IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 31, 2021, accepted August 12, 2021, date of publication August 16, 2021, date of current version August 24, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3105366

Improving Network Slimming With Nonconvex Regularization

KEVIN BUI[1], FREDRICK PARK[2], SHUAI ZHANG[1], YINGYONG QI[1], AND JACK XIN[1]
[1]Department of Mathematics, University of California at Irvine, Irvine, CA 92697, USA
[2]Department of Mathematics and Computer Science, Whittier College, Whittier, CA 90602, USA

Corresponding author: Kevin Bui (kevinb3@uci.edu)

This work was supported in part by NSF under Grant DMS-1854434 and Grant DMS-1952644, and in part by Qualcomm Faculty Award.

ABSTRACT Convolutional neural networks (CNNs) have developed to become powerful models for various computer vision tasks ranging from object detection to semantic segmentation. However, most of the state-of-the-art CNNs cannot be deployed directly on edge devices such as smartphones and drones, which need low latency under limited power and memory bandwidth. One popular, straightforward approach to compressing CNNs is network slimming, which imposes ℓ_1 regularization on the channel-associated scaling factors via the batch normalization layers during training. Network slimming thereby identifies insignificant channels that can be pruned for inference. In this paper, we propose replacing the ℓ_1 penalty with an alternative nonconvex, sparsity-inducing penalty in order to yield a more compressed and/or accurate CNN architecture. We investigate $\ell_p (0 < p < 1)$, transformed ℓ_1 (Tℓ_1), minimax concave penalty (MCP), and smoothly clipped absolute deviation (SCAD) due to their recent successes and popularity in solving sparse optimization problems, such as compressed sensing and variable selection. We demonstrate the effectiveness of network slimming with nonconvex penalties on three neural network architectures – VGG-19, DenseNet-40, and ResNet-164 – on standard image classification datasets. Based on the numerical experiments, Tℓ_1 preserves model accuracy against channel pruning, $\ell_{1/2,3/4}$ yield better compressed models with similar accuracies after retraining as ℓ_1, and MCP and SCAD provide more accurate models after retraining with similar compression as ℓ_1. Network slimming with Tℓ_1 regularization also outperforms the latest Bayesian modification of network slimming in compressing a CNN architecture in terms of memory storage while preserving its model accuracy after channel pruning.

INDEX TERMS Convolutional neural networks (CNN), machine learning, deep learning, network pruning, nonconvex optimization.

I. INTRODUCTION

In the past years, convolutional neural networks (CNNs) have evolved into superior models for various computer vision tasks, such as image classification [1]–[3], image segmentation [4]–[6], and object detection [7]–[9]. Unfortunately, training a highly accurate CNN is computationally demanding. State-of-the-art CNNs such as ResNet [1] 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 suggest several different directions to compress large CNNs or to learn smaller, more efficient models from scratch. Low-rank approximation [10]–[14] minimizes network redundancy by approximating the network's weight matrices with low-rank matrices. Weight quantization [15]–[19] replaces the floating-point weights with quantized weights, such as binary weights $\{-1, +1\}$ and ternary weights $\{-1, 0, +1\}$. Pruning [20]–[23] determines which weights, filters, and/or channels are unnecessary and removes them from the network. Lastly, another popular direction is to sparsify the CNN while training it [24]–[27]. Sparsity can be imposed on various types of structures existing in CNNs, such as filters and channels [27].

One interesting yet straightforward approach in sparsifying CNNs is *network slimming* [28]. This method imposes

The associate editor coordinating the review of this manuscript and approving it for publication was Charith Abhayaratne.

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. Since its inception, network slimming helps develop lightweight CNNs for various image classification tasks, such as traffic sign classification [29], facial expression recognition [30], and semantic segmentation. [31].

To improve the performance of network slimming, we propose replacing ℓ_1 regularization with an alternative regularization that promotes better sparsity and/or accuracy. Typically, better sparsity-promoting regularizers are nonconvex. Hence, we examine the ℓ_p penalty [32]–[34], transformed ℓ_1 (Tℓ_1) penalty [35], [36], the minimax concave penalty (MCP) [37], and the smoothly clipped absolute deviation (SCAD) penalty [38] due to their recent successes and popularity. These four regularizers have explicit formulas for their subgradients, which allow us to directly perform subgradient descent [39] when training CNNs.

Preliminary work in the conference version [40] of this paper demonstrated that Tℓ_1 regularization preserves the CNN's accuracy after pruning, and ℓ_p regularization yields a more compressed CNN than ℓ_1 with similar accuracy after retraining. This extended work includes discussion on the application of MCP and SCAD as additional regularization options for network slimming. Moreover, we provide more numerical results and analyses to validate the improvement in network slimming by using nonconvex regularization.

II. RELATED WORKS

A. COMPRESSION TECHNIQUES FOR CNN

1) LOW-RANK DECOMPOSITION

Low-rank decomposition aims to reduce weight matrices to their low-rank structures for faster computation and more efficient storage. One set of methods focuses on decomposing pre-trained weight tensors. Denton *et al*. [10] compressed the weight tensors of convolutional layers using singular value decomposition to approximate them. Jaderberg *et al*. [11] 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. On the other hand, there are methods that train CNNs with low-rank weight matrices from scratch. Tai *et al*. [41] incorporated low-rank tensor decomposition into their CNN training algorithm. Wen *et al*. [12] proposed *force regularization* to train a CNN towards having a low-rank representation. Xu *et al*. [13], [14] developed trained rank pruning, an optimization scheme that incorporates low-rank decomposition into the training process. Trained rank pruning was further strengthened by nuclear norm regularization.

