ADMM Pruning for Efficient Deep Learning

Tony Tran
University of Houston
4302 University Dr
thtran37@cougarnet.uh.edu

Abstract

Deep neural networks (DNNs) deliver strong performance across many domains but are often too computationally expensive for resource-constrained deployment. Weight pruning reduces model size by removing unimportant parameters, with magnitude-based pruning serving as a simple and effective baseline despite its heuristic nature. In contrast, the Alternating Direction Method of Multipliers (ADMM) provides a principled optimization framework that enforces sparsity through constrained training. This project implements ADMM-based pruning on a lightweight DNN and compares its performance against magnitude pruning. Results highlight the trade-offs between heuristic and optimization-driven approaches.

1. Introduction

Deep Neural Network (DNN) models are everywhere and have grown increasingly popular due to their ability to learn complex patterns and achieve considerable good performance across various domains. However, strong performance comes at the cost of high computation and memory demands which becomes a challenge for deploying DNNs on resource-constrained devices.

To address these challenges, model compression techniques. such as weight pruning. have become effective strategies to reduce model complexity while preserving performance. Weight pruning works by removing weights of less importance, effectively decreasing the model size and computational load while enabling more efficient inference. Traditional heuristic approaches, such as magnitude-based pruning [1], prove to be effective by pruning weights with the smallest magnitude under the assumption that they will have minimal impact on the final output of the model, but they lack a solid mathematical foundation. Alternating Direction Method of Multipliers (ADMM) [2, 3] offers an optimization-based framework for weight pruning, formulating pruning as a constrained optimization problem minimizing the DNN's loss function while enforcing sparsity constraints on the weights.

This project will focus on implementing ADMM on lightweight DNNs and compare the pruned DNNs performance against the magnitude-based pruning baseline. To complete the project, we will first formulate the problem and understand the method presented in [2, 3], apply the method to a tiny DNN, and compare with the baseline method [4].

2. Related Work

Research on model compression has expanded significantly as DNNs have grown in scale and computational demand. Magnitude-based pruning [4] is one of the most widely adopted heuristic approaches. By removing weights with the smallest absolute values, it assumes such parameters contribute least to the network's predictive capability. Formally, let $f(\{W\}\{b\})$ be the objective function of our model learning parameterized by model weights W, typically the cross-entropy loss, let $card(W)$ be the number of nonzero elements in weight matrix W, and l be the target number of weights characterized by the pruning ratio s, i.e. $l = s \times |W|$ where $|W|$ is the number of elements in matrix W. Then the objective of this method becomes

$$\min_{W} f(W) \quad (1)$$
$$\boldsymbol{s.t.}\ card(W) \leq l \quad (2)$$

Enforcing the hard constraint by pruning W after every step. To address these limitations, ADMM-based pruning frameworks adopt a more principled, optimization-based approach [3]. They first reformulate equation 1 into the optimization problem

