Nonconvex Regularization for Network Slimming: Compressing CNNs Even More

Kevin Bui[1], Fredrick Park[2], Shuai Zhang[1], Yingyong Qi[1], and Jack Xin[1](✉)

[1] Department of Mathematics, University of California, Irvine, Irvine, CA 92697, USA
{kevinb3,szhang3,yqi,jack.xin}@uci.edu

[2] Department of Mathematics and Computer Science, Whittier College, Whittier 90602, CA, USA
fpark@whittier.edu

Abstract. In the last decade, convolutional neural networks (CNNs) have evolved to become the dominant models for various computer vision tasks, but they cannot be deployed in low-memory devices due to its high memory requirement and computational cost. One popular, straightforward approach to compressing CNNs is network slimming, which imposes an ℓ_1 penalty on the channel-associated scaling factors in the batch normalization layers during training. In this way, channels with low scaling factors are identified to be insignificant and are pruned in the models. In this paper, we propose replacing the ℓ_1 penalty with the ℓ_p and transformed ℓ_1 (Tℓ_1) penalties since these nonconvex penalties outperformed ℓ_1 in yielding sparser satisfactory solutions in various compressed sensing problems. In our numerical experiments, we demonstrate network slimming with ℓ_p and Tℓ_1 penalties on VGGNet and Densenet trained on CIFAR 10/100. The results demonstrate that the nonconvex penalties compress CNNs better than ℓ_1. In addition, Tℓ_1 preserves the model accuracy after channel pruning, and $\ell_{1/2,3/4}$ yield compressed models with similar accuracies as ℓ_1 after retraining.

Keywords: Convolutional neural networks · Sparse optimization · ℓ_1 regularization · ℓ_p regularization · Batch normalization · Channel pruning · Nonconvex optimization

1 Introduction

In the past years, convolutional neural networks (CNNs) evolved into superior models for various computer vision tasks, such as image classification [18,26,41] and image segmentation [10,32,38]. Unfortunately, training a highly accurate CNN is computationally demanding. State-of-the-art CNNs such as Resnet [18] can have up to at least a hundred layers and thus require millions of parameters to train and billions of floating-point-operations to execute. Consequently, deploying CNNs in low-memory devices, such as mobile smartphones, is difficult, making their real-world applications limited.

To make CNNs more practical, many works proposed several different directions to compress large CNNs or to learn smaller, more efficient models from scratch. These directions include low-rank approximation [13,23,45–47], weight quantization [11,12,27,53,59], and weight pruning [1,16,19,28]. One popular direction is to

G. Bebis et al. (Eds.): ISVC 2020, LNCS 12509, pp. 39–53, 2020.
https://doi.org/10.1007/978-3-030-64556-4_4

sparsify the CNN while training it [2,6,39,44]. Sparsity can be imposed on various types of structures existing in CNNs, such as filters and channels [44].

One interesting yet straightforward approach in sparsifying CNNs was *network slimming* [31]. This method imposes ℓ_1 regularization on the scaling factors in the batch normalization layers. Due to ℓ_1 regularization, scaling factors corresponding to insignificant channels are pushed towards zeroes, narrowing down the important channels to retain, while the CNN model is being trained. Once the insignificant channels are pruned, the compressed model may need to be retrained since pruning can degrade its original accuracy. Overall, network slimming yields a compressed model with low run-time memory and number of computing operations.

In this paper, we propose replacing ℓ_1 regularization in network slimming with an alternative nonconvex regularization that promotes better sparsity. Because the ℓ_1 norm is a convex relaxation of the ℓ_0 norm, a better penalty would be nonconvex and it would interpolate ℓ_0 and ℓ_1. Considering these properties, we examine ℓ_p [7,9,48] and transformed ℓ_1 (Tℓ_1) [56,57] because of their superior performances in recovering satisfactory sparse solutions in various compressed sensing problems. Furthermore, both regularizers have explicit formulas for their subgradients, which allow us to directly perform subgradient descent [40].

2 Related Works

2.1 Compression Techniques for CNNs

Low-rank Decomposition. Denton *et al.* [13] compressed the weight tensors of convolutional layers using singular value decomposition to approximate them. Jaderberg *et al.* [23] exploited the redundancy between different feature channels and filters to approximate a full-rank filter bank in CNNs by combinations of a rank-one filter basis. These methods focus on decomposing pre-trained weight tensors. Wen *et al.* [45] proposed *force regularization* to train a CNN towards having a low-rank representation. Xu *et al.* [46,47] proposed trained rank pruning, an optimization scheme that incorporates low-rank decomposition into the training process. Trained rank pruning is further strengthened by nuclear norm regularization.

Weight Quantization. Quantization aims to represent weights with low-precision ($\leq$8 bits arithmetic). The simplest form of quantization is binarization, constraining weights to only two values. Courbariaux *et al.* [12] proposed BinaryConnect, a method that trains deep neural networks (DNNs) with strictly binary weights. Neural networks with ternary weights have also been developed and investigated. Li *et al.* [27] proposed ternary weight networks, where the weights are only $-1, 0$, or $+1$. Zhu *et al.* [59] proposed Trained Ternary Quantization that constrains the weights to more general values $-W^n, 0$, and W^p, where W^n and W^p are parameters learned through the training process. For more general quantization, Yin *et al.* [53] proposed BinaryRelax, which relaxes the quantization constraint into a continuous regularizer for the optimization problem needed to be solved in CNNs.

Pruning. Han *et al.* [16] proposed a three-step framework to first train a CNN, prune weights if below a fixed threshold, and retrain the compressed CNN. Aghasi *et al.* [1]

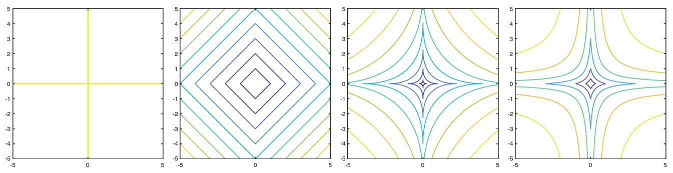

(a) ℓ_0 (b) ℓ_1 (c) $\ell_{1/2}$ (d) $\mathrm{T}\ell_1, a = 1$

Fig. 1. Contour plots of sparse regularizers.

proposed using convex optimization to determine which weights to prune while preserving model accuracy. For CNNs, channel or filter pruning are preferred over individual weight pruning since the former significantly eliminates more unnecessary weights. Li *et al.* [28] calculated the sum of absolute weights for each filter of the CNN and pruned the filters with the lowest sums. On the other hand, Hu *et al.* [19] proposed a metric that measures the redundancies in channels to determine which to prune. Network slimming [31] is also another method of channel pruning since it prunes channels with the lowest associated scaling factors.