2) WEIGHT QUANTIZATION

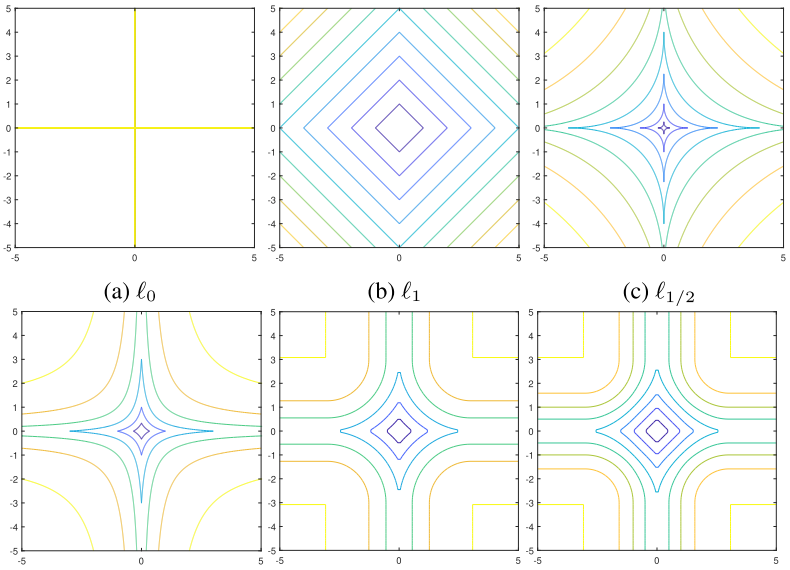
Quantization aims to represent weights with low-precision values ($\leq$8 bits arithmetic). The simplest form of quantization is binarization, constraining weights to only two values. Courbariaux *et al*. [16] proposed BinaryConnect, a method that trains deep neural networks (DNNs) with strictly binary weights. Neural networks with ternary weights have also been developed and investigated. Li *et al*. [17] created ternary weight networks, where the weights are only $-1, 0$, or $+1$. Zhu *et al*. [18] 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*. [19] developed BinaryRelax, which relaxes the quantization constraint into a continuous regularizer for the optimization problem needed to be solved in CNNs. Later, Bai *et al*. [42] proposed Proxquant, a stochastic proximal gradient method for quantizing networks while training them.

3) PRUNING

Pruning methods identify which weights, filters, and/or channels in CNNs are redundant and remove them from the networks. Early works focus on pruning weights. Han *et al*. [21] proposed a three-step framework to first train a CNN, prune weights if their norms are below a fixed threshold, and retrain the compressed CNN. Aghasi *et al*. [20], [43] proposed using convex optimization to determine which weights to prune while preserving model accuracy. For CNNs, channel or filter pruning is preferred over individual weight pruning since the former significantly eliminates more unnecessary weights. Li *et al*. [22] 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*. [23] proposed a metric that measures the redundancies in channels to determine which to prune. Network slimming [28] is also another method of channel pruning since it prunes channels with the smallest associated scaling factors. Zhao *et al*. [44] improved network slimming by incorporating a variational Bayesian framework.

4) SPARSE OPTIMIZATION

Sparse optimization methods introduce a sparse regularizer term to the loss function of the CNN so that the CNN is trained to have a compressed structure from scratch. BinaryRelax [19] and network slimming [28] are examples of sparse optimization methods for CNNs. Alvarez and Salzmann [24] and Scardapane *et al*. [26] applied group lasso [45] and sparse group lasso [26] to CNNs to obtain group-sparse networks. Nonconvex regularizers have also been examined recently. Xue and Xin [46] used ℓ_0 and Tℓ_1 regularization in three-layer CNNs that classify shaky vs. normal handwriting. Both Ma *et al*. [47] and Pandit *et al*. [48] proposed a regularizer that combines group sparsity and Tℓ_1 and applied it to CNNs for



(a) ℓ_0 (b) ℓ_1 (c) $\ell_{1/2}$

(d) T$\ell_1, a = 1$ (e) MCP, $\lambda = 1, a = 3$ (f) SCAD, $\lambda = 1, a = 3$

FIGURE 1. Contour plots of sparse regularizers.

image classification. Bui *et al*. [49] generalized sparse group lasso to incorporate nonconvex regularizers and applied it to various CNN architectures. Li *et al*. [50] introduced sparsity-inducing matrices into CNNs and imposed group sparsity on the rows or columns via ℓ_1 or other nonconvex regularizers to prune filters and/or channels.

B. REGULARIZATION PENALTY

Let $z = (z_1, \ldots, z_n) \in \mathbb{R}^n$. The ℓ_1 penalty is described by

$$\|z\|_1 = \sum_{i=1}^{n} |z_i|, \tag{1}$$

while the ℓ_0 penalty is described by

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

Although ℓ_1 regularization is popular in sparse optimization in various applications such as compressed sensing [51]–[53] and compressive imaging [54], [55], it may not actually yield the sparsest solution [32], [34], [36], [56], [57]. Moreover, it is sensitive to outliers and it may yield biased solutions [38].