$$\min_{W} f(W) + g(Z) \quad (3)$$
$$\boldsymbol{s.t.}\ W = Z \quad (4)$$

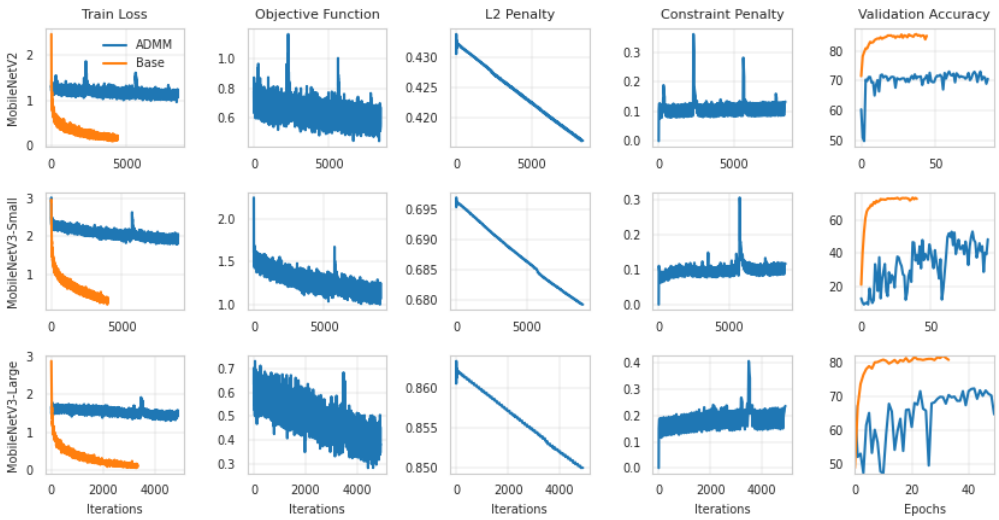

Figure 1. Training Objective and Validation Accuracy

Where $g(\cdot)$ is the indicator function of the sparsity set $S = \{W|card(W) \leq l\}$ in which

$$g(W) = \begin{cases} 0 & if\ card(W) < l, \\ +\infty & otherwise. \end{cases} \quad (5)$$

And Z is an auxiliary variable. The Augmented Lagrangian of equation 3 is then

$$L_p(W, Z, U) = f(W) + g(Z) + \frac{\rho}{2}\|W - Z + U\|_F^2 - \frac{\rho}{2}\|U\|_F^2 \quad (6)$$

Where U are the dual variables and $\|\cdot\|_F$ is the Frobenius norm of a matrix. The ADMM Algorithm [3] proceeds by iterative updating

$$W^{k+1} := \underset{W}{\operatorname{argmin}}\, L_p(W, Z^k, U^k) \quad (7)$$

$$Z^{k+1} := \underset{Z}{\operatorname{argmin}}\, L_p(W^{k+1}, Z, U^k) \quad (8)$$

$$U^{k+1} := U^k + W^{k+1} - Z^{k+1} \quad (9)$$

We can solve equation 7 by gradient descent and equation 8 analytically by projecting Z onto the sparsity set S or

$$Z^{k+1} = \Pi_S(W^{k+1} + U^k) \quad (8)$$

Where $\Pi_S(\cdot)$ denotes the Euclidean projection onto the set S.

Recent work also propose a symmetric accelerated stochastic ADMM [2] to improve convergence speed and sparsity quality.

3. Data

We aim to use the CIFAR10 dataset [5] to validate our experiments which can be used for classification of tiny images. It contains 10 mutually exclusive classes of airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck with 50k training images and 10k test images with 6k images per class, 5k images for training and 1k images for testing in 32x32 RGB. We preprocess this dataset with random cropping and horizontal flip.

We chose this dataset due to the low computation cost of conducting our experiments. While a larger scale dataset

Base/ADMM Performance Before/After Pruning												
	Base						ADMM					
Model	Before Pruning			After Pruning			Before Pruning			After Pruning		
	Acc	Params	Iters	Acc	Params	Iters	Acc	Params	Iters	Acc	Params	Iters
MobileNetV2	85.06	2.24M	4.4k	67.10	1.13M	12.8k	70.60	1.63M	8.4k	72.08	1.13M	2.6k
MobileNetV3-L	80.66	4.21M	3.3k	68.48	2.12M	5.6k	64.57	4.14M	4.9k	60.13	2.12M	2.9k
MobileNetV3-S	72.47	1.53M	4.0k	10.00	0.77M	6.5k	48.33	1.49M	9.0k	35.97	0.77M	4.2k

