Surrogate Lagrangian Relaxation: A Path to Retrain-Free Deep Neural Network Pruning

SHANGLIN ZHOU, MIKHAIL A. BRAGIN, DENIZ GUREVIN, LYNN PEPIN, FEI MIAO, and CAIWEN DING, University of Connecticut, USA

Network pruning is a widely used technique to reduce computation cost and model size for deep neural networks. However, the typical three-stage pipeline (i.e., training, pruning, and retraining (fine-tuning)) significantly increases the overall training time. In this article, we develop a systematic weight-pruning optimization approach based on surrogate Lagrangian relaxation (SLR), which is tailored to overcome difficulties caused by the discrete nature of the weight-pruning problem. We further prove that our method ensures fast convergence of the model compression problem, and the convergence of the SLR is accelerated by using quadratic penalties. Model parameters obtained by SLR during the training phase are much closer to their optimal values as compared to those obtained by other state-of-the-art methods. We evaluate our method on image classification tasks using CIFAR-10 and ImageNet with state-of-the-art multi-layer perceptron based networks such as MLP-Mixer; attention-based networks such as Swin Transformer; and convolutional neural network based models such as VGG-16, ResNet-18, ResNet-50, ResNet-110, and MobileNetV2. We also evaluate object detection and segmentation tasks on COCO, the KITTI benchmark, and the TuSimple lane detection dataset using a variety of models. Experimental results demonstrate that our SLR-based weight-pruning optimization approach achieves a higher compression rate than state-of-the-art methods under the same accuracy requirement and also can achieve higher accuracy under the same compression rate requirement. Under classification tasks, our SLR approach converges to the desired accuracy 3× faster on both of the datasets. Under object detection and segmentation tasks, SLR also converges 2× faster to the desired accuracy. Further, our SLR achieves high model accuracy even at the hardpruning stage without retraining, which reduces the traditional three-stage pruning into a two-stage process. Given a limited budget of retraining epochs, our approach quickly recovers the model's accuracy.

CCS Concepts: • **Computer systems organization → Real-time system architecture;**

Additional Key Words and Phrases: Surrogate Lagrangian relaxation, model compression, weight pruning, image classification, object detection and segmentation

ACM Reference format:
Shanglin Zhou, Mikhail A. Bragin, Deniz Gurevin, Lynn Pepin, Fei Miao, and Caiwen Ding. 2023. Surrogate Lagrangian Relaxation: A Path to Retrain-Free Deep Neural Network Pruning. *ACM Trans. Des. Autom. Electron. Syst.* 28, 6, Article 102 (October 2023), 19 pages.
https://doi.org/10.1145/3624476

This work was partially supported by the Semiconductor Research Corporation (SRC) Artificial Intelligence Hardware program, the National Science Foundation (grant CNS-2047354), and the USDA-NIFA Agriculture and Food Research Initiative (award 2022-67023-36399).

Authors' address: S. Zhou, M. A. Bragin, D. Gurevin, L. Pepin, F. Miao, and C. Ding, University of Connecticut, 371 Fairfield Way, Storrs, CT 06269; e-mails: shanglin.zhou@uconn.edu, mikhail.bragin@uconn.edu, deniz.gurevin@uconn.edu, lynn.pepin@uconn.edu, fei.miao@uconn.edu, caiwen.ding@uconn.edu.

1084-4309/2023/10-ART102 $15.00
https://doi.org/10.1145/3624476

1 INTRODUCTION

Deep Neural Network (DNN)-based statistical models are increasingly demanding of computational and storage resources, with costs proportional to the model size (i.e., the number of parameters in a model). This resource consumption is especially an issue for embedded or IoT devices [16]. By reducing model size, one can decrease both storage costs and computation costs when evaluating a model. Various techniques exist for reducing model size while maintaining its performance, such as weight pruning, sparsity regularization, quantization, and clustering. These techniques are collectively known as *model compression* [13, 28, 29, 32, 33, 49].

These works leverage the observation that training a compact model from scratch is more difficult and less effective than retraining a pruned model [9, 26]. Therefore, a typical three-stage pipeline has been used: training (large model), pruning, and retraining (also called *fine-tuning*). The pruning process involves setting the redundant weights to zero while keeping the important weights to maintain performance. The retraining process is necessary since the model accuracy significantly decreases after hardpruning. However, this three-stage weight pruning approach substantially adds to the overall training cost. For example, although the state-of-the-art weight pruning methods achieve a very high compression rate while maintaining the prediction accuracy on many DNN architectures, the retraining process requires more time—for example, 80 epochs for ResNet-18 on ImageNet, which is 70% of the original training epochs using the **Alternating Direction Method of Multipliers (ADMM)** [38, 50].

Given the pros and cons of the current weight pruning method, this article aims to answer the following questions: Is there an optimization method that can achieve high model accuracy even at the hardpruning stage and can significantly reduce retraining trails? Given a limited budget of retraining epochs, is there an optimization method that can rapidly recover model accuracy (much faster than the state-of-the-art methods)?

The primary obstacle in addressing these questions is the discrete nature of the model compression problems caused by "cardinality" constraints, which ensure that a certain proportion of weights is pruned. In this work, we develop a weight-pruning optimization approach based on recent **Surrogate Lagrangian Relaxation (SLR)** [6], which overcomes all major convergence difficulties of standard Lagrangian relaxation. Within the SLR approach, Lagrangian multipliers converge to their optimal values much faster as compared to those within other methods (e.g., ADMM).

We summarize our contributions/findings as follows:

- We adapt the SLR-based approach to overcome difficulties caused by the discrete nature of the weight-pruning problem while ensuring fast convergence.
- We use quadratic penalties to further accelerate the SLR convergence. The method possesses nice convergence properties inherited from the rapid reduction of constraint violations owing to quadratic penalties, and quadratic penalties ultimately lead to faster convergence. In addition, unlike previous methods such as ADMM, the SLR method guarantees convergence, thereby leading to unmatched performance compared to other methods. Therefore, model parameters obtained by SLR are much closer to their optimal values as compared to those obtained by other state-of-the-art methods.
- We provide a convergence proof of the SLR method for weight-pruning problems. Existing coordination-based weight pruning approaches do not converge when solving non-convex problems. Other coordination techniques (e.g., ADMM) are not designed to handle discrete variables and other types of non-convexities.
- Our proposed SLR-based model compression method achieves high model accuracy even at the hardpruning stage using our SLR-based weight-pruning optimization approach; given a limited budget of retraining epochs, our method quickly recovers the model accuracy.

We conduct comprehensive experiments on various tasks and datasets to further prove the effectiveness of our proposed SLR-based model compression method. For classification tasks, we test our method on not only **Convolutional Neural Network (CNN)**-based models like VGG-16, ResNet-18, ResNet-50, ResNet-110, and MobileNetV2 but also on non-CNN based models such as MLP-Mixer, a **Multi-Layer Perceptron (MLP)**-based network, and Swin Transformer, an attention-based network. We also test and compare our SLR method with other state-of-the-art pruning methods on segmentation and detection tasks. Our experiments involve various dataset benchmarks like CIFAR-10, ImageNet, COCO, KITTI, and TuSimple. The results demonstrate that our proposed SLR method outperforms the state-of-the-art compression methods. Our method converges 3× faster to the desired accuracy on both CIFAR-10 and ImageNet datasets under both CNN-based and non-CNN-based classification tasks, and 2× faster on COCO object detection tasks. Moreover, up to a 6% accuracy gap can be achieved between SLR and the state of the art at the hardpruning stage under classification tasks, and a 44% accuracy gap in object detection and segmentation tasks.

2 RELATED RESEARCH

2.1 Model Compression

Given the increasing computational and storage demands of DNNs, model compression has become increasingly essential when we implement highly efficient deep learning applications in the real world. There are two common compression techniques: weight pruning and weight quantization. As numerous researchers have investigated that some portion of weights in neural networks are redundant, weight pruning aims to remove these less important coefficient values and achieves model compression while maintaining performance similar to the uncompressed model. Structured and non-structured (irregular) weight pruning are two mainstream methods. Weight quantization is another technique that reduces weight storage by decreasing the number of bits used to represent weights.