Sparse Optimization. Sparse optimization methods aim to train DNNs towards a compressed structure from scratch by introducing a sparse regularization term to the objective function being minimized. BinaryRelax [53] and network slimming [31] are examples of sparse optimization methods for CNNs. Alvarez and Salzmann [2] and Scardapane *et al.* [39] applied group lasso [55] and sparse group lasso [39] to CNNs to obtain group-sparse networks. Non-convex regularizers have also been examined recently. Xue and Xin [51] applied ℓ_0 and transformed ℓ_1 to three-layer CNNs that classify shaky vs. normal handwriting. Ma *et al.* [36] proposed integrated $\mathrm{T}\ell_1$, which combines group sparsity and $\mathrm{T}\ell_1$, and applied it to CNNs for image classification.

2.2 Regularization Penalty

Let $x = (x_1, \dots, x_n) \in \mathbb{R}^n$. The ℓ_1 penalty is described by

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|, \tag{1}$$

while the ℓ_0 penalty is described by

$$\|x\|_0 = \sum_{i=1}^{n} \mathbb{1}_{\{x_i \neq 0\}}, \quad \text{where} \quad \mathbb{1}_{\{z \neq 0\}} = \begin{cases} 1 & \text{if } z \neq 0 \\ 0 & \text{if } z = 0. \end{cases} \tag{2}$$

Although ℓ_1 regularization is popular in sparse optimization in various applications such as compressed sensing [3,4,54] and compressive imaging [24,35], it may not actually yield the sparsest solution [7,33,34,48,57]. Moreover, it is sensitive to outliers and it may yield biased solutions [15].

A nonconvex alternative to the ℓ_1 penalty is the ℓ_p penalty

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} \tag{3}$$

for $p \in (0, 1)$. The ℓ_p penalty interpolates ℓ_0 and ℓ_1 because as $p \to 0^+$, we have $\ell_p \to \ell_0$, and as $p \to 1^-$, we have $\ell_p \to \ell_1$. It was shown to recover sparser solution than did ℓ_1 for certain compressed sensing problems [8,9]. Empirical studies [9,49] demonstrated that for $p \in [1/2, 1)$, as p decreases, the solution becomes sparser by ℓ_p minimization, but for $p \in (0, 1/2)$, the performance becomes no longer significant. In [50], $\ell_{1/2}$ was verified to be an unbiased estimator. Moreover, it demonstrated success in image deconvolution [5,25], hyperspectral unmixing [37], and image segmentation [29]. Numerically, in compressed sensing, a small value ϵ is added to x_i to avoid blowup in the subgradient when $x_i = 0$. In this work, we will examine across different values of p since ℓ_p regularization may work differently in deep learning than in other areas.

Lastly, the Tℓ_1 penalty is formulated as

$$P_a(x) = \sum_{i=1}^{n} \frac{(a+1)|x_i|}{a + |x_i|} \tag{4}$$

for $a > 0$. Tℓ_1 interpolates ℓ_0 and ℓ_1 because as $a \to 0^+$, we have T$\ell_1 \to \ell_0$, and as $a \to +\infty$, we have T$\ell_1 \to \ell_1$. This penalty enjoys three properties – unbiasedness, sparsity, and continuity – that a sparse regularizer should have [15]. The Tℓ_1 penalty was demonstrated to be robust by outperforming ℓ_1 and ℓ_p in compressed sensing problems with both coherent and incoherent sensing matrices [56,57]. Additionally, the Tℓ_1 penalty yields satisfactory, sparse solutions in matrix completion [58] and deep learning [36].

Figure 1 displays the contour plots of the aforementioned regularizers. With ℓ_1 regularization, the solution tends to coincide with one of the corners of the rotated squares, making it sparse. For $\ell_{1/2}$ and Tℓ_1, the level lines are more curved compared to ℓ_1, which encourages the solutions to coincide with one of the corners. Hence, solutions tend to be sparser with $\ell_{1/2}$ and Tℓ_1 regularization than with ℓ_1 regularization.

3 Proposed Method

3.1 Batch Normalization Layer

Batch normalization [22] has been instrumental in speeding the convergence and improving generalization of many deep learning models, especially CNNs [18,43]. In most state-of-the-arts CNNs, a convolutional layer is always followed by a batch normalization layer. Within a batch normalization layer, features generated by the preceding convolutional layer are normalized by their mean and variance within the same

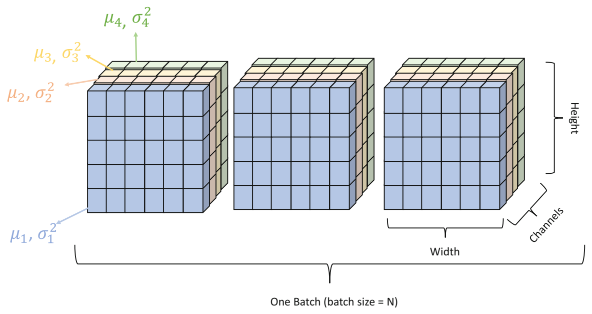


Fig. 2. Visualization of batch normalization on a feature map. The mean and variance of the values of the pixels of the same colors corresponding to the channels are computed and are used to normalize these pixels.

channel. Afterward, a linear transformation is applied to compensate for the loss of their representative abilities.