A nonconvex alternative to the ℓ_1 penalty is the ℓ_p penalty

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

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 recovers sparser solution than ℓ_1 for certain compressed sensing problems [33], [58]. Empirical studies [33], [59] demonstrate 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. Moreover, it is used in image deconvolution [60], [61], hyperspectral unmixing [62], computed topography reconstruction [63], and image segmentation [64], [65]. Numerically, in compressed sensing, a small value ϵ is added to z_i to avoid blowup in the subgradient when $z_i = 0$. 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.

Although ℓ_p may yield sparser solutions than ℓ_1, it is still biased because parameters with large weights could be overpenalized [66]. Hence, a better regularizer should also be unbiased. In fact, Fan and Li [38] suggested three properties that a regularizer should have: (1) continuity to avoid model instability; (2) sparsity to reduce model complexity; and (3) unbiasedness to avoid modeling bias due to overpenalization of large parameters. Hence, we consider regularizers that have all three properties, such as Tℓ_1, MCP, and SCAD.

The Tℓ_1 penalty is formulated as

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

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$\ell_1 \to \ell_1$. It was validated to have the three aforementioned properties [67]. The Tℓ_1 penalty outperforms ℓ_1 and ℓ_p in compressed sensing problems with both coherent and incoherent sensing matrices [35], [36]. Additionally, the Tℓ_1 penalty yields satisfactory, sparse solutions in matrix completion [68] and deep learning [47].

The MCP penalty [37] is provided by

$$p_{\lambda,a}(z) = \sum_{i=1}^{n}\left[\left(\lambda|z_i| - \frac{z_i^2}{2a}\right)\mathbb{1}_{\{|z_i|\le a\lambda\}} + \frac{a\lambda^2}{2}\mathbb{1}_{\{|z_i|>a\lambda\}}\right], \quad (5)$$

where $\lambda \ge 0$ and $a > 1$. The parameter λ acts as a regularization parameter while the parameter a controls the level of sparsity, where the smaller a is, the sparser the solution becomes. In fact, a allows MCP to roughly interpolate between ℓ_0 and ℓ_1. Originally, MCP is developed for variable selection [37], but it has been utilized in various other applications such as image restoration [69] and matrix completion [70].

Lastly, the SCAD penalty [38] is given by

$$\tilde{p}_{\lambda,a}(z) = \sum_{i=1}^{n}\left[\lambda|z_i|\mathbb{1}_{\{|z_i|\le\lambda\}} + \frac{2\,a\lambda|z_i| - z_i^2 - \lambda^2}{2(a-1)}\mathbb{1}_{\{\lambda<|z_i|\le a\lambda\}} + \frac{\lambda^2(a+1)}{2}\mathbb{1}_{\{|z_i|>a\lambda\}}\right], \quad (6)$$

where $\lambda \ge 0$ is the regularization parameter and $a > 2$ controls the level of sparsity similarly to MCP. In both linear and logistic regression problems, SCAD outperforms ℓ_1 in variable selection [38]. Beyond variable selection, it is applied in compressed sensing [71], bioinformatics [72], [73], image processing [74], and wavelet approximation [75].

Figure 1 displays the contour plots of the aforementioned regularizers. With ℓ_1 regularization, the solution tends towards one of the corners of the rotated squares, making it sparse. Compared with ℓ_1, the level lines of the nonconvex regularizers bend more inward towards the axes, encouraging the solutions to coincide with one of the corners. In addition, the contour plots of the nonconvex regularizers appear more similar to the contour plot of ℓ_0. Therefore, solutions tend to be sparser with nonconvex regularization than with ℓ_1 regularization.

For further discussion on the aforementioned nonconvex regularizers, [76] and [77] provide detailed survey, application, and analysis.

Throughout the rest of the paper, we define $\lambda p_{1,a}(\cdot) := p_{\lambda,a}(\cdot)$ and $\lambda\tilde{p}_{1,a}(\cdot) := \tilde{p}_{\lambda,a}(\cdot)$.

III. PROPOSED METHOD

A. BATCH NORMALIZATION LAYER

Batch normalization [78] has been instrumental in speeding the convergence and improving generalization of many deep learning models, especially CNNs [1], [79]. 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 channel. Afterward, a linear transformation is applied to compensate for the loss of their representative abilities.

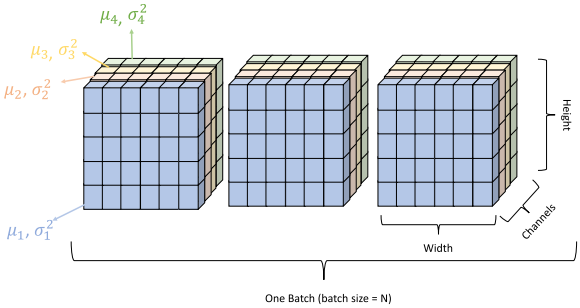


FIGURE 2. Visualization of batch normalization on a feature map. The mean and variance of the values of the pixels of the same colors corresponding to the channels are computed and are used to normalize these pixels.

