Channel Pruning in Quantization-aware Training: an Adaptive Projection-gradient Descent-shrinkage-splitting Method

Zhijian Li  
Department of Mathematics  
University of California, Irvine  
Irvine, USA  
zhijjl2@uci.edu

Jack Xin  
Department of Mathematics  
University of California, Irvine  
Irvine, USA  
jack.xin@uci.edu

Abstract—We propose an adaptive projection-gradient descent-shrinkage-splitting method (APGDSSM) to integrate penalty-based channel pruning into quantization-aware training (QAT). APGDSSM concurrently searches weights in both the quantized subspace and the sparse subspace. APGDSSM uses shrinkage operator and a splitting technique to create sparse weights, as well as the Group Lasso penalty to push the weight sparsity into channel sparsity. In addition, we propose a novel complementary transformed $l_{1}$ penalty to stabilize the training for extreme compression.

Index Terms—convolutional neural network, quantization, channel pruning, LASSO.

I. INTRODUCTION

Convolutional neural networks (CNNs) have been widely used for computer vision tasks such as image classification and segmentation. To increase efficiency and reduce memory costs in mobile and IoT applications, network compression is necessary. Quantization and channel pruning are two commonly adopted methods. QAT searches the optimal weight subspace and the sparse subspace. APGDSSM uses shrinkage operator and a splitting technique to create sparse weights, as well as the Group Lasso penalty to push the weight sparsity into channel sparsity. In addition, we propose a novel complementary transformed $l_{1}$ penalty to stabilize the training for extreme compression.

II. RELATED WORK

For a loss function $l$, the Lasso regularized problem is

$$
\mathcal{L}(w) = l(w) + \lambda ||w||_1. \tag{2}
$$

It is well-known that Lasso regularization does parameter selection for the model, and several approaches exist for solving problem (2). In [1], an iterative algorithm of proximal operator (FISTA) solves (2), where the proximal operator for a penalty function $g$ is defined as $\text{Prox}_{\gamma}(w) = \text{argmin}_{w} g(w) + \frac{1}{2}||w - x||^{2}$. The algorithm is:

$$
w^{t+1} = \text{Prox}_{\lambda}(w^{t} - \gamma \nabla f(w^{t}))
$$

where

$$
\text{Prox}_{\lambda}(x) = \text{sgn}(x) \cdot \max(|x| - \lambda, 0).
$$

An alternative method to solve (2) is the Alternating Direction Method of Multipliers (ADMM), through an augmented Lagrangian (Boyd et al. [2]):

$$
\mathcal{L}(w, u, z) = f(w) + \lambda ||u||_1 + \langle z, w - u \rangle + \frac{\beta}{2} ||w - u||^2 \tag{3}
$$

and studying the sparsity of quantized models are performed in [9].

The main contribution of our work here is to propose an integrated objective to do channel pruning and weight quantization in one shot. This is achieved by minimizing a new objective function with group sparse penalty over $Q$ through an adaptive splitting, projection, gradient descent and proximal operations (APGDSSM algorithm). The adaptive step is to avoid weights in a layer all becoming very small, or fix potential model collapse when trained by the integrated steps of the algorithm. Besides adapting training schedule, we also found a new penalty, the so called complementary transformed-$l_{1}$ ($\text{CT}_{l_{1}}$), to steer weights away from the trivial state in each layer. Using $\text{CT}_{l_{1}}$ gives more room to trade-off accuracy for efficiency than adapting training schedule. Experimental results on CIFAR-10, CIFAR-100, and Imagenet support our proposed methodology and framework.
ADMM is adapted to neural network training in [12], [15]. The convergence theorems of ISTA and ADMM require both the loss function and penalty function to be convex, which does not apply to deep neural networks. The relaxed splitting variable method (RSVM, [6]) sparsifies non-convex neural networks by minimizing a simplified augmented Lagrangian:

$$l_{\lambda/\beta}(w, u) = f(w) + \lambda ||u||_1 + \frac{\beta}{2}||u - w||^2.$$  