In early work [11], the researchers proposed an iterative irregular weight pruning method where most reductions are achieved in fully connected layers, and the reduction achieved in convolutional layers can hardly achieve significant acceleration in GPUs. For weight storage, it reduces 9× the number of parameters in AlexNet and 13× in VGGNet-16. To address the limitation in irregular weight pruning, structured weight pruning methods were proposed by Wen et al. [44], which investigated structured sparsity at the levels of filters, channels, and filter shapes. However, the overall compression rate in structured pruning is limited compared to unstructured pruning. In AlexNet without accuracy degradation, the average weight pruning rate in convolutional layers is only 1.4×. The recent work [14] achieved 2× channel pruning with a 1% accuracy degradation on VGGNet-16. Later, Louizos et al. [27] proposed a framework for L_0 norm regularization for neural networks, aiming to prune the network during training by selecting weights and setting them to exactly zero. Frankle and Carbin [9] introduced the lottery ticket hypothesis, which observes that a subnetwork of a randomly initialized network can replace the original network with the same performance. In this work, our focus is on irregular pruning, which can achieve much higher accuracy compared to structured pruning [44] due to its flexibility in selecting weights [52].

2.2 Alternating Direction Method of Multipliers

ADMM is an optimization algorithm that breaks optimization problems into smaller subproblems, each of which is then solved iteratively and more easily. The early studies of ADMM can be traced back to the 1970s, and a variety of statistical and machine learning problems that can be efficiently solved by using ADMM have been discussed [3]. Recently, weight pruning studies have

achieved a high compression rate and avoided significant accuracy loss by integrating the powerful ADMM. The successful applications with ADMM outperform prior approaches by applying dynamic penalties on all targeted weights. The algorithm can be applied to various schemes of both non-structured pruning and structured pruning. The work of Zhang et al. [50] was the first implementing an ADMM-based framework on DNN weight pruning, achieving 21× irregular weight pruning with almost no accuracy loss in AlexNet. A pattern-based weight pruning approach was proposed with high efficiency specifically designed and optimized for mobile devices [31]; it explored a fine-grained sparsity to maximize the utilization of devices with limited resources. Li et al. [22] improved the previous ADMM-based structured weight pruning framework by adopting a soft constraint based formulation to achieve a higher compression rate and tune fewer hyperparameters.

3 WEIGHT PRUNING USING SLR

Consider a DNN with N layers indexed by $n \in 1, ..., N$, where the weights in layer n are denoted by $\mathbf{W}_n$. The objective is to minimize a loss function

$$\min_{\mathbf{W}_n} \left\{ f\left(\mathbf{W}_n\right) \right\} \tag{1}$$

subject to constraints on the cardinality of weights within each layer n, where the number of non-zero weights should be less than or equal to the pre-defined number l_n. This constraint can be captured using an indicator function $g_n(\cdot)$ as follows:

$$g_n(\mathbf{W}_n) = \begin{cases} 0 & \text{if } \mathrm{card}(\mathbf{W}_n) \le l_n,\ n = 1, \ldots, N. \\ +\infty & \text{otherwise} \end{cases} \tag{2}$$

In its entirety, the problem cannot be solved either analytically or by using **Stochastic Gradient Descent (SGD)**. To enable the decomposition into smaller manageable subproblems, duplicate variables are introduced and the problem is equivalently rewritten as

$$\min_{\mathbf{W}_n, \mathbf{Z}_n} \left\{ f\left(\mathbf{W}_n\right) + \sum_{n=1}^{N} g_n(\mathbf{Z}_n) \right\}, \quad \text{subject to } \mathbf{W}_n = \mathbf{Z}_n,\ n = 1, \ldots, N. \tag{3}$$

Here, the first term is a non-linear smooth loss function and the second term is a non-differentiable "cardinality" penalty term [50]. To solve the problem, constraints are first relaxed by introducing Lagrangian multipliers to decompose the resulting problem into manageable subproblems, which will be coordinated by the multipliers. The constraint violations are also penalized by using quadratic penalties to speed up convergence. The resulting *augmented* Lagrangian function [3, 50] of the preceding optimization problem is thus given by

$$L_\rho\left(\mathbf{W}_n, \mathbf{Z}_n, \Lambda_n\right) = f\left(\mathbf{W}_n\right) + \sum_{n=1}^{N} g_n(\mathbf{Z}_n) + \sum_{n=1}^{N} \mathrm{tr}[\Lambda_n^T(\mathbf{W}_n - \mathbf{Z}_n)] + \sum_{n=1}^{N} \frac{\rho}{2} \|\mathbf{W}_n - \mathbf{Z}_n\|_F^2, \tag{4}$$

where Λ_n is a matrix of Lagrangian multipliers corresponding to constraints $\mathbf{W}_n = \mathbf{Z}_n$ and has the same dimension as $\mathbf{W}_n$. The positive scalar ρ is the penalty coefficient, $\mathrm{tr}(\cdot)$ denotes the trace, and $\|\cdot\|_F^2$ denotes the Frobenius norm.

In the following, we are motivated by decomposability enabled by SLR [6], which overcame all major difficulties of standard Lagrangian relaxation, significantly reducing zigzagging and ensuring convergence. The relaxed problem will be decomposed into two manageable subproblems, andthese subproblems will then be coordinated by Lagrangian multipliers.

Step 1: Solve the "Loss Function" Subproblem for $\mathbf{W}_n$ *by Using SGD.* At iteration k, for given values of multipliers Λ_n^k, the first "loss function" subproblem is to minimize the Lagrangian function while

keeping $\mathbf{Z}_n$ at previously obtained values $\mathbf{Z}_n^{k-1}$ as

$$\min_{\mathbf{W}_n} L_\rho\left(\mathbf{W}_n, \mathbf{Z}_n^{k-1}, \Lambda_n^k\right). \tag{5}$$

Since the regularizer is a differentiable quadratic function, and the loss function is differentiable, the subproblem can be solved by SGD [2]. To ensure that multiplier-updating directions are "proper," the following "surrogate" optimality condition needs to be satisfied following Bragin et al. [6, p. 179, Equation (12)]:

$$L_\rho\left(\mathbf{W}_n^k, \mathbf{Z}_n^{k-1}, \Lambda_n^k\right) < L_\rho\left(\mathbf{W}_n^{k-1}, \mathbf{Z}_n^{k-1}, \Lambda_n^k\right). \tag{6}$$

If (6) is satisfied, multipliers are updated following Bragin et al. [6, p. 179, Equation (15)] as

$$\Lambda'^{k+1}_n := \Lambda_n^k + s'^k(\mathbf{W}_n^k - \mathbf{Z}_n^{k-1}), \tag{7}$$

where stepsizes are updated as in the work of Bragin et al. [6, p. 180, Equation (20)]:

$$s'^k = \alpha^k \frac{s^{k-1}||\mathbf{W}^{k-1} - \mathbf{Z}^{k-1}||}{||\mathbf{W}^k - \mathbf{Z}^{k-1}||}. \tag{8}$$

Step 2: Solve the "Cardinality" Subproblem for $\mathbf{Z}_n$ through Pruning by Using Projections onto Discrete Subspace. The second "cardinality" subproblem is solved with respect to $\mathbf{Z}_n$ while fixing other variables at values $\mathbf{W}_n^k$ as

$$\min_{\mathbf{Z}_n} L_\rho\left(\mathbf{W}_n^k, \mathbf{Z}_n, \Lambda'^{k+1}_n\right). \tag{9}$$

Since $g_n(\cdot)$ is an indicator function, the globally optimal solution of this problem can be explicitly derived as follows [3]:

$$\mathbf{Z}_n^k = \Pi_{\mathbf{S}_n}\left(\mathbf{W}_n^k + \frac{\Lambda'^{k+1}_n}{\rho}\right), \tag{10}$$

where $\Pi_{\mathbf{S}_n}(\cdot)$ denotes the Euclidean projection onto the set $\mathbf{S}_n = \{\mathbf{W}_n \mid \text{card}(\mathbf{W}_n) \leq l_n\}, n = 1, \ldots, N$. To ensure that multiplier-updating directions are "proper," the following "surrogate" optimality condition needs to be satisfied:

$$L_\rho\left(\mathbf{W}_n^k, \mathbf{Z}_n^k, \Lambda'^{k+1}_n\right) < L_\rho\left(\mathbf{W}_n^k, \mathbf{Z}_n^{k-1}, \Lambda'^{k+1}_n\right). \tag{11}$$

Once (11) is satisfied,[1] multipliers are updated as

$$\Lambda_n^{k+1} := \Lambda'^{k+1}_n + s^k(\mathbf{W}_n^k - \mathbf{Z}_n^k), \tag{12}$$

where stepsizes and stepsize-setting parameters [6, p. 188, Equation (67)] are updated as

$$s^k = \alpha^k \frac{s'^k||\mathbf{W}^{k-1} - \mathbf{Z}^{k-1}||}{||\mathbf{W}^k - \mathbf{Z}^k||}; \quad \alpha^k = 1 - \frac{1}{M \times k^{(1-\frac{1}{k^r})}}, \; M > 1, 0 < r < 1. \tag{13}$$

The theoretical results are based on other works [5, 7, 10] and are summarized in the following.