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. Each entry of x' is denoted by x'_i, where $i = (i_N, i_C, i_H, i_W)$ indexes the features in (N, C, H, W) order. Here, N is the batch axis, C is the image channel axis, H is the image height axis, and W is the image width axis. We define the index set $S_i = \{k = (k_N, k_C, k_H, k_W) : k_C = i_C\}$, where k_C and i_C are the respective subindices of k and i along the C axis. In other words, the index set consists of pixels that belong to the same channel. The mean μ_i and variance σ_i^2 are computed as follows:

$$\mu_i = \frac{1}{|S_i|}\sum_{k\in S_i} x'_k, \quad \sigma_i^2 = \frac{1}{|S_i|}\sum_{k\in S_i}(x'_k - \mu_i)^2 + \epsilon \quad (7)$$

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

$$z_i = \gamma_{i_C}\hat{x}_i + \beta_{i_C}, \quad (8)$$

where $\gamma_{i_C}, \beta_{i_C} \in \mathbb{R}$ are trainable parameters. Additionally, γ_{i_C} is defined to be the scaling factor related to the channel i_C.

B. NETWORK SLIMMING WITH NONCONVEX SPARSE REGULARIZATION

Since the scaling factors γ_{i_C}'s in (8) are associated with the channels of a convolutional layer, we aim to penalize them with a sparse regularizer in order to identify 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 two sets of vectors $\{\gamma_l\}_{l=1}^L$ and $\{\beta_l\}_{l=1}^L$, where $\gamma_l = (\gamma_{l,1}, \ldots, \gamma_{l,C_l})$ and $\beta_l = (\beta_{l,1}, \ldots, \beta_{l,C_l})$ with C_l being the number of channels in the lth convolutional layer. Let $\mathcal{W}$ be the weight parameters that include $\{\gamma_l\}_{l=1}^L$ and $\{\beta_l\}_{l=1}^L$.

TABLE 1. Sparse regularizers and their (limiting) subgradients.

Name	$\mathcal{R}(z)$	$\partial R(z)$
ℓ_1	$\|z\|_1 = \sum_{i=1}^{n} \|z_i\|$	$\partial\|z\|_1 = \left\{\zeta \in \mathbb{R}^n : \zeta_i = \begin{cases} \operatorname{sgn}(z_i) & \text{if } z_i \neq 0 \\ \zeta_i \in [-1,1] & \text{if } z_i = 0 \end{cases}\right\}$
ℓ_p	$\|z\|_p^p = \sum_{i=1}^{n} \|z_i\|^p$	$\partial\|z\|_p^p = \left\{\zeta \in \mathbb{R}^n : \zeta_i = \begin{cases} \frac{p \cdot \operatorname{sgn}(z_i)}{\|z_i\|^{1-p}} & \text{if } z_i \neq 0 \\ \zeta_i \in \mathbb{R} & \text{if } z_i = 0 \end{cases}\right\}$
Tℓ_1	$P_a(z) = \sum_{i=1}^{n} \frac{(a+1)\|z_i\|}{a+\|z_i\|}$	$\partial P_a(z) = \left\{\zeta \in \mathbb{R}^n : \zeta_i = \begin{cases} \frac{a(a+1)\operatorname{sgn}(z_i)}{(a+\|z_i\|)^2} & \text{if } z_i \neq 0 \\ \zeta_i \in \left[-\frac{a+1}{a}, \frac{a+1}{a}\right] & \text{if } z_i = 0 \end{cases}\right\}$
MCP	$p_{\lambda,a}(z) = \sum_{i=1}^{n} \left[\left(\lambda\|z_i\| - \frac{z_i^2}{2a}\right)\mathbb{1}_{\{\|z_i\| \leq a\lambda\}} + \frac{a\lambda^2}{2}\mathbb{1}_{\{\|z_i\| > a\lambda\}}\right]$	$\partial p_{\lambda,a}(z) = \left\{\zeta \in \mathbb{R}^n : \zeta_i = \begin{cases} 0 & \text{if } \|z_i\| > a\lambda \\ \lambda\operatorname{sgn}(z_i) - \frac{z_i}{a} & \text{if } 0 < \|z_i\| \leq a\lambda \\ \zeta_i \in [-\lambda, \lambda] & \text{if } z_i = 0 \end{cases}\right\}$
SCAD	$\tilde{p}_{\lambda,a}(z) = \sum_{i=1}^{n} \left[\lambda\|z_i\|\mathbb{1}_{\{\|z_i\| \leq \lambda\}} + \frac{2a\lambda\|z_i\| - z_i^2 - \lambda^2}{2(a-1)}\mathbb{1}_{\{\lambda < \|z_i\| \leq a\lambda\}} + \frac{\lambda^2(a+1)}{2}\mathbb{1}_{\{\|z_i\| > a\lambda\}}\right]$	$\partial\tilde{p}_{\lambda,a}(z) = \left\{\zeta \in \mathbb{R}^n : \zeta_i = \begin{cases} 0 & \text{if } \|z_i\| > a\lambda \\ \frac{a\lambda\operatorname{sgn}(z_i) - z_i}{a-1} & \text{if } \lambda < \|z_i\| \leq a\lambda \\ \lambda\operatorname{sgn}(z_i) & \text{if } 0 < \|z_i\| \leq \lambda \\ \zeta_i \in [-\lambda, \lambda] & \text{if } z_i = 0 \end{cases}\right\}$

Hence, the trainable parameters $\mathcal{W}$ of the CNN are learned by minimizing the following objective function:

$$\frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(h(x_i, \mathcal{W}), y_i) + \lambda \sum_{l=1}^{L} \mathcal{R}(\gamma_l), \tag{9}$$

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 and it could potentially be biased, we investigate the method with a nonconvex regularizer, where $\mathcal{R}(\cdot)$ is $\|\cdot\|_p^p$, $P_a(\cdot)$, $p_{1,a}(\cdot)$, or $\tilde{p}_{1,a}(\cdot)$.