RVSM updates weights as

$$w \leftarrow w - \gamma \nabla f(w) - \gamma \beta (w - u), \quad u \leftarrow \text{Prox}_{\lambda/\beta}(w)$$  \hspace{1cm} (4)

which extends to non-differential penalties (e.g. $l_0$) with the corresponding proximal operator. The RVSM does not require convex or differentiable penalty function for convergence [6], and it applies to adversarially trained networks [5]. Though models trained by RVSM usually have unstructured sparsity with limited channel sparsity, RVSM extends readily to a group-wise variable splitting method (RGSM, [14]) based on Group Lasso (GL) penalty:

$$||w||_{GL} = \sum_{l=1}^{L} \sum_{i \in I_l} ||w_{l,i}||_2$$

to increase channel sparsity, where $I_l$ is the collection of channels in the $l$-th layer. GL penalty with its proximal operator in closed form is applied channel-wise in network training to realize sparse channels [5], [14]. In [11], RGSM and QAT are combined in a multi-stage process to achieve both channel pruning and binary weights.

### III. METHODOLOGY AND APGSSM ALGORITHM

To train quantized neural networks with sparse channels, we propose an algorithm to concurrently search the optimal weights in the quantized subspace and the sparse subspace, as shown in Algorithm 1. The objective is

$$\min_{u \in \mathcal{Q}} L(u) := l(u) + \lambda_2 ||u||_{GL} + \lambda_1 ||u||_1$$  \hspace{1cm} (5)

The procedure of training is shown in Algorithm 1. We note that the Lasso regularization term in equation (5) is imposed implicitly; the $l_1$ penalty does not contribute to the gradient. Instead, we use the shrinkage operator to minimize it. For parameters, we use symbols against the epoch number $t$, e.g. $\lambda_1^t$, to indicate that there is an adaptive scheme for the values.

This algorithm concurrently searches both the quantized subspace and the subspace of sparse weight (with small $l_1$ norm). We can either use only shrinkage operator (APGDSM) or use it together with the splitting (APGDSSM). The splitting term updates the gradient descent of $\frac{\beta}{2}||\omega^t - \omega^t||^2$, which makes the float weight $\omega^t$ close to the quantized weight $\omega^t$. Since $\omega^t$ is much more sparse than $\omega^t$, the splitting step renders $\omega^t$ with more small elements, which strengthens the performance of the following shrinkage operator. However, pushing $\omega^t$ close to $\omega^t$ can jeopardize the performance, as it is not the descending direction guided by gradient.

### Algorithm 1 APGDSM and APGDSSM

**Input:** Float weights $u^0$, Hyperparameters $\lambda_1, \lambda_2, \beta$.  
**Output:** Quantized weights $u$.

for $t = 1, \ldots, 200$ do

$$u^t = \text{Proj}_Q(u^t)$$

$$f(u^t) = l(u^t) + \lambda_2^t ||u^t||_{GL}$$

$$w^t = w^{t-1} - \alpha \nabla f(u^t)$$

if Splitting then: \hspace{1cm} $\triangledown$ Split if APGDSSM

$$w^t = w^t - \gamma^t \beta (w^t - u^t)$$

end if

$$w^t = \text{Proj}_Q(w^t)$$

end for

$$u = \text{Proj}_Q(u^{200})$$

### IV. IMPLEMENTATION AND EXPERIMENTS

We use the standard adaptive scheme for the learning rate $\gamma^t$. The initial learning rate is 0.1, and we multiply the learning rate by a factor of 0.1 at epochs 80, 120, and 160. During the training, we need to change the scale of the regularization parameters to fit the current learning rate. For both $\lambda_1$, $\lambda_2$, and $\beta$, we empirically design a scheme to adapt the values of parameters. The reason we have a different adaptive scheme from the learning rate is that the training has a high probability to collapse if the parameters are re-scaled too late. As in Algorithm 1, all GL regularization, shrinkage operator, and splitting terms drive the weights to be sparse, which can lead the neural network to reach 100% channel sparsity at some point. When it happens, the training collapses as the cross-entropy loss becomes infinity.

### V. RESULTS

We validate Algorithm 1 in CIFAR10 and CIFAR100 with ResNet ( [8] ). The results are shown in Table II. As the table shows, the GL penalty and the shrinkage operator can significantly improve the weight sparsity and the channel sparsity with minor reduction on accuracy. The splitting step before the shrinkage operator can greatly improve the sparsity. Of course, the model performance would be somewhat affected.