Theorem 1 (Sufficient Condition for Convergence). *Assuming for any integer number κ there exists $k > \kappa$ such that surrogate optimality conditions (6) and (11) are satisfied, then under stepsizing conditions (8) and (13), the Lagrangian multipliers converge to their optimal values Λ_n^* that maximize the following dual function:*

$$q(\Lambda) \equiv \min_{\mathbf{W}_n, \mathbf{Z}_n} L_\rho\left(\mathbf{W}_n, \mathbf{Z}_n, \Lambda_n\right). \tag{14}$$

[1] If condition (11) is not satisfied, subproblems (5) and (9) are solved again by using the latest available values for $\mathbf{W}_n$ and $\mathbf{Z}_n$.

Proof. The proof will be based on that of Bragin et al. [6]. The major difference between the original SLR method [6] and the SLR method of this article is the presence of quadratic terms within the Lagrangian function (4).

It is important to mention that the weight-pruning problem can be equivalently rewritten in a generic form as (15), where $\mathbf{X}$ collectively denotes the decision variables $\{\mathbf{W}_n, \mathbf{Z}_n\}$

$$\min_{\mathbf{X}} \mathbf{F}(\mathbf{X}), \quad s.t. \quad \mathbf{G}(\mathbf{X}) = \mathbf{0}, \tag{15}$$

where

$$\mathbf{F}(\mathbf{X}) \equiv f\left(\mathbf{W}_n\right) + \sum_{n=1}^{N} g_n(\mathbf{Z}_n) + \sum_{n=1}^{N} \frac{\rho}{2} \|\mathbf{W}_n - \mathbf{Z}_n\|_F^2, \quad \mathbf{G}(\mathbf{X}) \equiv \mathbf{W}_n - \mathbf{Z}_n, \quad n = 1, \ldots, N. \tag{16}$$

The feasible set of (15) is equivalent to the original model compression problem. Feasibility requires $\mathbf{W}_n = \mathbf{Z}_n$, which makes the term $\frac{\rho}{2}\|\mathbf{W}_n - \mathbf{Z}_n\|_F^2$ within (16) disappear. Therefore, the Lagrangian function corresponding to (15) is the *augmented* Lagrangian function (4) to the original model compression problem. Furthermore, the surrogate optimality conditions (6) and (11) are the surrogate optimality conditions that correspond to the Lagrangian function $\mathbf{F}(\mathbf{X}) + \Lambda\mathbf{G}(\mathbf{X})$ that corresponds to (15). Therefore, since within the original SLR [6, Prop. 2.7, p. 188] convergence was proved under conditions on stepsizes (8) and (13) and the satisfaction of surrogate optimality conditions, which are assumed to be satisfied here, multipliers converge to their optimal values for the model compression under consideration as well. □

Although the convergence proof in Theorem 1 is valuable for distinguishing SLR from previous decomposition and coordination methods (e.g., ADMM) in terms of convergence, it does not provide insight into the solution quality on its own. This issue is addressed in Theorem 2, where the faster convergence of SLR is rigorously quantified. Since both ADMM and SLR methods are dual methods to maximize the dual function (14), it is common practice to determine upper bounds for the maximization problems. In the context of the problem being examined, an upper bound for the optimal dual value q^* will be established within each method, enabling the evaluation of the quality of dual solutions—specifically, Lagrangian multipliers serve as the decision variables in the dual space.[2]

Theorem 2 (Dual Solution Quality: Best-Case Performance). *Assuming that the "sufficient condition for convergence" stated within Theorem 1 is satisfied, then surrogate Lagrangian relaxation provides a better dual solution quality as compared to ADMM: in particular, there exists an iteration κ so that for all iterations $k > \kappa$, the following condition holds:*

$$\overline{q}_{\kappa}^{SLR} < \overline{q}_{\kappa}^{ADMM}. \tag{17}$$

Proof. There exists an overestimate of the optimal dual value [5, 7], which in terms of our problem under consideration can be expressed as follows:

$$\overline{q}_{k}^{SLR} = \gamma \cdot s^k \cdot \|\mathbf{W}_n^k - \mathbf{Z}_n^k\|_F^2 + L_\rho\left(\mathbf{W}_n^k, \mathbf{Z}_n^k, \Lambda_n^k\right). \tag{18}$$

Here, $\gamma \in [0, 1]$ is a parameter. Following Nedic and Bertsekas [30], γ can be set as $\frac{1}{2}$ —a reciprocal of the number of subproblems; however, for this theoretical analysis, the specific value of γ is of secondary importance. Since within the SLR the stepsizes are approaching 0, then

$$\overline{q}_{k}^{SLR} \to L_\rho\left(\mathbf{W}_n^k, \mathbf{Z}_n^k, \Lambda_n^k\right). \tag{19}$$

[2]Higher quality of primal variables—weights and biases are implied since superior coordination through Lagrangian multipliers significantly improves the quality of primal solutions [7].

Moreover, assuming that a sufficient condition for convergence is satisfied, then the Lagrangian dual value approaches the dual value (as proved, for example, in the work of Bragin et al. [6]), and therefore

$$\overline{q}_k^{SLR} \to q(\lambda^*). \tag{20}$$

Within ADMM, however, since stepsizes/penalty coefficients do not approach zero, then the overestimate of the optimal dual value $q(\lambda^*)$ is bounded away from it:

$$\overline{q}_k^{ADMM} = \gamma \cdot \rho \cdot \|\mathbf{W}_n^k - \mathbf{Z}_n^k\|_F^2 + L_\rho\left(\mathbf{W}_n^k, \mathbf{Z}_n^k, \Lambda_n^k\right). \tag{21}$$

Therefore, since the first term does not approach zero, and the second term does not approach the optimal dual value from above, there exists κ such that (17) holds. □

ALGORITHM 1: Surrogate Lagrangian relaxation.

```
Initialize W_n^0, Z_n^0, Λ_n^0 and s^0
while Stopping criteria are not satisfied do
    1 solve subproblem (5),
    if surrogate optimality condition (6) is satisfied then
        keep W_n^k, Z_n^k, and update multipliers Λ_n^k per (7),
    else
        keep W_n^k, Z_n^k, do not update multipliers Λ_n^k,
    end if
    2 solve subproblem (9),
    if surrogate optimality condition (11) is satisfied then
        keep W_n^k, Z_n^k, and update multipliers Λ_n^k per (12),
    else
        keep W_n^k, Z_n^k, do not update multipliers Λ_n^k,
    end if
end while
```

The algorithm of the proposed SLR method is summarized in Algorithm 1. The SLR method benefits from efficient subproblem solution coordination with guaranteed convergence enabled by stepsizes (8) and (13) approaching zero. Without this requirement, multipliers (12) would not exhibit convergence. The satisfaction of surrogate optimality conditions (6) and (11) ensures that multipliers are updated along "good" directions, promoting convergence. Section 4 will empirically verify that there always exists iteration κ after which the "surrogate" optimality conditions are satisfied ensuring that multipliers approach their optimal values during the entire iterative process.

The SLR method also benefits from the independent and systematic adjustment of two hyperparameters: penalty coefficient ρ and the stepsize s^k. In contrast, other coordination methods are not designed to handle discrete variables and other types of non-convexities. For example, ADMM does not converge when solving non-convex problems [3, p. 73] because stepsizes ρ within the method do not converge to zero. Lowering stepsizes to zero within ADMM would also result in a decreased penalty coefficient, leading to slower convergence.

4 EVALUATION

In this section, we discuss our experimental results for the image classification task using CNN-based models and non-CNN-based models. We also evaluate our method under object detection and image segmentation tasks.

4.1 Experimental Setup

All of our code, including image classification tasks and object detection and segmentation tasks, is implemented with Python 3.6 and PyTorch 1.6.0. For our experiments on the COCO 2014 dataset, we used the pycocotools v2.0 packages. For our experiments on the TuSimple lane detection

Table 1. Comparison of SLR and ADMM on CIFAR-10 and ImageNet

	Baseline (%)	Epoch	ADMM (%)	SLR (%)	Compression Rate
CIFAR-10					
ResNet-18	93.33	40	72.84	*89.93*	8.71×
ResNet-50	93.86	50	78.63	*88.91*	6.57×
VGG-16	93.27	110	69.05	*87.31*	12×
ResNet-56	93.39	30	90.5	*92.3*	6.5×
ResNet-110	93.68	30	89.71	*92.31*	9.7×
ImageNet					
ResNet-18	69.7 / 89.0	40	58.9 / 81.7	*60.9 / 84.4*	6.5×
ResNet-50	76.1 / 92.8	30	64.8 / 85.1	*65.9 / 87.5*	3.89×
MobileNetV2	71.8 / 91.0	50	61.8 / 84.3	*63.2 / 85.5*	1.76×

ImageNet results show Top-1 / Top-5 accuracy. *Epoch* means when SLR or ADMM converges. The accuracy values for all experiments are reported at the corresponding epoch.

benchmark dataset,[3] we used the SpConv v1.2 package. We conducted our experiments on Ubuntu 18.04 using an NVIDIA Quadro RTX 6000 GPU with 24 GB of GPU memory.