To minimize (9), stochastic gradient descent is applied to the loss function term while subgradient descent is applied to the regularizer term [39]. The algorithm is summarized in Algorithm 1. Subgradient descent is applicable to the nonconvex regularizers $\mathcal{R}(z)$ for $z \in \mathbb{R}^n$ as it is for ℓ_1. Like ℓ_1, the nonconvex regularizers are of the form $\sum_{i=1}^{n} r(z_i)$, where $r : \mathbb{R} \to \mathbb{R}$ has the following properties:

(i) $r(0) = 0$;
(ii) r is an even, proper, and continuous function;
(iii) r is increasing on $[0, +\infty)$;
(iv) r is differentiable on $(-\infty, 0) \cup (0, +\infty)$.

These properties ensure that r is differentiable everywhere except at 0 and 0 is the global minimum of r while being its only local minimum. As a result, the regularizers are differentiable when $z_i \neq 0$ for all $i = 1, \ldots, n$. Hence, subgradient descent becomes gradient descent at these points. If $z_i = 0$ for at least one index i, then we need to compute its (limiting) subgradient [80, Definition 6.1] and decide a candidate descent direction. Fortunately, because $\mathcal{R}(z) = \sum_{i=1}^{n} r(z_i)$, we have

$$\partial\mathcal{R}(z) = (\partial r(z_1), \partial r(z_2), \ldots, \partial r(z_n))$$

Algorithm 1 Algorithm for Minimizing (9)

Input: Regularization parameter λ, learning rate η, sparse regularizer $\mathcal{R}$
Initialize $\mathcal{W}^0$, excluding $\{\gamma_l\}_{l=1}^{L}$, with random values.
Initialize $\{\gamma_l^0\}_{l=1}^{L}$ with entries 0.5.
1: **for** each epoch $t = 1, \ldots, T$ **do**
2: $\mathcal{W}^t = \mathcal{W}^{t-1} - \frac{\eta}{N}\sum_{i=1}^{N} \nabla\mathcal{L}(h(x_i, \mathcal{W}^{t-1}), y_i)$ by stochastic gradient descent or variant.
3: $\gamma_l^t = \gamma_l^{t-1} - \eta\lambda\partial\mathcal{R}(\gamma_l^{t-1})$ for $l = 1, \ldots, L$.
4: **end for**

by [80, Proposition 6.17(e)]. This means that at each component $r(z_i)$, we can compute its subgradient $\partial r(z_i)$ individually and select a descent direction from the set. Since 0 is a local minimum of r, we have $0 \in \partial r(0)$, so we can select 0 as a descent direction for simplicity. Table 1 presents the subgradients of the regularizers.

After the CNN is trained with (9) using Algorithm 1, we prune the channels whose scaling factors are small in magnitude, giving us a compressed model. However, the compressed model may lose its original accuracy, so it may need to be retrained but without the sparse regularizer in order to attain its original accuracy or better.

IV. EXPERIMENTAL RESULTS

We apply the proposed nonconvex network slimming using $\ell_p (0 < p < 1)$, Tℓ_1, MCP, and SCAD regularization on various networks and datasets and compare their results against the original network slimming with ℓ_1 regularization as the baseline.

Code for the experiments is available at https://github.com/kbui1993/NonconvexNetworkSlimming.

TABLE 2. Effect of channel pruning on the mean pruned parameter / FLOPs percentages (%) on VGG-19 trained on (a) CIFAR 10, (b) CIFAR 100, and (c) SVHN. The mean is computed from five runs for each regularizer. For each channel pruning ratio, **bold** indicates outperforming ℓ_1; * indicates best value; and NA indicates at least one of the five models is over-pruned.