We mathematically describe the process of the batch normalization layer. First we suppose that we are working with 2D images. Let x be a feature computed by a convolutional layer. Its entry x_i is indexed by We define the index set $S_i = \{k : k_C = i_C\}$, where k_C and i_C are the respective subindices of k and i along the C axis. The mean μ_i and variance σ_i^2 are computed as follows:

$$\mu_i = \frac{1}{|S_i|} \sum_{k \in S_i} x_k, \quad \sigma_i^2 = \frac{1}{|S_i|} \sum_{k \in S_i} (x_k - \mu_i)^2 + \epsilon \tag{5}$$

for some small value $\epsilon > 0$, where $|\mathcal{A}|$ denotes the cardinality of the set $\mathcal{A}$. Then x is normalized as $\hat{x}_i = \frac{x_i - \mu_i}{\sigma_i}$ for each index i. In short, the mean and variance are computed from pixels of the same channel index, and these values are used to normalize these pixels. Visualization is provided in Fig. 2. Lastly, the output of the batch normalization layer is computed as a linear transformation of the normalized features:

$$y_i = \gamma_{i_C} \hat{x}_i + \beta_{i_C}, \tag{6}$$

where $\gamma_{i_C}, \beta_{i_C} \in \mathbb{R}$ are trainable parameters.

3.2 Network Slimming with Nonconvex Sparse Regularization

Since the scaling factors γ_{i_C}'s in (6) are associated with the channels of a convolutional layer, we aim to penalize them with a sparse regularizer in order to determine which channels are irrelevant to the compressed CNN model. Suppose we have a training dataset that consists of N input-output pairs $\{(x_i, y_i)\}_{i=1}^N$ and a CNN with L convolutional layers, where each is followed by a batch normalization layer. Then we have a set of vectors $\{(\gamma_l, \beta_l)\}_{l=1}^L$ for each layer l, where $\gamma_l = (\gamma_{l,1}, \dots, \gamma_{l,C_l})$ and $\beta_l = (\beta_{l,1}, \dots, \beta_{l,C_l})$ with C_l being the number of channels in the lth convolutional layer. Let W be the set of weight parameters such that $\{(\gamma_l, \beta_l)\}_{l=1}^L \subset W$. Hence, the

Table 1. Sparse regularizers and their subgradients.

Name	$\mathcal{R}(x)$	$\partial R(x)$
ℓ_1	$\|x\|_1 = \sum_{i=1}^{n} \|x_i\|$	$\partial \|x\|_1 = \left\{ z \in \mathbb{R}^n : z_i = \begin{cases} \text{sgn}(x_i) & \text{if } x_i \neq 0 \\ z_i \in [-1,1] & \text{if } x_i = 0 \end{cases} \right\}$
ℓ_p	$\|x\|_p^p = \sum_{i=1}^{n} \|x_i\|^p$	$\partial \|x\|_p^p = \left\{ z \in \mathbb{R}^n : z_i = \begin{cases} \frac{p \cdot \text{sgn}(x_i)}{\|x_i\|^{1-p}} & \text{if } x_i \neq 0 \\ z_i \in \mathbf{R} & \text{if } x_i = 0 \end{cases} \right\}$
Tℓ_1	$P_a(x) = \sum_{i=1}^{n} \frac{(a+1)\|x_i\|}{a+\|x_i\|}$	$\partial P_a(x) = \left\{ z \in \mathbb{R}^n : z_i = \begin{cases} \frac{a(a+1)\text{sgn}(x_i)}{(a+\|x_i\|)^2} & \text{if } x_i \neq 0 \\ 0 & \text{if } x_i = 0 \end{cases} \right\}$

trainable parameters W of the CNN are learned by minimizing the following objective function:

$$\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(h(x_i, W), y_i) + \lambda \sum_{l=1}^{L}\mathcal{R}(\gamma_l), \tag{7}$$

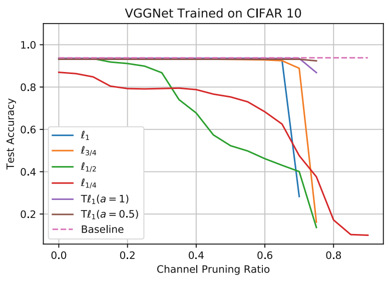
where $h(\cdot,\cdot)$ is the output of the CNN used for prediction, $\mathcal{L}(\cdot,\cdot)$ is a loss function, $\mathcal{R}(\cdot)$ is a sparse regularizer, and $\lambda > 0$ is a regularization parameter for $\mathcal{R}(\cdot)$. When $\mathcal{R}(\cdot) = \|\cdot\|_1$, we have the original network slimming method. As mentioned earlier, since ℓ_1 regularization may not yield the sparsest solution, we investigate the method with a nonconvex regularizer, where $\mathcal{R}(\cdot)$ is $\|\cdot\|_p^p$ or $P_a(\cdot)$. To minimize (7) in general, stochastic gradient descent is applied to the first term while subgradient descent is applied to the second term [40]. Subgradients of the regularizers are presented in Table 1. After the CNN is trained, channels with low scaling factors are pruned, leaving us with a compressed model.

4 Experiments

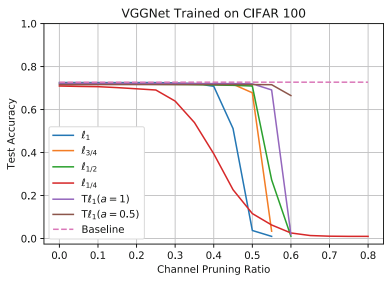
We apply the proposed nonconvex network slimming with ℓ_p and Tℓ_1 regularization on CIFAR 10/100 datasets on VGGNet [41] and Densenet [20]. Code for the experiments is given at https://github.com/kbui1993/NonconvexNetworkSlimming.

Both sets of CIFAR 10/100 consist of 32×32 natural images. CIFAR 10 has 10 classes; CIFAR 100 has 100 classes. CIFAR 10/100 is split between a training set of 50,000 images and a test set of 10,000 images. Standard augmentation [18,21,30] is applied to the CIFAR 10/100 images.

For our experiments, we train VGGNet with 19 layers and Densenet with 40 layers for five runs with and without scaling-factor regularization as done in [31]. (We refer "regularized models" as the models with scaling-factor regularization.) On CIFAR 10/100, the models are trained for 160 epochs with a training batch size of 64. They are optimized using stochastic gradient descent with learning rate 0.1. The learning rate decreases by a factor of 10 after 80 and 120 epochs. We use weight decay of 10^{-4} and Nesterov momentum [42] of 0.9 without dampening. Weight initialization is based on [17] and scaling factor initialization is set to all be 0.5 as done in [31].