We begin by pruning the pre-trained models through SLR training. Afterward, we perform hard-pruning on the model, completing the compression phase. We report the overall compression rate (or the percentage of remaining weights) and prediction accuracy.

4.2 Evaluation of Image Classification Tasks Using CNN-Based Models

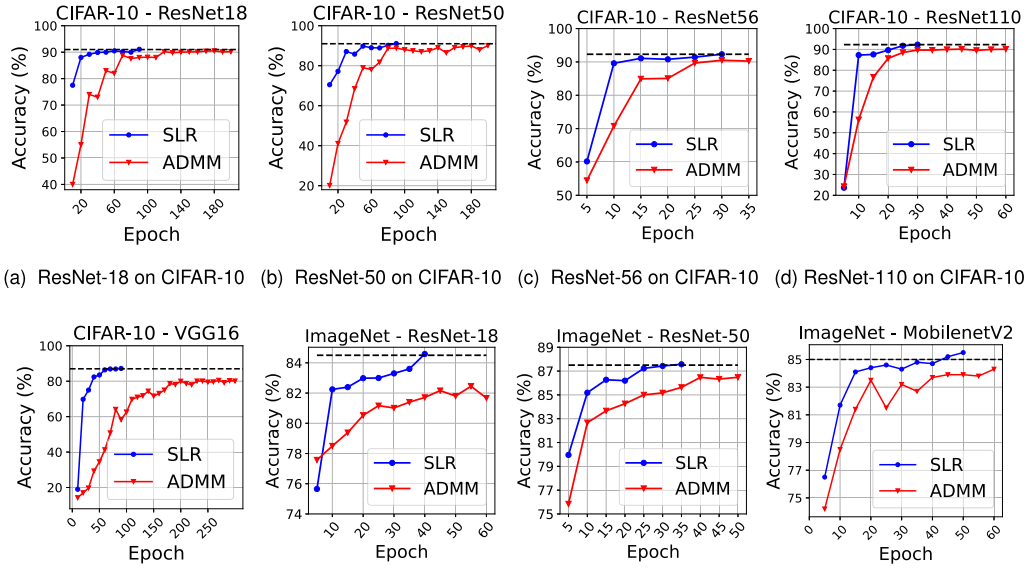
Models and Datasets. We use ResNet-18, ResNet-50, ResNet-56, ResNet-110 [12], and VGG-16 [40] on CIFAR-10. On the ImageNet ILSVRC 2012 benchmark, we use ResNet-18, ResNet-50 [12], and MobileNetV2 [39]. We use the pre-trained ResNet models on ImageNet from Torchvision's "models" subpackage.

Training Settings. In all the experiments, we set $\rho = 0.1$. In CIFAR-10 experiments, we use a learning rate of 0.01, batch size of 128, and the ADAM optimizer. In ImageNet experiments, we use a learning rate of 10^{-4}, batch size of 256, and the SGD optimizer. For a fair comparison of SLR and ADMM methods, we use the same sparsity configurations for both methods in the experiments.

Table 1 shows our comparison of SLR and ADMM on CIFAR-10 and ImageNet benchmarks. For both experiments, SLR parameters are set as $M = 300$, $r = 0.1$, and $s_0 = 10^{-2}$. After the SLR and ADMM training, hardpruning is performed, and the hardpruning accuracy is reported without any additional retraining, given a limited budget of training epochs. The *epoch* listed in the table corresponds to the specific point in the training process where either SLR or ADMM successfully converges, indicating when optimal performance is reached. The accuracy values for all the experiments are reported at the corresponding epoch. According to our results, SLR outperforms the ADMM method in terms of accuracy under the same compression rate. When the compression rate is the same, SLR always obtains higher classification accuracy compared with ADMM under the same number of epochs of training. As compression rates increase, the results reveal a more significant accuracy gap between SLR and ADMM across various network architectures. This demonstrates that our SLR converges faster and quickly recovers the model accuracy at the hardprune stage, which achieves retrain-free given the limited training budget.

Figure 1 shows the hardpruning accuracy for SLR vs. ADMM on CIFAR-10 and ImageNet, corresponding to Table 1. During training, hardpruning accuracy is checked periodically. If the

[3] https://github.com/TuSimple/tusimple-benchmark



(a) ResNet-18 on CIFAR-10 (b) ResNet-50 on CIFAR-10 (c) ResNet-56 on CIFAR-10 (d) ResNet-110 on CIFAR-10

(e) VGG-16 on CIFAR-10 (f) ResNet-18 on ImageNet (g) ResNet-50 on ImageNet (h) MobilenetV2 on ImageNet

Fig. 1. Hardpruning accuracy after SLR and ADMM training on CIFAR-10 and ImageNet benchmarks. Accuracy is reported periodically and training is stopped when desired accuracy is reached.

hardpruning accuracy meets the accuracy criteria, the training is stopped. As seen in Figure 1, SLR converges quickly and reaches the desired accuracy almost 3× faster than ADMM on CIFAR-10. Moreover, in Figure 1(e), ADMM is still below the desired accuracy even after 300 epochs of training on VGG-16, whereas SLR finishes training at 80 epochs. Similarly, as shown in Figure 1(f) and (g), ADMM cannot achieve the desired accuracy after 60 and 50 epochs of training onImageNet, whereas SLR reaches the threshold quicker.

Table 2 shows the comparison of SLR with other recent model compression works on the CIFAR-10 benchmark. We report the percentage of parameters pruned after SLR training and the final accuracy. To ensure fair comparison against the state-of-the-art method, we have made adjustments to the compression rate of our SLR algorithm. Our experimental results consistently demonstrate the superior performance of our SLR algorithm compared to the state-of-the-art algorithms across a range of model architectures and compression rates. It is evident that our method consistently outperforms the existing approach, making it a compelling choice for various application scenarios.

4.3 Evaluation on Image Classification Tasks Using State-of-the-Art Non-CNN-Based Models

In this subsection, we demonstrate our experimental results of applying the SLR weight pruning method on an MLP-based architecture(MLP-Mixer [41]) and on an attention-based network (Swin Transformer [25]).

Models and Datasets. MLP-Mixer does not contain any convolution layer or self-attention block. This architecture relies solely on MLPs. Alternating between channel-mixing MLPs and token (image patch)-mixing MLPs, the MLP-Mixer model can achieve decent accuracy with much fewer computational resources than state-of-the-art methods. Swin Transformer is a hierarchical transformer based architecture that utilizes shifted windows for representation. It has linear

Table 2. SLR Performance Comparison with VGG-16, ResNet-18, ResNet-50, and ResNet-56 on CIFAR-10

Model	Method	Accuracy (%)	Parameters Pruned (%)
VGG-16	**SLR**	91.2	90
	AMC [13]	91.0	
	L0 [27]	80.0	
	SLR	93.1	60
	One-shot pruning [26]	92.4	
	SLR	93.2	50
	Iter. Prun. [11]	92.2	
ResNet-18	**SLR**	91.76	95
	EarlyCroP-U [36]	91.1	
	GRASP [42]	88.4	
	SLR	92.3	85
	Iter. Prun. [11]	75	
	SLR	92.93	75
	Cprune [15]	92.7	
ResNet-50	**SLR**	93.6	60
	AMC [13]	93.5	
ResNet-56	**SLR**	93.8	80
	GSM [8]	94.1	
	Group Sparsity [19]	92.65	
	SLR	93.64	50
	GNN-RL [48]	93.49	
	KSE [21]	93.23	
	EB [46]	92.44	
	RST [1]	92.11	
	SLR	93.66	40
	3D [43]	93.46	
	DHP [20]	93.32	
	HRank [23]	93.17	
	NISP [47]	93.01	
	SLR	93.52	15
	GAL-0.6 [24]	93.38	
	[18]	93.06	

computational complexity with respect to input image size and achieves state-of-the-art performance on both image classification and object detection and semantic segmentation tasks. We conduct experiments on the ImageNet ILSVRC 2012 benchmark. We utilize the pre-trained models from the PyTorch Image Models code base [45].