Meanwhile, we numerically verify the convergence of the sparsity in Figure 2. Although the weight sparsity will decrease every time the values of parameters updated, the channel sparsity has a nice convergence along training. The channel-wise GL penalty is the key to push the weight sparsity created by shrinkage and potential splitting into channel sparsity. In Figure 3, we show the comparison of a float ResNet56 and a 4-
the transformed $l_1$ (CTL$_1$) regularization in robust compressed sensing [17]. We define

$$||x||_{CTL_1,a} := 1 - \rho_a(x) = 1 - \frac{|x|}{a + |x|}$$

We remark that $||\cdot||_{CTL,a}$ is not a norm but only a regularization. We abuse the norm notation here for convenience. Note that

$$\lim_{a \rightarrow 0^+} ||x||_{CTL_1,a} = 1 - ||x||_0 = \begin{cases} 1 & x = 0 \\ 0 & x \neq 0 \end{cases}$$

For small choice of $a$, the value of $||x||_{CTL_1,a}$ is negligible when $|x|$ is large. The behavior of the CTL$_1$ penalty is illustrated in figure 4. To prevent the neural network from having a zero layer, we apply it to each layer of our model

$$||w||_{CTL_1,a} := \sum_{l=1}^{L} 1 - \frac{||w_l||_1}{a + ||w_l||_1}$$

By imposing this CTL$_1$ penalty, we force each layer to have some nonzero weights, so the training will not collapse. The augmented objective is

$$\min_{u \in Q} \ell(u) := f(u) + \lambda_2 ||u||_{GL} + \lambda_3 ||u||_{CTL_1,a} + \lambda_1 ||u||_1$$

(6)

As a result, we can have more ‘aggressive’ choices for the values of parameters and the adaptive scheme to further pruning the neural networks.

Algorithm 2 APDSSM with CTL$_1$ penalty

**Input**: Float weights $w^0$. Hyperparameters $\lambda_1, \lambda_2, \beta$.

**Output**: Quantized weights $u$.

for $t = 1, \cdots, 200$ do:

$$u^t = \text{Proj}_Q(w^t)$$

$$f(u^t) = f(u^t) + \gamma^t \lambda_2 ||u^t||_{GL} + \lambda_3 ||u^t||_{CTL_1,a}$$

$$w^t = w^{t-1} - \gamma^t \nabla f(u^t)$$

$$w^t = w^t - \gamma^t \beta (w^t - u^t)$$

$$w^t = \text{Proj}_{\gamma^t \lambda_1} (w^{t-1})$$

end for

$$u = \text{Proj}_Q(w^{200})$$

Table III

| Model | $\lambda_2$ initial | Ch. sp. | Wt. sp. | Accuracy |
|-------|---------------------|---------|---------|----------|
| CIFAR-10 | R.56, 1.5 $\cdot 10^{-2}$ | 73.67% | 95.80% | 90.27% |
| CIFAR-100 | R.56, 5 $\cdot 10^{-3}$ | 82.90% | 90.76% | 88.71% |
| CIFAR-10 | R.110, 5 $\cdot 10^{-4}$ | 55.12% | 80.07% | 70.75% |
| CIFAR-100 | R.110, 1 $\cdot 10^{-3}$ | 58.06% | 80.75% | 70.16% |

In Algorithm 2, we let the parameters $\lambda_1$, and $\lambda_2$, $\lambda_3$ and $\beta$ have the same adaptive scheme by multiply it by the learning rate. This scheme makes the parameters decrease slower. Hence, as shown in Table III, the channel sparsity increases significantly. The CTL$_1$ penalty allows us to further trader-off the performance to efficiency based on our needs.
In this paper, we proposed APGDSSM to integrate the penalty based channel pruning and QAT. We remark that relaxations of QAT (7, 16) will lead to sub-optimal outcomes, because such methods search the sparse subspace first and then find local optimal quantized weights around the searched sparse weights. The two subspaces need to be searched concurrently from the beginning. We verify that APGDSSM can deliver sparse quantized neural network with minor trade-off for performance. Further, we designed an auxiliary complementary transformed $l_1$ penalty to prevent training from collapsing, so we can trade more performance for efficiency if needed.