(a) VGGNet trained on CIFAR 10 with $\lambda = 10^{-4}$



(b) VGGNet trained on CIFAR 100 with $\lambda = 10^{-4}$

Fig. 3. Effect of channel pruning ratio on the mean test accuracy of five runs of VGGNet on CIFAR 10/100. Baseline refers to the mean test accuracy of the unregularized model that is not pruned.

Table 2. Effect of channel pruning ratio on the mean pruned ratio of parameters of five runs of VGGNet trained on CIFAR 10/100 for each regularization.

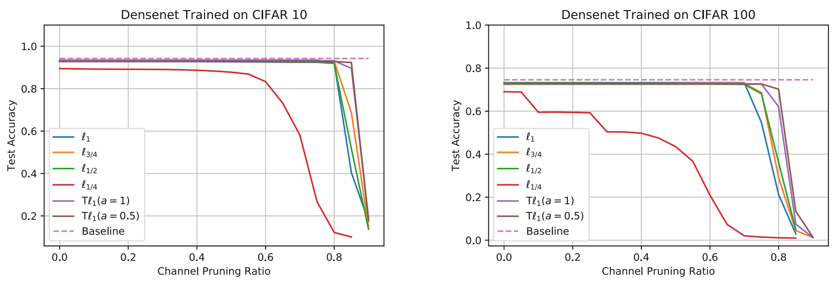
Pruning Ratio	CIFAR 10						CIFAR 100					
	ℓ_1	$\ell_{3/4}$	$\ell_{1/2}$	$\ell_{1/4}$	Tℓ_1 $(a=1)$	Tℓ_1 $(a=0.5)$	ℓ_1	$\ell_{3/4}$	$\ell_{1/2}$	$\ell_{1/4}$	Tℓ_1 $(a=1)$	Tℓ_1 $(a=0.5)$
0.10	0.2110	0.2114	0.2112	0.1995	**0.2116**	0.2094	0.2191	0.2198	**0.2202**	0.2200	0.2187	0.2167
0.20	0.3934	0.3955	**0.3962**	0.3766	0.3935	0.3929	0.4036	0.4064	**0.4085**	0.4071	0.4047	0.4033
0.30	0.5488	0.5513	**0.5529**	0.5299	0.5494	0.5492	0.5583	0.5604	**0.5629**	0.5621	0.5599	0.5597
0.40	0.6756	0.6796	**0.6809**	0.6620	0.6788	0.6783	0.6745	0.6801	0.6841	**0.6853**	0.6822	0.6849
0.50	0.7753	0.7799	0.7810	0.7707	0.7806	**0.7822**	0.7535	0.7654	0.7719	0.7816	0.7718	**0.7799**
0.60	0.8471	0.8524	0.8543	0.8576	0.8555	**0.8592**	N/A	N/A	0.8307	**0.8571**	0.8290	0.8409
0.70	0.8881	0.8969	0.9001	**0.9214**	0.9034	0.9088	N/A	N/A	N/A	**0.9148**	N/A	N/A
0.80	N/A	N/A	N/A	**0.9654**	N/A	N/A	N/A	N/A	N/A	**0.9654**	N/A	N/A
0.90	N/A	N/A	N/A	**0.9905**	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

With regularization parameter $\lambda = 10^{-4}$, we train the regularized models with ℓ_1, $\ell_p(p = 0.25, 0.5, 0.75)$, and T$\ell_1(a = 0.5, 1)$ penalties on the scaling factors.

4.1 Channel Pruning

After training, we prune the regularized models globally. In particular, we specify a ratio such as 0.35 or a percentage such as 35%, determine the 35th percentile among all scaling factors of the network and set it as a threshold, and prune away channels whose scaling factors are below that threshold. After pruning, we compute the compressed networks' mean test accuracies. Mean test accuracies are compared against the baseline test accuracy computed from the unregularized models. We evaluate the mean test accuracies as we increase the channel pruning ratios in increment of 0.05 to the point where a layer has no more channels.

For VGGNet, the mean test accuracies across the channel pruning ratios are shown in Fig. 3. The mean pruned ratios of parameters (the number of parameters pruned to the total number of parameters) are shown in Table 2. For CIFAR 10, according to Fig. 3a, the mean test accuracies for $\ell_{1/2}$ and $\ell_{1/4}$ are not robust against pruning



(a) Densenet-40 trained on CIFAR 10 with $\lambda = 10^{-4}$

(b) Densenet-40 trained on CIFAR 100 with $\lambda = 10^{-4}$

Fig. 4. Effect of channel pruning ratio on the mean test accuracy of five runs of Densenet on CIFAR 10/100. Baseline refers to the mean test accuracy of the unregularized model that is not pruned.

Table 3. Effect of channel pruning ratio on the mean pruned ratio of parameters of five runs of Densenet-40 trained on CIFAR 10/100 for each regularization.

Pruning Ratio	CIFAR 10						CIFAR 100					
	ℓ_1	$\ell_{3/4}$	$\ell_{1/2}$	$\ell_{1/4}$	Tℓ_1 $(a=1)$	Tℓ_1 $(a=0.5)$	ℓ_1	$\ell_{3/4}$	$\ell_{1/2}$	$\ell_{1/4}$	Tℓ_1 $(a=1)$	Tℓ_1 $(a=0.5)$
0.10	0.0922	0.0932	0.0933	**0.0935**	**0.0935**	**0.0935**	0.0918	0.0919	0.0920	**0.0926**	**0.0926**	0.0925
0.20	0.1835	0.1864	0.1859	0.1871	0.1863	**0.1872**	0.1834	0.1839	0.1841	**0.1853**	0.1846	0.1849
0.30	0.2757	0.2787	0.2797	**0.2813**	0.2785	0.2808	0.2753	0.2757	0.2762	**0.2785**	0.2772	0.2775
0.40	0.3673	0.3714	0.3726	**0.3752**	0.3717	0.3739	0.3669	0.3676	0.3685	**0.3717**	0.3691	0.3698
0.50	0.4595	0.4642	0.4662	**0.4705**	0.4641	0.4673	0.4584	0.4595	0.4606	**0.4651**	0.4615	0.4624
0.60	0.5515	0.5562	0.5588	**0.5669**	0.5573	0.5616	0.5498	0.5513	0.5526	**0.5594**	0.5535	0.5546
0.70	0.6438	0.6490	0.6512	**0.6656**	0.6514	0.6549	0.6412	0.6433	0.6444	**0.6573**	0.6455	0.6471
0.80	0.7375	0.7425	0.7447	**0.7702**	0.7446	0.7488	0.7339	0.7356	0.7367	**0.7628**	0.7378	0.7392
0.90	0.8376	0.8402	0.8436	N/A	0.8423	**0.8445**	N/A	0.8334	N/A	N/A	0.8348	**0.8360**