Training Settings. Similar to experiments with CNN-based models, we set $\rho = 0.1$. We set the learning rate at 0.01 and the batch size at 128, and we use the SGD optimizer with the momentum of 0.9 and a weight-decay of 10^{-4}. To ensure a fair comparison, we used the same number of training epochs and sparsity configurations for both SLR and ADMM methods.

Comparison of SLR and ADMM.. The comparison of SLR and ADMM applied on the MLP-Mixer model on the ImageNet benchmark is presented in Table 3. We compare the two methods using three distinct compression rates. The final hardpruning accuracy is reported after 100 epochs

Table 3. SLR Pruning Results with MLP-Mixer on ImageNet through Different Compression Rates

Baseline Acc (%)		Compression Rate	Hardpruning Acc. (%)				Retraining Acc. (%)			
			ADMM		SLR		ADMM		SLR	
Top@1	Top@5		Top@1	Top@5	Top@1	Top@5	Top@1	Top@5	Top@1	Top@5
76.598	92.228	1.96×	75.332	91.734	75.392	91.752	75.332	91.714	75.392	91.752
		3.16×	71.798	89.64	72.578	90.136	73.362	90.672	73.56	90.750
		8.28×	47.592	71.698	54.834	77.696	70.864	89.38	71.036	89.494

Table 4. SLR Pruning Results with Swin Transformer (Tiny) on ImageNet through Different Compression Rates

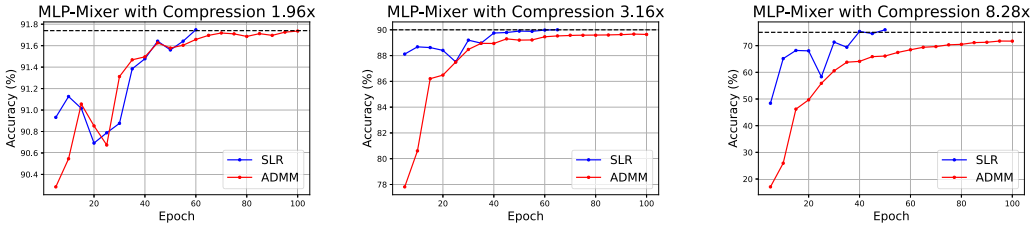
Baseline Acc (%)		Compression Rate	Hardpruning Acc. (%)				Retraining Acc. (%)			
			ADMM		SLR		ADMM		SLR	
Top@1	Top@5		Top@1	Top@5	Top@1	Top@5	Top@1	Top@5	Top@1	Top@5
81.350	95.532	1.95×	78.910	94.288	79.018	94.446	79.146	94.424	79.108	94.456
		3.13×	73.074	91.370	74.322	91.988	75.278	92.55	75.508	92.630
		4.50×	63.892	85.104	67.398	87.494	72.626	90.966	73.046	91.364

of training without further retraining. Retrain accuracy is also reported after 50 epochs. For all SLR experiments, we set the parameters to $M = 300$, $r = 0.1$, and $s_0 = 0.01$. As indicated in Table 3, in terms of hardpruning, when the compression rate is low at 1.96×, the two methods have similar performance. As the compression rate increases to 3.16× and 8.28×, SLR outperforms ADMM, with the accuracy gap widening as the compression rate increases. This demonstrates that SLR outperforms ADMM by achieving higher accuracy during the hardpruning stage, which leads to more efficient use of training resources.

The comparison of SLR and ADMM on the Swin Transformer model on the ImageNet benchmark is shown in Table 4. In all the experiments, we choose the tiny version (Swin-T) because it balances the model size and accuracy very well. For SLR experiments, when the compression rate is 1.96×, we use the parameters $M = 150$, $r = 0.1$, and $s_0 = 0.05$. When the compression rate is 3.16×, we use the parameters $M = 300$, $r = 0.1$, and $s_0 = 0.01$. And when compression rate is 4.5×, we use the parameters $M = 150$, $r = 0.05$, and $s_0 = 0.005$. As demonstrated, when the compression rate is 1.95×, SLR obtains higher accuracy than ADMM at the hardpruning stage. After 50 epochs of retraining, the accuracy of ADMM improves 0.2% and becomes closer to the accuracy of SLR, whereas SLR improves 0.01%. This demonstrates that our SLR can obtain higher accuracy at the hardpruning stage. As the compression rate increases, the accuracy gap between SLR and ADMM also widens. For instance, at a compression rate of 4.5×, the accuracy of SLR is 2.4% higher than that of ADMM at the hardpruning stage.

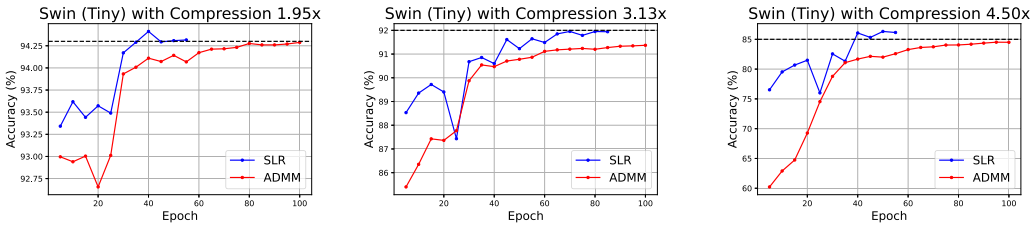
Figure 2 plots periodically checked the hardpruning accuracy of SLR and ADMM with MLP-Mixer on ImageNet. The compression rates correspond to Table 3, following the same procedure described in Section 4.2 that if the hardpruning accuracy meets the accuracy criteria, the training is stopped. As shown, SLR quickly converges and achieves higher accuracy compared with ADMM. When the compression rate is low, such as 1.96×, two methods converge at a similar rate. However, as the compression rate increases, SLR not only converges faster than ADMM but also reaches much higher accuracy. ADMM even cannot achieve the desired accuracy after 60 epochs of training when the compression rate is 8.28×. In Figure 2(c), we can see that SLR quickly converges to the threshold at around 50 epochs and results in an 8.6% accuracy gap between ADMM when the training of SLR is stopped, and ADMM's accuracy continues to improve, albeit slowly, until epoch 100.

Figure 3 plots periodically checked hardpruning accuracy of SLR and ADMM with Swin Transformer on ImageNet, with the compression rates corresponding to those reported in Table 4. It



(a) Compression rate = 1.96× (b) Compression rate = 3.16× (c) Compression rate = 8.28×

Fig. 2. Top-5 hardprune accuracy of MLP-Mixer on ImageNet after SLR and ADMM pruning. Accuracy is reported periodically and training stops when the method reaches the accuracy threshold.



(a) Compression rate = 1.95× (b) Compression rate = 3.13× (c) Compression rate = 4.5×

Fig. 3. Top-5 hardprune accuracy of Swin Transformer (Tiny) on ImageNet benchmarks after applying SLR and ADMM weight pruning. Accuracy is reported periodically and training stops when the method reaches the accuracy threshold.

follows the same procedure described in Section 4.2 that the training stops if the hardpruning accuracy meets the accuracy criteria. Under the three compression rates, SLR consistently converges to the desired accuracy in half the number of training epochs compared to ADMM. Even when the compression rate is 3.16×, SLR results in an approximately 1% accuracy gap with ADMM when it stops training. This further demonstrates that our SLR can quickly recover the model accuracy at the hardpruning stage itself.

4.4 Evaluation on Object Detection and Segmentation Tasks

In this subsection, we evaluate our SLR-based weight pruning method on three object detection and segmentation benchmarks.

Models and Datasets. In the first experiment, we test YOLOv3 and YOLOv3-tiny models [37] on the COCO 2014 benchmark. We followed the publicly available Ultralytics repository[4] for YOLOv3 and its pre-trained models. The second experiment focuses on lane detection. We use the pre-trained model from Ultra-Fast-Lane-Detection [35] on the TuSimple lane detection benchmark dataset. The third experiment involves 3D point cloud object detection experiments. We use the PointPillars [17] pre-trained model on the KITTI 2017 dataset following the OpenPCDet repository.[5] In this experiment, we use LIDAR point cloud data as input. Each point cloud data point is stored as a large collection of 3D elevation points, and each point is represented as a $1 * 4$ vector containing x, y, z (3D coordinates) and intensity [51].

Training Settings. In all experiments, we use $\rho = 0.1$. We set SLR parameters as $M = 300$, $r = 0.1$, and $s_0 = 10^{-2}$. We follow the same training settings provided by the repositories we use.