Table 1. Main Results Base vs ADMM Method

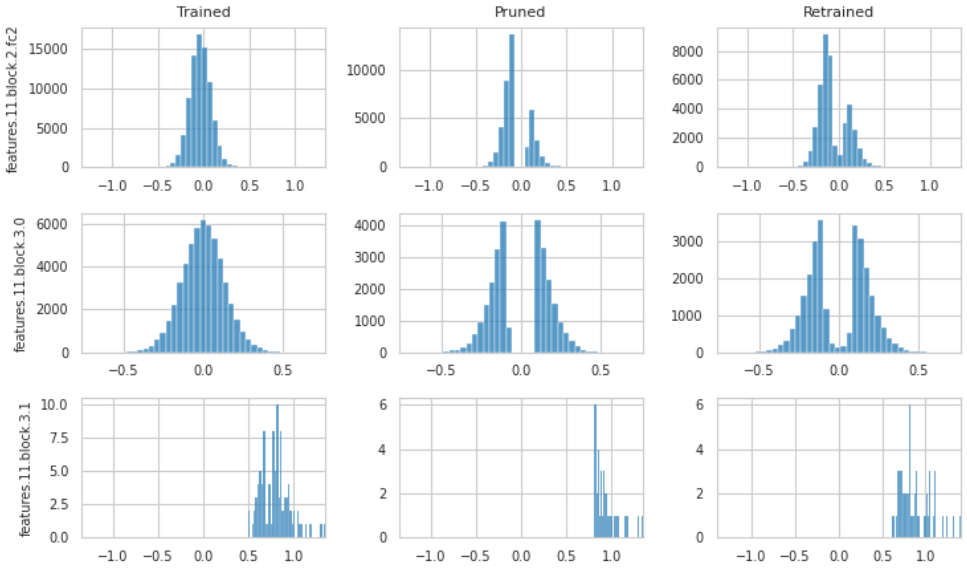

Figure 2. Weight Distribution of Base Method

would better validate our method, like ImageNet, the given time constraint does not allow us to fully explore these datasets. We can also validate the method quickly in a short time.

4. Methods

The pruning framework presented in [3] demonstrates the effectiveness of ADMM-based sparsification but is validated primarily on outdated classifiers such as LeNet and early convolutional architectures. These models are far removed from modern lightweight designs and do not reflect the challenges faced in deploying efficient deep networks. To address this gap, we apply the ADMM pruning method to contemporary mobile architectures such as MobileNetV2, MobileNetV3-Small, and MobileNetV3-Large, which are widely used in resource-constrained environments. This setup allows us to evaluate not only pruning effectiveness but also convergence behavior on compact, state-of-the-art models.

For each architecture, we integrate the ADMM formulation into the training pipeline, alternating between loss minimization and sparsity-enforcing projections. We compare its performance against the classical magnitude-based pruning baseline to assess accuracy retention, sparsity structure, and training stability. Extensive experiments are conducted following the procedures outlined in Section 5, enabling a thorough analysis of convergence rate, robustness to compression, and overall model efficiency across different pruning ratios.

5. Experiments

For all experiments, we use a uniform pruning ratio $s = 0.5$ across all layers, starting from PyTorch ImageNet pretrained MobileNet models. Training uses the cross-entropy loss for the baseline and the ADMM objective in equation 6 with $\rho = 0.01$. Equation 7 is solved using the Adam optimizer with learning rate of 0.001.

Figure 1 presents the training curves for both methods along with their corresponding validation accuracies. The ADMM loss remains higher throughout training and its validation accuracy is consistently lower compared to the base method. Moreover, ADMM converges slower, indicating the increased optimization difficulty under the imposed sparsity constraints.

Table 1 summarizes the main results before and after pruning for both baseline and ADMM methods. In terms of pre-pruning accuracy, the baseline consistently outperforms ADMM, which is expected given that the ADMM introduces additional constraints that make optimization more difficult. However, after pruning, ADMM achieves higher accuracy than the baseline for most models, suggesting that its structured sparsity may help preserve important weights. But, for MobileNetV3-L, the baseline achieves substantially higher post-pruning accuracy than ADMM. A plausible explanation is that the larger model size amplifies the optimization challenge of ADMM, causing the method to break down under increased architectural complexity.

Finally, Figures 3 and 4 show the weight distributions of three randomly selected layers from MobileNetV3-Small for the baseline and ADMM methods, respectively. The baseline exhibits an approximately normal weight distribution, and pruning primarily removes weights clustered near zero. After retraining, the distribution remains similar, with fewer weights collapsing toward zero. In contrast, the ADMM-trained model shows that