since they gradually decrease as the channel pruning ratio increases. On the other hand, $\ell_{3/4}$ and Tℓ_1 are more robust than ℓ_1 to channel pruning since their accuracies drop at higher pruning ratios. So far, we see T$\ell_1(a = 0.5)$ to be the most robust with its mean test accuracy to be close to its pre-pruned mean test accuracy. For CIFAR 100, in Fig. 3b, ℓ_1 is less robust than $\ell_{3/4}$, $\ell_{1/2}$ and Tℓ_1. Like for CIFAR 10, T$\ell_1(a = 0.5)$ is the most robust since its accuracy does not drop off until after 55% of channels are pruned while the accuracies of the other regularizers drop by when 50% of channels are pruned. According to Table 2, the pruned ratio of parameters are comparable among the regularizers for each channel pruning percentage, but always a nonconvex regularizer prunes more weight parameters than does ℓ_1.

For Densenet-40, the mean test accuracies across the channel pruning ratios are depicted in Fig. 4. The mean pruned ratios of parameters are shown in Table 3. For both CIFAR 10/100, $\ell_{1/4}$ is the least robust among the regularizers and following it is ℓ_1. T$\ell_1(a = 0.5)$ is the most robust because its test accuracy drops at a higher pruning ratio than do other regularizers. According to Table 3, ℓ_1 compresses the models the least while generally $\ell_{1/4}$ prunes the most number of parameters for both CIFAR 10/100.

Overall, we see that as $p \to 0^+$, ℓ_p regularization tends to prune more weight parameters, but its mean test accuracy decreases and it becomes less robust against pruning. Because smaller value of p strongly encourages sparsity, many of the scaling factors are close to zeroes, causing their respective subgradients to become larger and thus affecting the model accuracy. For Tℓ_1, $a = 0.5$ manages to prune more weight parameters than does $a = 1.0$ and it improves the robustness of the model against pruning.

Table 4. Results from retrained VGGNet on CIFAR 10/100 after pruning. Baseline refers to the VGGNet model trained without regularization on the scaling factors.

	Number of Parameters	Pruning Percentage (%)	Average Test Accuracy before Retraining (%)	Average Test Accuracy after Retraining (%)
Baseline	20.04M	0.00	93.83	N/A
ℓ_1 (0% Pruned)	20.04M	0.00	93.63	N/A
ℓ_1 (70% Pruned)	2.24M	88.81	28.28	93.91
$\ell_{3/4}$ (0% Pruned)	20.04M	0.00	93.53	N/A
$\ell_{3/4}$ (70% Pruned)	2.07M	89.69	88.87	93.90
$\ell_{3/4}$ (75% Pruned)	1.79M	91.06	16.18	93.79
$\ell_{1/2}$ (0% Pruned)	20.04M	0.00	93.57	N/A
$\ell_{1/2}$ (70% Pruned)	2.00M	90.01	40.07	93.77
$\ell_{1/2}$ (75% Pruned)	1.66M	91.70	13.65	93.82
$\ell_{1/4}$ (0% Pruned)	20.04M	0.00	86.97	N/A
$\ell_{1/4}$ (70% Pruned)	1.58M	92.14	47.59	92.15
$\ell_{1/4}$ (90% Pruned)	0.19M	99.05	10.00	81.57
T$\ell_1(a = 1)$ (0% Pruned)	20.04M	0.00	93.55	N/A
T$\ell_1(a = 1)$ (70% Pruned)	1.93M	90.35	93.54	93.86
T$\ell_1(a = 1)$ (75% Pruned)	1.66M	91.71	86.83	93.82
T$\ell_1(a = 0.5)$ (0% Pruned)	20.04M	0.00	93.15	N/A
T$\ell_1(a = 0.5)$ (70% Pruned)	1.83M	90.88	93.14	93.75
T$\ell_1(a = 0.5)$ (75% Pruned)	1.53M	92.38	92.38	93.77

(a) CIFAR 10

	Number of Parameters	Pruning Percentage (%)	Average Test Accuracy before Retraining (%)	Average Test Accuracy after Retraining (%)
Baseline	20.08M	0.00	72.73	N/A
ℓ_1 (0% Pruned)	20.08M	0.00	72.57	N/A
ℓ_1 (55% Pruned)	4.31M	78.53	1.00	72.98
$\ell_{3/4}$ (0% Pruned)	20.08M	0.00	72.14	N/A
$\ell_{3/4}$ (55% Pruned)	4.10M	79.59	3.40	73.26
$\ell_{1/2}$ (0% Pruned)	20.08M	0.00	72.06	N/A
$\ell_{1/2}$ (55% Pruned)	3.95M	80.35	27.32	73.25
$\ell_{1/2}$ (60% Pruned)	3.40M	91.70	1.08	71.45
$\ell_{1/4}$ (0% Pruned)	20.08M	0.00	70.95	N/A
$\ell_{1/4}$ (55% Pruned)	3.58M	82.19	6.30	72.20
$\ell_{1/4}$ (80% Pruned)	0.69M	99.05	1.00	15.43
T$\ell_1(a = 1)$ (0% Pruned)	20.08M	0.00	72.07	N/A
T$\ell_1(a = 1)$ (55% Pruned)	3.94M	80.37	69.13	73.08
T$\ell_1(a = 1)$ (60% Pruned)	3.43M	91.71	1.84	72.93
T$\ell_1(a = 0.5)$ (0% Pruned)	20.08M	0.00	71.63	N/A
T$\ell_1(a = 0.5)$ (55% Pruned)	3.72M	81.46	71.57	72.69
T$\ell_1(a = 0.5)$ (60% Pruned)	3.20M	92.38	66.50	72.61

(b) CIFAR 100

4.2 Retraining After Pruning

After a model is pruned, we retrain it without regularization on the scaling factors with the same optimization setting as the first time training it. The purpose of retraining is to at least recover the model's original accuracy prior to pruning. For VGGNet, the results are shown in Table 4; for Densenet-40, the results are shown in Table 5.