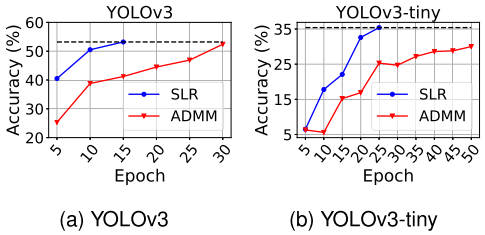
[4] https://github.com/ultralytics/yolov3
[5] https://github.com/open-mmlab/OpenPCDet

Table 5. ADMM and SLR Pruning Results with Different Structures of YOLOv3 on the COCO Dataset

Architecture	Epoch	Method	Hardprune mAP	Comp. Rate
YOLOv3-tiny *(mAP = 37.1)*	15	ADMM	35.2	1.19×
		SLR	*36.1*	
	20	ADMM	32.2	2×
		SLR	*36.0*	
	25	ADMM	25.3	3.33×
		SLR	*35.4*	
YOLOv3-SPP *(mAP = 64.4)*	15	ADMM	41.2	2×
		SLR	*53.2*	

Table 6. SLR Pruning Results with ResNet-18 on the TuSimple Benchmark through Different Compression Rates

Comp. Rate	Hardprune Acc. (%)		Retrain Acc. (%)	
	ADMM	SLR	ADMM	SLR
1.82×	92.49	94.64	94.28	94.63
2.54×	92.25	94.56	94.04	94.93
4.21×	90.97	94.66	94.18	94.68
12.10×	88.41	94.51	94.45	94.7
16.85×	78.75	94.55	94.23	94.65
22.80×	67.79	94.62	94.08	94.55
35.25×	57.05	93.93	93.63	94.34
77.67×	46.09	89.72	88.33	90.18



(a) YOLOv3 (b) YOLOv3-tiny

Fig. 4. Hardpruning accuracy of YOLOv3 and YOLOv3-tiny. Accuracy is reported every 5 epochs and training is stopped when methods reach the accuracy threshold.

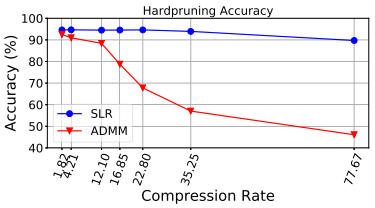


Fig. 5. Hardpruning accuracy on the TuSimple benchmark with ADMM vs. SLR training for several compression rates. SLR has a greater advantage over ADMM as the compression rate increases.

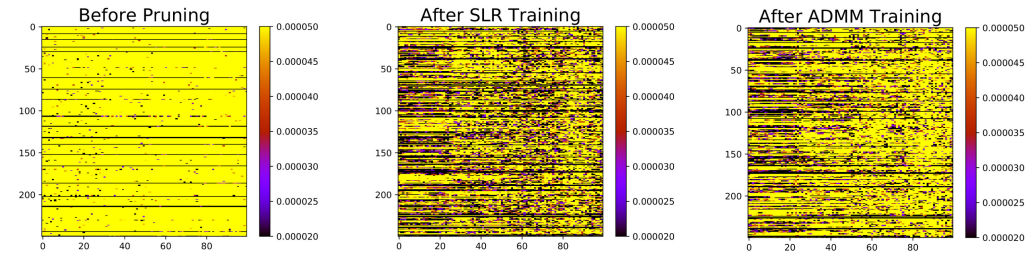
Finally, weuse the same number of training epochs and sparsity configurations for ADMM and SLR.

Testing Settings. On the YOLOv3 models, we calculate the COCO mAP with IoU = 0.50 with an image size of 640 for testing. In lane detection experiments, the evaluation metric is "accuracy," which is calculated as $\frac{\sum_{\text{clip}} C_{\text{clip}}}{\sum_{\text{clip}} S_{\text{clip}}}$, where C_{clip} is the number of lane points predicted correctly and S_{clip} is the total number of ground truth in each clip.

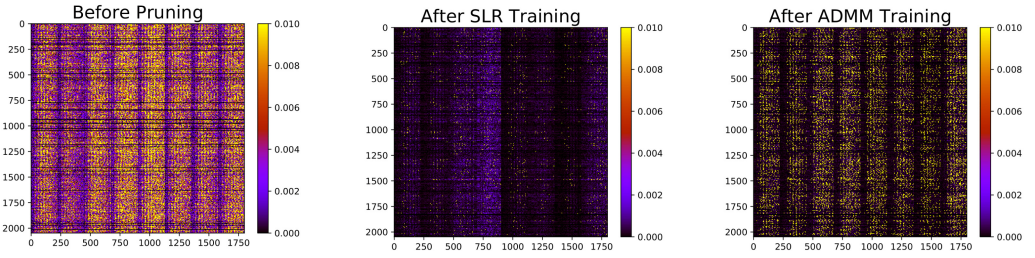
The KITTI dataset is stratified into easy, moderate, and hard difficulty levels. Here, *Easy level* means that the minimum height of the bounding box is 40 pixels, all objects in the images are fully visible, and the percentage of truncation of objects is less than 15%; *Moderate level* means that the minimum bounding box height is 25 pixels, objects in the images are partly visible, and the percentage of truncation of objects is less than 30%; and *Hard level* means that the minimum bounding box height is 25 pixels but some objects in the image are difficult to see, and the maximum percentage of truncation is 50%. mAP is calculated under each difficulty strata with $IoU = 0.5$.

Comparison of SLR and ADMM. Our comparison of SLR and ADMM methods on the COCO dataset is shown in Table 5. We have compared the two methods under three different compression rates for YOLOv3-tiny and tested the YOLOv3-SPP pre-trained model with a compression rate of 1.98×. We can see that the model pruned with the SLR method has higher accuracy after hardpruning in all cases. At a glance at YOLOv3-tiny results, we can observe that the advantage of SLR is higher with an increased compression rate.

At a compression rate of 3.33× on YOLOv3-tiny, given a limit of 25 epochs, we can observe that the gap between ADMM and SLR is much higher, which is due to the faster convergence of SLR as shown in Figure 4(b). Similarly, Figure 4(a) shows the mAP progress of YOLOv3 during SLR and

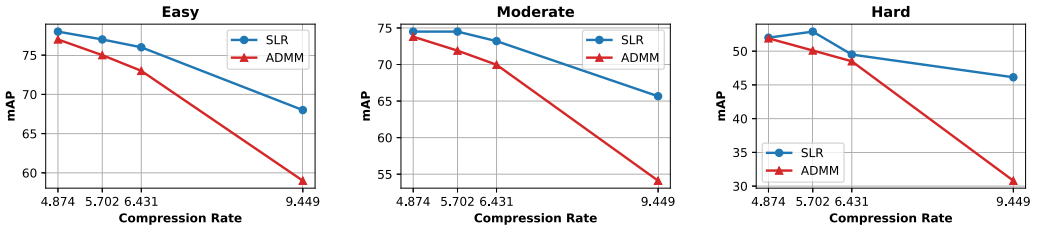


(a) Weights before after pruning with SLR (middle) and ADMM (right) under the same compression rate (77.6×)



(b) Weights before and after pruning with SLR (middle) and ADMM (right) under the same accuracy (89.0%)

Fig. 6. Heatmap of ResNet-18 weights on the TuSimple benchmark before and after pruning with SLR and ADMM. Weights are more zeroed out with SLR compared to ADMM.



(a) Pruning on Easy task level (b) Pruning on Moderate task level (c) Pruning on Hard task level

Fig. 7. mAP of the PointPillars model after hardpruning using different compression rates under different difficulty levels.

ADMM training for 50 epochs, pruned with a 2× compression rate. SLR reaches the mAP threshold only at epoch 15.

Table 6 reports our result for the lane detection task on the TuSimple lane detection benchmark after 40 epochs of training and 5 epochs of masked retraining. We conducted experiments under eight different compression rates. Figure 5 illustrates the accuracy gap between ADMM and SLR methods after hardpruning as the compression rate increases.

From Figure 5, our observation is that for a small compression rate such as 1.82×, SLR has little advantage over ADMM in terms of hardpruning accuracy. However, as the compression rate increases, SLR starts to perform better than ADMM. For example, SLR survives 77.67× compression with slight accuracy degradation and results in 89.72% accuracy after hardpruning, whereas ADMM accuracy drops to 46.09%. This demonstrates that our SLR-based training method has a greater advantage over ADMM, especially at higher compression rates, as it achieves compression with less accuracy loss and reduces the time required to retrain after hardpruning.