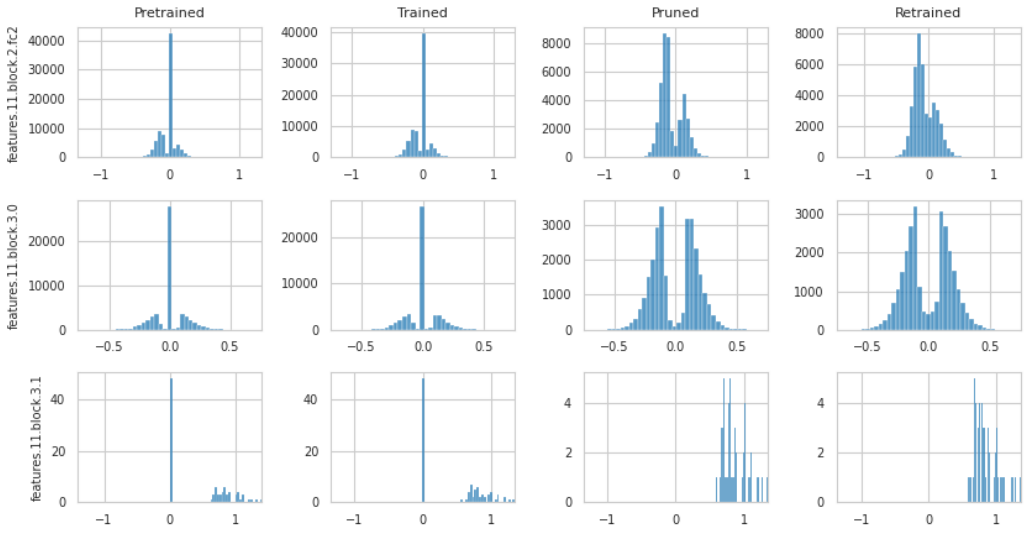

Figure 3. Weight Distribution of ADMM Method

many weights are driven exactly to zero during optimization, reflecting the enforced sparsity constraints. After pruning, the ADMM distributions become more multimodal, suggesting that the method induces sharper separation between important and unimportant weights compared to the baseline.

6. Conclusion

This project examined the effectiveness of ADMM-based pruning on modern lightweight architectures and compared its performance to the classical magnitude-based pruning baseline. While ADMM provides a principled optimization framework for enforcing structured sparsity, our experiments show that it introduces significant optimization challenges for compact models such as MobileNetV2 and MobileNetV3. ADMM consistently exhibited higher training loss, slower convergence, and lower pre-pruning accuracy, indicating the difficulty of jointly minimizing task loss while satisfying strict sparsity constraints. Nevertheless, after pruning, ADMM outperformed the baseline on several architectures, suggesting that its constraint-driven formulation can better preserve important weights under moderate compression. However, this advantage did not hold for larger models such as MobileNetV3-L, where ADMM failed to maintain accuracy, likely due to increased architectural complexity exacerbating optimization instability. Analysis of weight distributions further highlights fundamental behavioral differences between the two methods, with ADMM producing sharper sparsity patterns but less stable convergence. Overall, our results demonstrate that ADMM has potential for structured pruning but is not yet reliably effective for all lightweight architectures, motivating future work on improved optimization strategies and adaptive sparsity constraints

References

[1] Cheng, H., Zhang, M., & Shi, J. Q. (2024). A Survey on Deep Neural Network Pruning: Taxonomy, Comparison, Analysis, and Recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *46*(12), 10558–10578. https://doi.org/10.1109/TPAMI.2024.3447085.

[2] Yuan, M., Bai, J., Jiang, F., & Du, L. (2024). A systematic DNN weight pruning framework based on symmetric accelerated stochastic ADMM. *Neurocomputing*, *575*, 127327. https://doi.org/10.1016/j.neucom.2024.127327.

[3] Zhang, T., Ye, S., Zhang, K., Tang, J., Wen, W., Fardad, M., & Wang, Y. (2018). *A Systematic DNN Weight Pruning Framework using Alternating Direction Method of Multipliers*. 184–199. https://openaccess.thecvf.com/content_ECCV_2018/html/Tianyun_Zhang_A_Systematic_DNN_ECCV_2018_paper.html.

[4] Han, S., Pool, J., Tran, J., & Dally, W. J. (2015). Learning both weights and connections for efficient neural networks. arXiv preprint arXiv:1506.02626.

[5] Krizhevsky, A. (2009). Learning multiple layers of features from tiny images (Technical Report). University of Toronto.