(a)	CIFAR 10								
Channel Pruning Ratio	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90
ℓ_1	21.10% / 11.83%	39.34% / 22.18%	54.88% / 31.09%	67.56% / 38.51%	77.53% / 44.78%	84.71% / 49.68%	88.81% / 51.95%	NA	NA
$\ell_{3/4}$	**21.14% / 12.11%**	**39.55% / 22.71%**	**55.13% / 32.06%**	**67.96% / 39.97%**	**77.99% / 46.08%**	**85.24% / 51.37%**	**89.69% / 54.96%**	NA	NA
$\ell_{1/2}$	**21.12% / 12.38%**	**39.62% / 22.65%**	**55.29%* / 32.06%**	**68.09%* / 39.75%**	**78.10% / 46.45%**	**85.43% / 52.02%**	**90.01% / 56.12%**	**93.66% / 65.52%**	NA
$\ell_{1/4}$	19.95% / **15.49%***	37.66% / **29.82%***	52.99% / **42.93%***	66.20% / **54.39%***	77.07% / **64.47%***	**85.76% / 73.44%***	**92.14%* / 81.89%***	**96.54%* / 91.07%***	**99.05%* / 98.32%***
T$\ell_1(a = 10.0)$	**21.15% / 11.92%**	**39.41% / 22.59%**	**54.95% / 31.85%**	**67.71% / 39.39%**	**77.69% / 45.77%**	**84.89% / 50.47%**	**89.06% / 52.75%**	NA	NA
T$\ell_1(a = 1.0)$	**21.16% / 12.12%**	**39.35% / 23.13%**	**54.94% / 32.60%**	**67.88% / 40.69%**	**78.06% / 47.94%**	**85.55% / 53.40%**	**90.34% / 57.43%**	NA	NA
T$\ell_1(a = 0.5)$	20.94% / **12.59%**	39.29% / **23.66%**	**54.92% / 33.61%**	**67.83% / 42.34%**	**78.22%* / 49.58%**	**85.92%* / 55.36%**	**90.88% / 59.84%**	NA	NA
MCP($a = 15000$)	**21.18%** / 11.56%	**39.48%** / 21.43%	**54.96%** / 30.31%	**67.62%** / 37.75%	77.53% / 43.76%	84.58% / 48.05%	88.58% / 50.69%	NA	NA
MCP($a = 10000$)	20.99% / 11.24%	**39.35%** / 21.23%	**54.97%** / 29.94%	**67.64%** / 37.71%	**77.55%** / 43.42%	84.61% / 47.88%	88.63% / 50.49%	NA	NA
MCP($a = 5000$)	**21.20%** / 10.97%	**39.71%*** / 20.92%	**55.24%** / 29.31%	**67.87%** / 36.32%	**77.61%** / 41.97%	84.53% / 46.08%	88.56% / 49.49%	NA	NA
SCAD($a = 15000$)	21.10% / 11.70%	**39.42%** / 21.83%	**54.97%** / 30.75%	**67.68%** / 37.83%	**77.56%** / 43.62%	84.66% / 48.09%	88.71% / 50.70%	NA	NA
SCAD($a = 10000$)	**21.21%** / 11.23%	**39.45%** / 20.95%	**54.95%** / 29.89%	**67.60%** / 37.33%	77.44% / 43.23%	84.53% / 47.52%	88.57% / 50.43%	NA	NA
SCAD($a = 5000$)	**21.24%*** / 11.25%	**39.65%** / 21.08%	**55.16%** / 29.64%	**67.77%** / 36.77%	**77.58%** / 42.69%	84.58% / 46.96%	88.62% / 50.19%	NA	NA

(b)	CIFAR 100								
Channel Pruning Ratio	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90
ℓ_1	21.91% / 12.44%	40.36% / 22.89%*	55.83% / 28.19%	67.45% / 31.75%	75.35% / 36.08%	NA	NA	NA	NA
$\ell_{3/4}$	**21.98%** / 10.92%	**40.64%** / 20.75%	**56.04% / 28.64%**	**68.01% / 34.77%**	**76.54% / 38.40%**	NA	NA	NA	NA
$\ell_{1/2}$	**22.02%** / 10.89%	**40.85%*** / 20.04%	**56.29%*** / 27.83%	**68.41% / 34.23%**	**77.19% / 39.40%**	**83.07% / 43.82%**	NA	NA	NA
$\ell_{1/4}$	**22.00%** / 10.80%	**40.71%** / 20.52%	**56.21% / 29.10%**	**68.53%* / 36.59%**	**78.16%* / 44.28%***	**85.71%* / 54.15%***	**91.48%* / 68.94%***	**96.54%* / 86.86%***	NA
T$\ell_1(a = 10.0)$	21.83% / 12.41%	40.34% / 22.62%	55.62% / **29.71%**	**67.80% / 32.97%**	**76.07% / 37.00%**	NA	NA	NA	NA
T$\ell_1(a = 1.0)$	21.87% / 11.21%	**40.47%** / 20.85%	**55.99% / 29.04%**	**68.22% / 36.22%**	**77.18% / 40.47%**	**82.90% / 43.94%**	NA	NA	NA
T$\ell_1(a = 0.5)$	21.67% / 11.42%	40.33% / 21.52%	**55.97% / 30.16%***	**68.49% / 37.52%***	**77.99% / 43.24%**	**84.09% / 47.15%**	NA	NA	NA
MCP($a = 15000$)	21.86% / 12.37%	40.28% / 22.40%	55.67% / 28.17%	67.29% / 31.62%	**75.38%** / 35.91%	NA	NA	NA	NA
MCP($a = 10000$)	21.90% / 12.40%	40.24% / 22.60%	55.73% / 27.94%	67.28% / 31.60%	75.20% / 35.95%	NA	NA	NA	NA
MCP($a = 5000$)	**22.03%*** / 11.90%	**40.49%** / 21.45%	**55.94%** / 26.46%	67.35% / 30.43%	75.03% / 34.78%	NA	NA	NA	NA
SCAD($a = 15000$)	**21.96% / 12.48%***	**40.42%** / 22.36%	55.83% / 28.04%	**67.50%** / 31.70%	75.34% / 35.91%	NA	NA	NA	NA
SCAD($a = 10000$)	21.90% / 11.76%	40.28% / 21.82%	55.71% / 27.34%	67.18% / 31.07%	75.02% / 35.56%	NA	NA	NA	NA
SCAD($a = 5000$)	**22.01%** / 11.59%	**40.49%** / 20.60%	55.75% / 25.63%	66.91% / 29.91%	74.50% / 34.47%	NA	NA	NA	NA