Table 7. ADMM and SLR Results of the PointPillars Model on KITTI in Different Task Difficulty Levels and Compression Rates

Level	Orig. mAP	Compression	ADMM		SLR	
			After Hardprune	After Retrain	After Hardprune	After Retrain
Easy	80.7	4.874×	77.0	81.5	77.8	82.2
		5.702×	74.7	74.7	77.3	79.3
		6.431×	72.9	77.5	76.6	75.3
		9.449×	58.6	70.9	68.0	79.9
Moderate	78.5	4.874×	73.8	77.1	74.5	78.4
		5.702×	71.9	71.0	74.5	75.3
		6.431×	69.9	73.2	73.2	72.7
		9.449×	54.1	66.8	65.6	75.8
Hard	60.7	4.874×	51.9	51.2	52.0	56.4
		5.702×	50.1	50.2	52.9	51.3
		6.431×	48.5	47.8	49.5	50.7
		9.449×	30.8	35.6	46.1	51.4

Finally, in Figure 6, we show the difference between the weights of one layer before and after pruning with SLR and ADMM. In Figure 6(a), we show the initial (non-pruned) weights and then show the sparsity of weights under the same compression rate (77×) with SLR and ADMM. Initially, the layer has low sparsity. After training with SLR and ADMM, we can see an increased number of zeroed weights. SLR moves toward the desired sparsity level faster than ADMM. In Figure 6(b), we compare the sparsity of weights under the same accuracy (89.0%). It can be observed that SLR significantly reduced the number of non-zero weights, and ADMM has more non-zero weights remaining compared with SLR.

Last, Table 7 shows the 3D point cloud object detection model compression results on the KITTI benchmark. The SLR and ADMM results are reported after 40 epochs of training and 3 epochs of masked retraining. We have two observations. First, SLR has much higher accuracy than ADMM after hardpruning with the compression rate increased. For example, as shown in Figure 7, when tested under a 9.44× compression rate, under "hard" strata, SLR has a mAP that is more than 15% higher than ADMM, as illustrated in Figure 7(c). In "easy" and "moderate" difficulty strata, as shown in Figure 7(a) and (b), the hardpruning mAP of SLR is still more than 10% higher than ADMM under a 9.44× compression rate. Second, we can observe that since SLR has higher mAP after the hardpruning stage, it also reaches significantly higher mAP after retraining. There is almost 16% mAP difference between SLR and ADMM in the "hard" difficulty level under a 9.44× compression rate. This shows that when the retraining budget is limited, our SLR method can quickly recover the model accuracy.

4.5 Ablation Studies

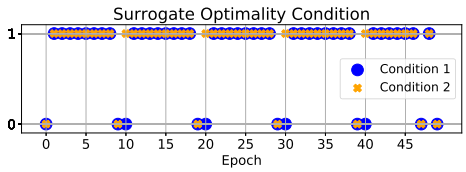
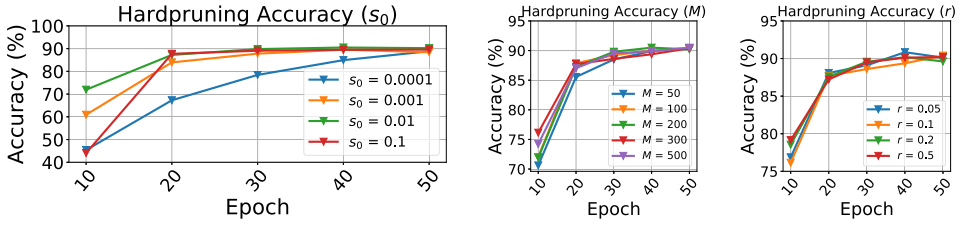


Fig. 8. Surrogate optimality condition satisfaction graph during the SLR training of ResNet-18 on CIFAR-10 for 50 epochs (1: satisfied, 0: not satisfied). Conditions are satisfied periodically.

We conduct several experiments to observe SLR behavior with respect to SLR parameters ρ, s_0, r, and M on the ResNet-18 model (93.33% accuracy) on CIFAR-10. We prune the model through SLR training for 50 epochs with a compression rate of 8.71× and observe the hardpruning accuracy every 10 epochs. Figure 9 shows the accuracy of the model through SLR training based on the different values of s_0, M, and r. Based on the hardpruning accuracy throughout training, it can be seen that even though the parameters do not have a great im-



(a) s_0 parameter (b) M parameter (c) r parameter

Fig. 9. Hardpruning accuracy of ResNet-18 on CIFAR-10 during SLR training with respect to different values of s_0, M, and r.

pact on the end result, the choice of s_0 can impact the convergence of the model. From Figure 9(a), we can state that $s_0 = 10^{-2}$ provides higher starting accuracy and converges quickly. Figure 9(b) and (c) demonstrate the impact of M and r on the hardpruning accuracy, respectively.

Figure 8 demonstrates that there exists iteration κ (as required in the theorem) so that the surrogate optimality condition, the high-level convergence criterion of the SLR method, is satisfied during training with $s_0 = 10^{-2}$, $\rho = 0.1$, thereby signifying that "good" multiplier-updating directions are always found. For example, after the conditions are violated at epoch 9, there exits $\kappa = 10$ so that at iteration 11, after $\kappa = 10$, the surrogate conditions are satisfied again.

5 DISCUSSION

ADMM and SLR are both optimization techniques. Although possessing certain similarities, each is marked by unique attributes and workflows. Primarily, they aim to decompose complex optimization problems into a series of simpler subproblems, which can be solved more efficiently. These subproblems are then coordinated in a manner dictated by specific requirements and the choice of stepsizes.

ADMM, although widely used, strictly speaking, imposes a significant computational burden due to its stringent requirements for subproblem optimality. These requirements become especially challenging due to the non-convex nature of the subproblems. Even though the preceding requirement can be implicitly relaxed (e.g., optimization can stop after one epoch in our problem context), the method does not explicitly provide criteria for the quality of the resulting multiplier-updating directions. However, SLR presents a more approachable option by necessitating the fulfillment of the surrogate optimality condition, which is easier to satisfy (as compared to obtaining the exact optimum at each iteration) and has been proven to be instrumental to guarantee convergence [6], because of better multiplier-updating directions.

The selection of stepsizes also defines a key difference between ADMM and SLR. Due to the discrete nature of the dual problems, stepsizes need to asymptotically approach zero—a concept echoed in numerous historical studies (e.g., [34]) as well as recent ones (e.g., [4]). Although ADMM does not follow this rule, SLR stepsizes do approach zero, which significantly influences the efficacy of each method.

In our research, we provide empirical and theoretical evidence indicating the superiority of SLR over ADMM, primarily presented in Theorem 2. To quantitatively measure the superiority of SLR solutions, we introduce a novel quality metric.

In sum, although both ADMM and SLR methods hold distinct merits, our recent research [4] provides a critical conceptual juxtaposition. Our findings indicate that ADMM may not always converge due to the non-differentiability of the underlying dual function. However, our proposed quality measure providing a theoretical advantage of SLR over ADMM is specific to this work.

6 CONCLUSION

In this article, we addressed the DNN weight-pruning problem as a non-convex optimization problem by adopting the cardinality function to induce weight sparsity. The SLR method decomposes the relaxed weight-pruning problem into subproblems, which are then efficiently coordinated by updating Lagrangian multipliers, resulting in fast convergence. We carried out weight-pruning experiments on image classification and object detection and segmentation tasks on various datasets to compare our SLR method against ADMM and other state of the art. We observed that our SLR method offers a significant advantage under high compression rates and achieves higher accuracy during weight pruning. Additionally, SLR reduces the accuracy loss caused by the hardpruning and shortens the retraining process. Given its effective optimization as well as coordination capabilities and clear advantages demonstrated through various examples, the SLR method holds strong potential for broader DNN-training applications.

REFERENCES

[1] Yue Bai, Huan Wang, Zhiqiang Tao, Kunpeng Li, and Yun Fu. 2022. Dual lottery ticket hypothesis. In *Proceedings of the International Conference on Learning Representations*.

[2] Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT '10)*. 177–186.

[3] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning* 3, 1 (2011), 34–37.

[4] Mikhail A. Bragin. 2023. Survey on Lagrangian relaxation for MILP: Importance, challenges, historical review, recent advancements, and opportunities. *Annals of Operations Research*. Published online July 11, 2023.

[5] Mikhail A. Bragin, Peter B. Luh, Bing Yan, Tongxin Zheng, Dane A. Schiro, Feng Zhao, and Jinye Zhao. 2022. Novel quality measure and efficient resolution of convex hull pricing for unit commitment. *arXiv 2304.07990* (2022).

[6] Mikhail A. Bragin, Peter B. Luh, Joseph H. Yan, Nanpeng Yu, and Gary A. Stern. 2015. Convergence of the surrogate Lagrangian relaxation method. *Journal of Optimization Theory and Applications* 164 (2015), 173–201.