Table 5. Results from retrained Densenet-40 on CIFAR 10/100 after pruning. Baseline refers to the Densenet-40 model trained without regularization on the scaling factors.

	Number of Parameters	Pruning Percentage (%)	Average Test Accuracy before Retraining (%)	Average Test Accuracy after Retraining (%)
Baseline	1.02M	0.00	94.25	N/A
ℓ_1 (0 % Pruned)	1.02M	0.00	93.46	N/A
ℓ_1 (82.5% Pruned)	0.25M	76.21	78.27	93.46
ℓ_1 (90% Pruned)	0.17M	83.76	17.47	91.42
$\ell_{3/4}$ (0% Pruned)	1.02M	0.00	93.19	N/A
$\ell_{3/4}$ (82.5% Pruned)	0.25M	76.57	90.17	93.33
$\ell_{3/4}$ (90% Pruned)	0.16M	84.02	15.06	91.54
$\ell_{1/2}$ (0% Pruned)	1.02M	0.00	93.28	N/A
$\ell_{1/2}$ (82.5% Pruned)	0.25M	76.84	83.17	93.43
$\ell_{1/2}$ (90% Pruned)	0.16M	84.36	13.76	91.31
$\ell_{1/4}$ (0% Pruned)	1.02M	0.00	89.48	N/A
$\ell_{1/4}$ (82.5% Pruned)	0.22M	79.81	11.29	91.68
$\ell_{1/4}$ (85% Pruned)	0.18M	82.57	10.05	91.44
T$\ell_1(a = 1)$ (0% Pruned)	1.02M	0.00	93.16	N/A
T$\ell_1(a = 1)$ (82.5% Pruned)	0.25M	76.80	93.17	93.26
T$\ell_1(a = 1)$ (90% Pruned)	0.16M	84.23	18.91	91.70
T$\ell_1(a = 0.5)$ (0% Pruned)	1.02M	0.00	92.78	N/A
T$\ell_1(a = 0.5)$ (82.5% Pruned)	0.24M	77.21	92.74	93.05
T$\ell_1(a = 0.5)$ (90% Pruned)	0.16M	84.45	18.12	91.69

(a) CIFAR 10

	Number of Parameters	Pruning Percentage (%)	Average Test Accuracy before Retraining (%)	Average Test Accuracy after Retraining (%)
Baseline	1.06M	0.00	74.58	N/A
ℓ_1 (0% Pruned)	1.06M	0.00	73.24	N/A
ℓ_1 (75% Pruned)	0.35M	68.74	54.68	73.73
ℓ_1 (85% Pruned)	0.23M	78.08	2.94	72.40
$\ell_{3/4}$ (0% Pruned)	1.06M	0.00	72.97	N/A
$\ell_{3/4}$ (75% Pruned)	0.34M	68.93	68.60	73.75
$\ell_{3/4}$ (85% Pruned)	0.23M	78.26	4.44	72.63
$\ell_{3/4}$ (90% Pruned)	0.18M	83.34	1.23	69.33
$\ell_{1/2}$ (0% Pruned)	1.06M	0.00	72.98	N/A
$\ell_{1/2}$ (75% Pruned)	0.34M	69.13	66.59	73.39
$\ell_{1/2}$ (85% Pruned)	0.23M	78.42	5.05	72.52
$\ell_{1/4}$ (0% Pruned)	1.06M	0.00	69.02	N/A
$\ell_{1/4}$ (75% Pruned)	0.32M	70.81	7.25	71.62
$\ell_{1/4}$ (85% Pruned)	0.19M	82.28	1.00	67.76
T$\ell_1(a = 1)$ (0% Pruned	1.06M	0.00	72.63	N/A
T$\ell_1(a = 1)$ (75% Pruned)	0.34M	69.13	72.34	73.42
T$\ell_1(a = 1)$ (85% Pruned)	0.23M	78.47	7.5	72.52
T$\ell_1(a = 1)$ (90% Pruned)	0.18M	83.49	1.24	69.98
T$\ell_1(a = 0.5)$ (0% Pruned)	1.06M	0.00	72.57	N/A
T$\ell_1(a = 0.5)$ (75% Pruned)	0.34M	69.33	72.59	73.23
T$\ell_1(a = 0.5)$ (85% Pruned)	0.23M	78.58	13.41	72.56
T$\ell_1(a = 0.5)$ (90% Pruned)	0.17M	83.60	1.37	70.16

(b) CIFAR 100

For VGGNet on CIFAR 10, we examine models pruned at 70%, the highest percentage that ℓ_1-regularized models can be pruned at. According to Table 4a, the nonconvex regularized models, except for $\ell_{1/4}$, attain similar mean test accuracy after retraining as the ℓ_1-regularized models. However, test accuracies of only ℓ_1, $\ell_{3/4}$, and T$\ell_1(a = 1.0)$ exceed the baseline mean test accuracy. Although ℓ_1 has higher test accuracy than other nonconvex regularized models, it is less compressed than the other regularized models. We also examine higher percentages for other nonconvex regularized models. Mean test accuracies improve for $\ell_{1/2}$ and T$\ell_1(a = 0.5)$, but they drop slightly for most other models. $\ell_{1/4}$ experiences the worst decrease, but it is due to having 90% of its channel pruned, resulting in significantly more weight parameters pruned compared to other nonconvex regularized models.

