tuning was conducted to regain accuracy. This process of pruning and fine tuning was repeated multiple times either using the same threshold (15%) or varying the threshold (10 - 35%) to increase sparsity over runs while maintaining accuracy.

Our implementation closely followed algorithms 1, 2, and 3 as seen in “Activation-Based Pruning of Neural Networks” Ganguli & Chong (2024). Like their method, we use an iterative prune-and-fine-tune loop but unlike their activation count method, we use average activation magnitude to score importance.

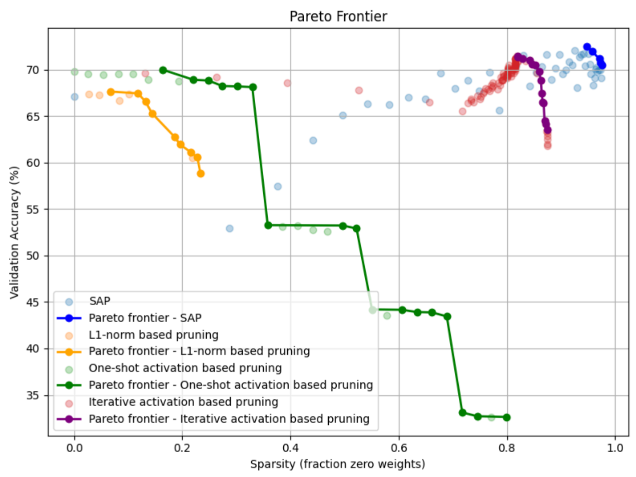
Sparsity-informed Adaptive Pruning (SAP)

Sparsity-informed Adaptive Pruning (SAP) is an algorithm based on a quantifiable measure to estimate the compressibility of a sub-network during each pruning iteration. Specifically, PQ Index (PQI) measures the potential compressibility of deep neural networks. SAP adaptively determines the number of pruned parameters at each iteration based on the PQI-bound. Our implementation is based on Diao, Enmao, et al. "Pruning deep neural networks from a sparsity perspective." *arXiv preprint arXiv:2302.05601* (2023).

Algorithm 1 Sparsity-informed Adaptive Pruning (SAP)

```
Input: model parameters $w$, mask $m$, norm $0 < p \leq 1 < q$, compression hyper-parameter $\eta_r$,
       scaling factor $\gamma$, maximum pruning ratio $\beta$, number of epochs $E$, and number of pruning
       iterations $T$.
Randomly generate model parameters $w_{\text{init}}$
Initialize mask $m_0$ with all ones
for each pruning iteration $t = 0, 1, 2, \ldots T$ do
    Initialize model parameters $\tilde{w}_t = w_{\text{init}} \odot m_t$
    Compute the number of model parameters $d_t = |m_t|$
    Train the model parameters $\tilde{w}_t$ with $m_t$ for $E$ epochs and arrive at $w_t$
    Compute PQ Index $\mathrm{I}(w_t) = 1 - d_t^{\frac{1}{q} - \frac{1}{p}} \frac{\|w_t\|_p}{\|w_t\|_q}$
    Compute the lower bound of the number of retained model parameters
    $r_t = d_t(1 + \eta_r)^{-q/(q-p)}[1 - \mathrm{I}(w_t)]^{\frac{qp}{q-p}}$
    Compute the number of pruned model parameters
    $c_t = \lfloor d_t \cdot \min(\gamma(1 - \frac{r_t}{d_t}), \beta) \rfloor$
    Prune $c_t$ model parameters with the smallest magnitude based on $w_t$ and $m_t$
    Create new mask $m_{t+1}$
end
Output: The pruned model parameters $w_T$ and mask $m_T$.
```

Performance Comparison



Among the three methods, the sparsity-informed adaptive pruning algorithm led to the best overall model score, with an accuracy of 70.84%, a sparsity of 96.43%, and a final score of 0.8345. The L1-norm based filter pruning was easiest to implement.

The plot shows that **SAP** achieves the best sparsity–accuracy trade-off, consistently outperforming all other methods across the full range of sparsity levels. **L1-norm pruning** maintains moderate accuracy at low sparsity. **One-shot activation-based pruning** over a range of thresholds from 0 to 100% pruning enables very high sparsity but suffers from large accuracy drops early, while **iterative activation-based pruning** using a constant threshold of 15% improves stability compared to one-shot but still falls short of SAP. Overall, SAP dominates the Pareto frontier, offering both higher compression and better preservation of model performance.

Reflection

As expected, the L1-norm pruning successfully compressed the model while maintaining reasonable accuracy. However, we see a steep drop of accuracy beyond 30% sparsity than we initially anticipated, which shows the limits of magnitude-based pruning.

In contrast, activation-based pruning demonstrated greater robustness at higher sparsity levels. By targeting neurons and filters with consistently low activation, this method first prunes the dense linear layers where the majority of parameters are concentrated, followed by the convolutional layers. This strategy achieved up to 85.94% sparsity while maintaining strong accuracy of 69.82%, indicating that activation statistics can serve as a more effective criterion for identifying redundancy than magnitude based approaches and this is what we expected.

The SAP algorithm shows the best performance with an accuracy of 70.84%, a sparsity of 96.43%, and a final score of 0.8345. The results highlight the effectiveness of adaptive pruning based on the compressibility of a sub-network and align with our expectation.

Key Lessons We Learned:

There's always a trade-off to consider. When you increase sparsity, accuracy tends to drop, but how much it drops really depends on the method you use. L1 pruning is straightforward and easy to implement, while SAP gives you more control over the process.

Also, fine-tuning is more crucial than we think. Even models that have been well-pruned still require a good amount of retraining to recover performance. The iterative process of pruning and then fine-tuning turned out to be critical.

Moreover, not all weights carry the same importance. Activation pruning showed that some weights that seem "small" in magnitude can actually play a significant role when considering their activation patterns. This challenges the idea that relying solely on L1-norm is sufficient.

We also realized that structured pruning does have its limits. While it can enhance efficiency by removing entire filters, unstructured pruning, like SAP, can sometimes achieve a higher level of sparsity without losing as much accuracy.