[7] Mikhail A. Bragin and Emily L. Tucker. 2022. Surrogate "level-based" Lagrangian relaxation for mixed-integer linear programming. *Scientific Reports* 12, 1 (2022), 22417.

[8] Xiaohan Ding, Guiguang Ding, Xiangxin Zhou, Yuchen Guo, Jungong Han, and Ji Liu. 2019. Global sparse momentum SGD for pruning very deep neural networks. *Advances in Neural Information Processing Systems* 32 (2019), 1–13.

[9] Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635* (2018).

[10] Deniz Gurevin, Mikhail A. Bragin, Caiwen Ding, Shanglin Zhou, Lynn Pepin, Bingbing Li, and Fei Miao. 2021. Enabling retrain-free deep neural network pruning using surrogate Lagrangian relaxation. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

[11] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS '15)*. 1135–1143.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[13] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. 2018. AMC: AutoML for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV '18)*. 784–800.

[14] Yihui He, Xiangyu Zhang, and Jian Sun. 2017. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*. 1389–1397.

[15] Taeho Kim, Yongin Kwon, Jemin Lee, Taeho Kim, and Sangtae Ha. 2022. CPrune: Compiler-informed model pruning for efficient target-aware DNN execution. In *Proceedings of the European Conference on Computer Vision*. 651–667.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems* 25 (2012), 1–9.

[17] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. 2019. PointPillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12697–12705.

[18] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2016. Pruning filters for efficient ConvNets. *arXiv preprint arXiv:1608.08710* (2016).

[19] Yawei Li, Shuhang Gu, Christoph Mayer, Luc Van Gool, and Radu Timofte. 2020. Group sparsity: The hinge between filter pruning and decomposition for network compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8018–8027.

[20] Yawei Li, Shuhang Gu, Kai Zhang, Luc Van Gool, and Radu Timofte. 2020. DHP: Differentiable meta pruning via hypernetworks. In *Computer Vision—ECCV 2020*. Lecture Notes in Computer Science, Vol. 12353. Springer, 608–624.

[21] Yuchao Li, Shaohui Lin, Baochang Zhang, Jianzhuang Liu, David Doermann, Yongjian Wu, Feiyue Huang, and Rongrong Ji. 2019. Exploiting kernel sparsity and entropy for interpretable CNN compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2800–2809.

[22] Zhengang Li, Yifan Gong, Xiaolong Ma, Sijia Liu, Mengshu Sun, Zheng Zhan, Zhenglun Kong, Geng Yuan, and Yanzhi Wang. 2020. SS-Auto: A single-shot, automatic structured weight pruning framework of DNNs with ultra-high efficiency. *arXiv preprint arXiv:2001.08839* (2020).

[23] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. 2020. HRank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1529–1538.

[24] Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David Doermann. 2019. Towards optimal structured CNN pruning via generative adversarial learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2790–2799.

[25] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin Transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030* (2021).

[26] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. 2018. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270* (2018).

[27] Christos Louizos, Max Welling, and Diederik P. Kingma. 2017. Learning sparse neural networks through L_0 regularization. *arXiv preprint arXiv:1712.01312* (2017).

[28] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. 2017. ThiNet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE International Conference on Computer Vision*. 5058–5066.

[29] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. 2017. Variational dropout sparsifies deep neural networks. In *Proceedings of the International Conference on Machine Learning*. 2498–2507.

[30] Angelia Nedic and Dimitri P. Bertsekas. 2001. Incremental subgradient methods for nondifferentiable optimization. *SIAM Journal on Optimization* 12, 1 (2001), 109–138.

[31] Wei Niu, Xiaolong Ma, Sheng Lin, Shihao Wang, Xuehai Qian, Xue Lin, Yanzhi Wang, and Bin Ren. 2020. PatDNN: Achieving real-time DNN execution on mobile devices with pattern-based weight pruning. In *Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems*. 907–922.

[32] Eunhyeok Park, Junwhan Ahn, and Sungjoo Yoo. 2017. Weighted-entropy-based quantization for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5456–5464.

[33] Hongwu Peng, Shanglin Zhou, Scott Weitze, Jiaxin Li, Sahidul Islam, Tong Geng, Ang Li, Wei Zhang, Minghu Song, Mimi Xie, Hang Liu, and Caiwen Ding. 2021. Binary complex neural network acceleration on FPGA. In *Proceedings of the 2021 IEEE 32nd International Conference on Application-Specific Systems, Architectures, and Processors (ASAP '21)*. IEEE, Los Alamitos, CA, 85–92.

[34] T. B. Poljak. 1967. A general method for solving extremal problems. *Soviet Mathematics–Doklady* 8 (1967), 593–597.

[35] Z. Qin, H. Wang, and X. Li. 2020. Ultra fast structure-aware deep lane detection. *arXiv preprint arXiv:2004.11757* (2020).

[36] John Rachwan, Daniel Zügner, Bertrand Charpentier, Simon Geisler, Morgane Ayle, and Stephan Günnemann. 2022. Winning the lottery ahead of time: Efficient early network pruning. In *Proceedings of the International Conference on Machine Learning*. 18293–18309.

[37] Joseph Redmon and Ali Farhadi. 2018. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).

[38] Ao Ren, Tianyun Zhang, Shaokai Ye, Jiayu Li, Wenyao Xu, Xuehai Qian, Xue Lin, and Yanzhi Wang. 2019. ADMM-NN: An algorithm-hardware co-design framework of DNNs using alternating direction methods of multipliers. In *Proceedings of International Conference on Architectural Support for Programming Languages and Operating Systems*. 925–938.

[39] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4510–4520.

[40] K. Simonyan and A. Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[41] Ilya O. Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. 2021. MLP-Mixer: An all-MLP architecture for vision. *Advances in Neural Information Processing Systems* 34 (2021), 24261–24272.

[42] Chaoqi Wang, Guodong Zhang, and Roger Grosse. 2020. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376* (2020).
[43] Wenxiao Wang, Minghao Chen, Shuai Zhao, Long Chen, Jinming Hu, Haifeng Liu, Deng Cai, Xiaofei He, and Wei Liu. 2021. Accelerate CNNs from three dimensions: A comprehensive pruning framework. In *Proceedings of the International Conference on Machine Learning*. 10717–10726.
[44] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning structured sparsity in deep neural networks. *Advances in Neural Information Processing Systems* 29 (2016), 1–9.
[45] Ross Wightman. 2019. PyTorch Image Models. Retrieved September 23, 2023 from https://github.com/rwightman/pytorch-image-models
[46] Haoran You, Chaojian Li, Pengfei Xu, Yonggan Fu, Yue Wang, Xiaohan Chen, Richard G. Baraniuk, Zhangyang Wang, and Yingyan Lin. 2020. Drawing early-bird tickets: Towards more efficient training of deep networks. In *Proceedings of the International Conference on Learning Representations*.
[47] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I. Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S. Davis. 2018. NISP: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 9194–9203.
[48] Sixing Yu, Arya Mazaheri, and Ali Jannesari. 2022. Topology-aware network pruning using multi-stage graph embedding and reinforcement learning. In *Proceedings of the International Conference on Machine Learning*. 25656–25667.
[49] Tianyun Zhang, Xiaolong Ma, Zheng Zhan, Shanglin Zhou, Caiwen Ding, Makan Fardad, and Yanzhi Wang. 2021. A unified DNN weight pruning framework using reweighted optimization methods. In *Proceedings of the 2021 58th ACM/IEEE Design Automation Conference (DAC '21)*. IEEE, Los Alamitos, CA, 493–498.
[50] Tianyun Zhang, Shaokai Ye, Kaiqi Zhang, Jian Tang, Wujie Wen, Makan Fardad, and Yanzhi Wang. 2018. A systematic dnn weight pruning framework using alternating direction method of multipliers. In *Proceedings of the European Conference on Computer Vision (ECCV '18)*. 184–199.
[51] Shanglin Zhou, Mimi Xie, Yufang Jin, Fei Miao, and Caiwen Ding. 2021. An end-to-end multi-task object detection using embedded GPU in autonomous driving. In *Proceedings of the 2021 22nd International Symposium on Quality Electronic Design (ISQED '21)*. IEEE, Los Alamitos, CA, 122–128.
[52] Shanglin Zhou, Xiaowei Xu, Jun Bai, and Mikhail Bragin. 2022. Combining multi-view ensemble and surrogate Lagrangian relaxation for real-time 3D biomedical image segmentation on the edge. *Neurocomputing* 512 (2022), 466–481.

Received 29 April 2023; revised 23 July 2023; accepted 26 August 2023