(c)	SVHN								
Channel Pruning Ratio	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90
ℓ_1	19.92% / 15.90%	37.51% / 30.99%	52.91% / 43.85%	66.08% / 55.25%	76.98% / 65.06%	85.60% / 73.63%	92.00% / 80.80%	96.12% / 86.39%	NA
$\ell_{3/4}$	**19.96% / 16.04%**	**37.60%** / 30.91%	**52.96%** / 43.82%	66.00% / **55.74%**	76.84% / **65.93%**	85.52% / **74.49%**	92.00% / **81.53%**	**96.23% / 87.23%**	NA
$\ell_{1/2}$	19.80% / **16.80%**	37.35% / **31.70%**	52.74% / **44.89%**	65.84% / **56.70%**	76.79% / **67.00%**	85.52% / **75.75%**	**92.01% / 83.14%**	**96.31% / 88.36%**	**98.94% / 95.45%**
$\ell_{1/4}$	19.36% / **17.72%***	36.77% / **32.96%***	52.07% / **47.02%***	65.13% / **59.17%***	76.01% / **70.57%***	84.86% / **79.92%***	91.59% / **87.85%***	**96.36% / 93.72%***	**99.08%* / 98.15%***
T$\ell_1(a = 10.0)$	**19.94%** / 15.90%	37.39% / 30.91%	52.71% / **44.18%**	65.94% / **55.64%**	76.83% / **65.68%**	85.53% / **73.99%**	91.96% / **80.96%**	**96.19% / 86.70%**	**98.60% / 93.70%**
T$\ell_1(a = 1.0)$	19.71% / **17.01%**	37.17% / **32.34%**	52.55% / **45.74%**	65.70% / **57.43%**	76.70% / **67.21%**	85.44% / **76.19%**	**92.02%* / 83.11%**	**96.38% / 88.58%**	NA
T$\ell_1(a = 0.5)$	**19.99% / 16.20%**	**37.52% / 31.27%**	52.70% / **44.95%**	65.70% / **57.29%**	76.68% / **67.60%**	85.40% / **76.66%**	91.98% / **83.79%**	**96.43%* / 89.53%**	**98.73% / 94.41%**
MCP($a = 15000$)	**20.14%** / 15.43%	**37.86%** / 29.52%	**53.15%** / 42.46%	**66.23%** / 53.76%	**77.07%** / 63.40%	**85.67%** / 71.68%	91.92% / 78.98%	95.87% / 84.33%	**98.61% / 93.64%**
MCP($a = 10000$)	**20.13%** / 15.60%	**37.91%** / 29.62%	**53.41%** / 41.85%	**66.40%** / 52.96%	**77.20%** / 62.47%	**85.72%** / 70.84%	91.88% / 77.91%	95.68% / 83.63%	NA
MCP($a = 5000$)	**20.34%*** / 14.91%	**38.20%*** / 28.64%	**53.63%** / 40.96%	**66.72%** / 51.97%	**77.46%** / 61.03%	**85.76%** / 68.51%	91.68% / 75.01%	95.41% / 82.79%	NA
SCAD($a = 15000$)	19.92% / **16.25%**	**37.55%** / 30.27%	**53.11%** / 42.75%	**66.19%** / 54.23%	**77.06%** / 63.81%	85.57% / 72.47%	91.91% / 79.38%	95.88% / 84.81%	NA
SCAD($a = 10000$)	**19.97%** / 15.32%	**37.60%** / 29.41%	**53.22%** / 41.88%	**66.31%** / 52.80%	**77.15%** / 62.58%	**85.66%** / 70.73%	91.81% / 77.63%	95.65% / 83.85%	NA
SCAD($a = 5000$)	**20.28%** / 15.07%	**38.10%** / 28.72%	**53.66%*** / 40.75%	**66.82%*** / 51.56%	**77.50%*** / 61.09%	**85.79%*** / 69.09%	91.73% / 75.80%	95.47% / 83.12%	NA

A. DATASETS

1) CIFAR 10/100

The CIFAR 10/100 dataset [81] consists of 50k training color images and 10k test color images with 10/100 classes total. The resolution of each image is 32×32. To preprocess the dataset, we apply the data augmentation techniques (horizontal flipping and translation by 4 pixels) that have been standard in practice [1], [28], [82]–[84] followed by global contrast normalization and ZCA whitening [84]. These preprocessing techniques help improve the classification accuracy of CNNs on CIFAR 10 and 100 as demonstrated in [83], [84].

2) SVHN

The SVHN dataset [85] consists of 32×32 color images. The entire training set has 604,388 images and the test set has 26,032 images. Before training on the dataset, each image is normalized by the channel means and standard deviations.