For VGGNet on CIFAR 100, we examine the mean test accuracy at 55%, the highest percentage that the ℓ_1-regularized models can be pruned at. By Table 4b, only $\ell_{3/4}$, $\ell_{1/2}$, and T$\ell_1(a = 1.0)$ outperform ℓ_1 in terms of compression and mean test accuracy. Increasing the pruning percentages higher for some other models, we observe slight decrease in test accuracies for $\ell_{1/2}$ and T$\ell_1(a = 0.5, 1)$. The $\ell_{1/4}$-regularized models are unable to recover its original test accuracy as evident by their mean test accuracy of 15.43% with 80% of channels pruned.

For Densenet-40 on CIFAR 10, from Table 5a, when 82.5% channels are pruned, ℓ_1 has the least number of weight parameters pruned. In addition, with better compression, the other nonconvex regularized models have slightly lower mean test accuracies after retraining. Models regularized with $\ell_{1/4}$ have the worst mean test accuracy of 91.68%. Increasing the channel pruning percentages, we observe that the mean test accuracies decrease from at least 93% to 91–92% for all models, except $\ell_{1/4}$. Models regularized with $\ell_{3/4}$ and T$\ell_1(a = 0.5, 1)$ have higher mean test accuracy and less weight parameters than models regularized with ℓ_1. For this set of models, the trade off between accuracy and compression is apparent.

In Table 5b, all regularized models, except for $\ell_{1/4}$ have at least 73% as their mean test accuracies after pruning 75% of their total channels and retraining them. The ℓ_1 regularized models are the least compressed compared to the nonconvex regularized models. Pruning at least 85% of the total channels decreases the mean test accuaracies after retraining. Again, accuracy is sacrificed by compressing the models even further.

5 Conclusion

We suggest a novel improvement to the network slimming method by replacing the ℓ_1 penalty with either the ℓ_p or Tℓ_1 penalties on the scaling factors in the batch normalization layer. We demonstrate the effectiveness of the nonconvex regularizers with VGGNet and Densenet-40 trained on CIFAR 10/100 in our experiments. We observe that nonconvex regularizers compress the models more than ℓ_1 at the same channel pruning ratios. In addition, Tℓ_1 preserves the model accuracy against channel pruning, while $\ell_{3/4}$ and $\ell_{1/2}$ result in more compressed models than does ℓ_1 with similar or higher model accuracy after retraining the pruned models. Hence, if deep learning practitioners do not have the option to retrain a compressed model, they should select Tℓ_1 penalty for network slimming. Otherwise, they should choose $\ell_p, p \geq 0.5$ for a model with better accuracy attained after retraining. For future direction, we plan to apply relaxed variable splitting method [14] to regularization of the scaling factors in order to apply other nonconvex regularizers such as $\ell_1 - \ell_2$ [34,52].

Acknowledgments. The work was partially supported by NSF grants IIS-1632935, DMS-1854434, DMS-1952644, and a Qualcomm Faculty Award. The authors thank Mingjie Sun for having the code for [31] available on GitHub.

References

1. Aghasi, A., Abdi, A., Romberg, J.: Fast convex pruning of deep neural networks. SIAM J. Math. Data Sci. **2**(1), 158–188 (2020)
2. Alvarez, J.M., Salzmann, M.: Learning the number of neurons in deep networks. In: Advances in Neural Information Processing Systems. pp. 2270–2278 (2016)
3. Candès, E.J., Romberg, J., Tao, T.: Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. IEEE Trans. Inf. Theory **52**(2), 489–509 (2006)
4. Candès, E.J., Romberg, J.K., Tao, T.: Stable signal recovery from incomplete and inaccurate measurements. Commun. Pure Appl. Math. **59**(8), 1207–1223 (2006)
5. Cao, W., Sun, J., Xu, Z.: Fast image deconvolution using closed-form thresholding formulas of $L_q(q = 1/2, 2/3)$ regularization. J. Vis. Commun. Image Represent. **24**(1), 31–41 (2013)
6. Changpinyo, S., Sandler, M., Zhmoginov, A.: The power of sparsity in convolutional neural networks. arXiv preprint arXiv:1702.06257 (2017)
7. Chartrand, R.: Exact reconstruction of sparse signals via nonconvex minimization. IEEE Signal Process. Lett. **14**(10), 707–710 (2007)
8. Chartrand, R., Staneva, V.: Restricted isometry properties and nonconvex compressive sensing. Inverse Prob. **24**(3), 035020 (2008)
9. Chartrand, R., Yin, W.: Iteratively reweighted algorithms for compressive sensing. In: 2008 IEEE International Conference on Acoustics, Speech and Signal Processing. pp. 3869–3872. IEEE (2008)
10. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE Trans. Pattern Anal. Mach. Intell. **40**(4), 834–848 (2017)
11. Chen, W., Wilson, J., Tyree, S., Weinberger, K., Chen, Y.: Compressing neural networks with the hashing trick. In: International conference on machine learning. pp. 2285–2294 (2015)
12. Courbariaux, M., Bengio, Y., David, J.P.: Binaryconnect: Training deep neural networks with binary weights during propagations. In: Advances in neural information processing systems. pp. 3123–3131 (2015)
13. Denton, E.L., Zaremba, W., Bruna, J., LeCun, Y., Fergus, R.: Exploiting linear structure within convolutional networks for efficient evaluation. In: Advances in neural information processing systems. pp. 1269–1277 (2014)
14. Dinh, T., Xin, J.: 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. pp. 360–374. Springer (2020)
15. Fan, J., Li, R.: Variable selection via nonconcave penalized likelihood and its oracle properties. J. Am. Stat. Assoc. **96**(456), 1348–1360 (2001)
16. Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. In: Advances in Neural Information Processing Systems. pp. 1135–1143 (2015)

17. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1026–1034 (2015)
18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)
19. Hu, H., Peng, R., Tai, Y.W., Tang, C.K.: Network trimming: a data-driven neuron pruning approach towards efficient deep architectures. arXiv preprint arXiv:1607.03250 (2016)
20. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4700–4708 (2017)
21. Huang, G., Sun, Yu., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 646–661. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_39
22. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning. pp. 448–456 (2015)
23. Jaderberg, M., Vedaldi, A., Zisserman, A.: Speeding up convolutional neural networks with low rank expansions. arXiv preprint arXiv:1405.3866 (2014)
24. Jung, H., Ye, J.C., Kim, E.Y.: Improved k-t blast and k-t sense using focuss. Phys. Med. Biol. **52**(11), 3201 (2007)
25. Krishnan, D., Fergus, R.: Fast image deconvolution using hyper-laplacian priors. In: Advances in Neural Information Processing Systems. pp. 1033–1041 (2009)
26. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems. pp. 1097–1105 (2012)
27. Li, F., Zhang, B., Liu, B.: Ternary weight networks. arXiv preprint arXiv:1605.04711 (2016)
28. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710 (2016)
29. Li, Y., Wu, C., Duan, Y.: The TVp regularized mumford-shah model for image labeling and segmentation. IEEE Trans. Image Process. **29**, 7061–7075 (2020)
30. Lin, M., Chen, Q., Yan, S.: Network in network. arXiv preprint arXiv:1312.4400 (2013)
31. Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., Zhang, C.: Learning efficient convolutional networks through network slimming. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2736–2744 (2017)
32. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3431–3440 (2015)
33. Lou, Y., Osher, S., Xin, J.: Computational aspects of constrained $L_1 - L_2$ minimization for compressive sensing. In: Le Thi, H.A., Pham Dinh, T., Nguyen, N.T. (eds.) Modelling, Computation and Optimization in Information Systems and Management Sciences. AISC, vol. 359, pp. 169–180. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18161-5_15
34. Lou, Y., Yin, P., He, Q., Xin, J.: Computing sparse representation in a highly coherent dictionary based on difference of L_1 and L_2. J. Sci. Comput. **64**(1), 178–196 (2015)
35. Lustig, M., Donoho, D., Pauly, J.M.: Sparse mri: the application of compressed sensing for rapid mr imaging. Magn. Res. Med. An Off. J. Int. Soc. Magn. Res. Med. **58**(6), 1182–1195 (2007)
36. Ma, R., Miao, J., Niu, L., Zhang, P.: Transformed ℓ_1 regularization for learning sparse deep neural networks. Neural Networks **119**, 286–298 (2019)

37. Qian, Y., Jia, S., Zhou, J., Robles-Kelly, A.: Hyperspectral unmixing via $L_{1/2}$ sparsity-constrained nonnegative matrix factorization. IEEE Trans. Geosci. Remote Sens. **49**(11), 4282–4297 (2011)
38. Ronneberger, O., Fischer, P., Brox, T.: U-net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
39. Scardapane, S., Comminiello, D., Hussain, A., Uncini, A.: Group sparse regularization for deep neural networks. Neurocomputing **241**, 81–89 (2017)
40. Shor, N.Z.: Minimization methods for non-differentiable functions, vol. 3. Springer Science & Business Media (2012)
41. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
42. Sutskever, I., Martens, J., Dahl, G., Hinton, G.: On the importance of initialization and momentum in deep learning. In: International Conference on Machine Learning. pp. 1139–1147 (2013)
43. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2818–2826 (2016)
44. Wen, W., Wu, C., Wang, Y., Chen, Y., Li, H.: Learning structured sparsity in deep neural networks. In: Advances in Neural Information Processing Systems. pp. 2074–2082 (2016)
45. Wen, W., Xu, C., Wu, C., Wang, Y., Chen, Y., Li, H.: Coordinating filters for faster deep neural networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 658–666 (2017)
46. Xu, Y., et al.: Trained rank pruning for efficient deep neural networks. arXiv preprint arXiv:1812.02402 (2018)
47. Xu, Y., et al.: Trp: Trained rank pruning for efficient deep neural networks. arXiv preprint arXiv:2004.14566 (2020)
48. Xu, Z., Chang, X., Xu, F., Zhang, H.: $\ell_{1/2}$ regularization: a thresholding representation theory and a fast solver. IEEE Trans. Neural Netw. Learn. Syst. **23**(7), 1013–1027 (2012)
49. Xu, Z., Guo, H., Wang, Y., Hai, Z.: Representative of $L_{1/2}$ regularization among $L_q(0 \leq q \leq 1)$ regularizations: an experimental study based on phase diagram. Acta Automatica Sinica **38**(7), 1225–1228 (2012)
50. Xu, Z., Zhang, H., Wang, Y., Chang, X., Liang, Y.: $L_{1/2}$ regularization. Sci. China Inf. Sci. **53**(6), 1159–1169 (2010)
51. Xue, F., Xin, J.: Learning sparse neural networks via ℓ_0 and tℓ_1 by a relaxed variable splitting method with application to multi-scale curve classification. In: Le Thi, H.A., Le, H.M., Pham Dinh, T. (eds.) WCGO 2019. AISC, vol. 991, pp. 800–809. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-21803-4_80
52. Yin, P., Lou, Y., He, Q., Xin, J.: Minimization of ℓ_{1-2} for compressed sensing. SIAM J. Sci. Comput. **37**(1), A536–A563 (2015)
53. Yin, P., Zhang, S., Lyu, J., Osher, S., Qi, Y., Xin, J.: Binaryrelax: a relaxation approach for training deep neural networks with quantized weights. SIAM J. Imag. Sci. **11**(4), 2205–2223 (2018)
54. Yin, W., Osher, S., Goldfarb, D., Darbon, J.: Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing. SIAM J. Imag. Sci. **1**(1), 143–168 (2008)
55. Yuan, M., Lin, Y.: Model selection and estimation in regression with grouped variables. J. Royal Stat. Soc.: Series B (Stat. Methodol.) **68**(1), 49–67 (2006)
56. Zhang, S., Xin, J.: Minimization of transformed l_1 penalty: closed form representation and iterative thresholding algorithms. Commun. Math. Sci. **15**(2), 511–537 (2017)

57. Zhang, S., Xin, J.: Minimization of transformed l_1 penalty: theory, difference of convex function algorithm, and robust application in compressed sensing. Math. Program. **169**(1), 307–336 (2018)
58. Zhang, S., Yin, P., Xin, J.: Transformed Schatten-1 iterative thresholding algorithms for low rank matrix completion. Commun. Math. Sci. **15**(3), 839–862 (2017)
59. Zhu, C., Han, S., Mao, H., Dally, W.J.: Trained ternary quantization. arXiv preprint arXiv:1612.01064 (2016)