VII. CONCLUSION

In this paper, we proposed APGDSSM to integrate the penalty based channel pruning and QAT. We remark that relaxations of QAT (7, 16) will lead to sub-optimal outcomes, because such methods search the sparse subspace first and then find local optimal quantized weights around the searched sparse weights. The two subspaces need to be searched concurrently from the beginning. We verify that APGDSSM can deliver sparse quantized neural network with minor trade-off for performance. Further, we designed an auxiliary complementary transformed $l_1$ penalty to prevent training from collapsing, so we can trade more performance for efficiency if needed.

REFERENCES

[1] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM journal on imaging sciences, 2(1):183–202, 2009.

[2] Stephen Boyd, Neal Parikh, and Eric Chu. Distributed optimization and statistical learning via the alternating direction method of multipliers. Now Publishers Inc, 2011.

[3] Kevin Bui, Fredrick Park, Shuai Zhang, Yingyong Qi, and Jack Xin. Nonconvex regularization for network pruning: Compressing cnns even more. In International Symposium on Visual Computing, pages 39–53. Springer, 2020.

[4] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binarized neural networks. In ICLR, 2016.

[5] Tha Dinh, Bao Wang, Andrea Bertozzi, Stanley Osher, and Jack Xin. Sparsity meets robustness: channel pruning for the Feynman-Kac formalism principled robust deep neural nets. In International Conference on Machine Learning, Optimization, and Data Science, pages 362–381. Springer, 2020.

[6] Tha Dinh and Jack Xin. Convergence of a relaxed variable splitting method for learning sparse neural networks via $l_1$, $ℓ_0$, and transformed-$l_1$ penalties. In Proceedings of SAI Intelligent Systems Conference, pages 360–374. Springer, 2020.

[7] Tim Dockhorn, Yaojiang Yu, Eyyub Sari, Mahdi Zolnouri, and Vahid Partovi Nia. Demystifying and generalizing binaryconnect. Advances in Neural Information Processing Systems, 34, 2021.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[9] Zhijian Li, Bao Wang, and Jack Xin. An integrated approach to produce robust deep neural network models with high efficiency. In International Conference on Machine Learning, Optimization, and Data Science, pages 451–465. Springer, 2021.

[10] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In ICCLR, 2019.

[11] Jiancheng Lyu and Spencer Sheen. A channel-pruned and weight-binarized convolutional neural network for keyword spotting. In Le Th H., Le H., Pham Dinh T., Nguyen N. (eds), Advanced Computational Intelligence and Knowledge-Based Systems, volume 1121. Springer, Cham, 2020.

[12] Gavin Taylor, Ryan Burmeister, Zheng Xu, Bharat Singh, Ankkt Patel, and Tom Goldstein. Training neural networks without gradients: A scalable admm approach. In International conference on machine learning, pages 2722–2731. PMLR, 2016.

[13] Wei Wen, Chumpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In Advances in neural information processing systems, pages 2074–2082, 2016.

[14] Biao Yang, Jiancheng Lyu, Shuai Zhang, Yingyong Qi, and Jack Xin. Channel pruning for deep neural networks via a relaxed groupwise splitting method. In IEEE International Conference on Artificial Intelligence for Industries, pages 97–98, 2019.

[15] Shaokai Ye, Kaidi Xu, Siija Liu, Hao Cheng, Jan-Henrik Lambrechts, Huan Zhang, Aojun Zhou, Kaisheng Ma, Yanzi Wang, and Xue Lin. Adversarial robustness vs. model compression, or both? In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 111–120, 2019.

[16] Penghang Yin, Shuai Zhang, Jiancheng Lyu, Stanley Osher, Yingyong Qi, and Jack Xin. BinaryRelax: A Relaxation Approach for Training Deep Neural Networks with Quantized Weights. SIAM Journal on Imaging Sciences, 11(4):2205–2223, 2018.

[17] Shuai Zhang and Jack Xin. Minimization of transformed $l_1$ penalty: Closed form representation and iterative thresholding algorithms. Comm. Math Sci., 15(2):511–537, 2017.