We evaluate the proposed methods on VGG-19 [3], DenseNet-40 [86], and ResNet-164 [1], three networks that were examined in [28]. More specifically, we use a variation of VGG-19 from https://github.com/szagoruyko/cifar.torch, a 40-layer DenseNet with a growth rate of 12, and a 164-layer pre-activation ResNet with a bottleneck structure.

B. IMPLEMENTATION DETAILS

1) TRAINING THE NETWORK

To perform a fair comparison between the original network slimming and the proposed nonconvex network slimming, we emulate most of the training settings in the original work [28]. All networks are trained from scratch using stochastic gradient descent. The initial learning rate is set at 0.1, and it is reduced by a factor of 10 at the 50% and 75% of the total number of epochs. In addition, we use weight decay of 10^{-4} and Nesterov momentum [87] of 0.9 without dampening. On CIFAR 10/100, we train for 160 epochs, while on SVHN,

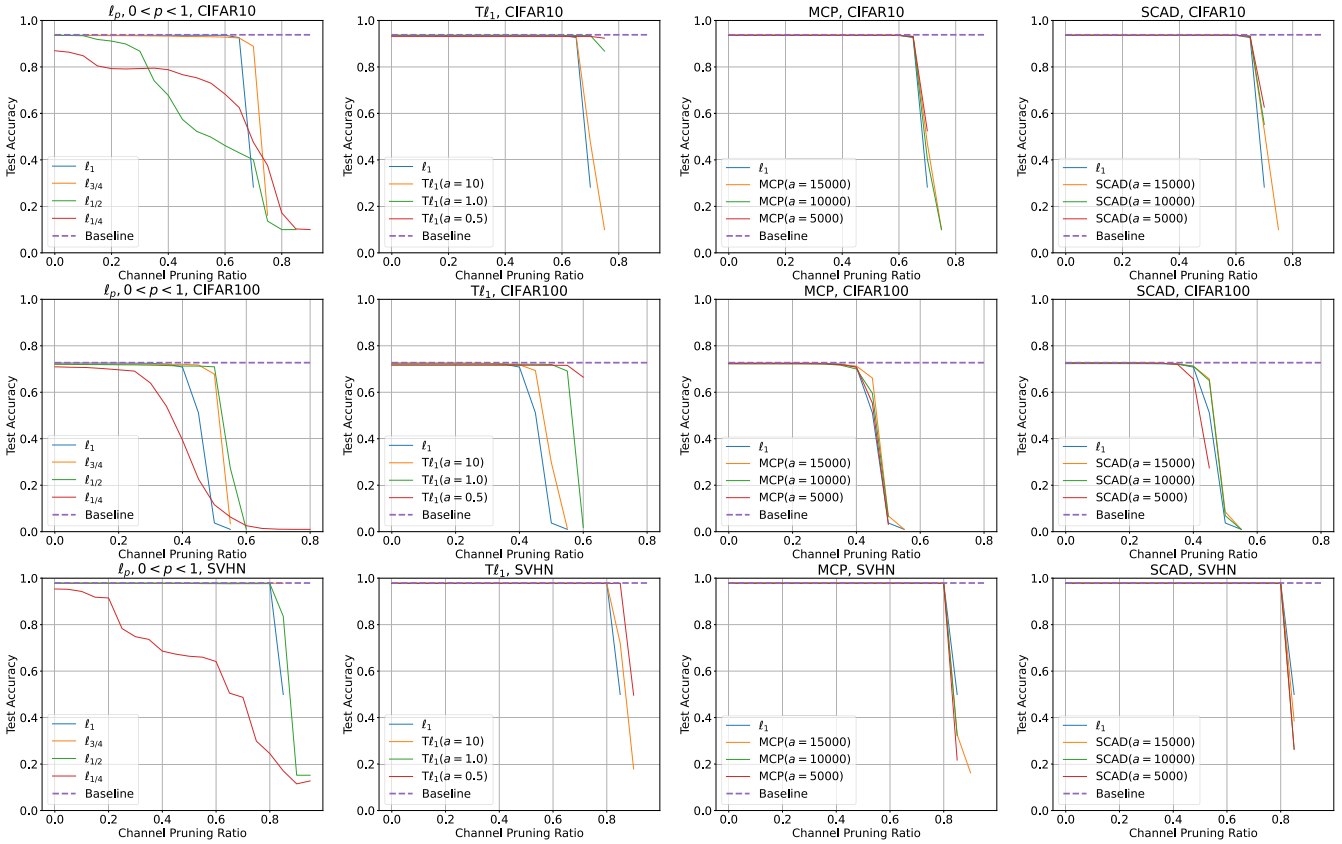


FIGURE 3. Effect of channel pruning on the mean test accuracy of five runs of VGG-19 on CIFAR 10/100 and SVHN. Baseline refers to the mean test accuracy of the unregularized model that is not pruned. Baseline accuracies are 93.83% for CIFAR 10, 72.73% for CIFAR 100, and 97.91% for SVHN.

we train for 20 epochs. On both datasets, the training batch size is 64. Weight initialization is based on [88] and scaling factor initialization is set to 0.5 as done in [28]. We examine the following regularizers for network slimming: ℓ_1, $\ell_p(p = 0.25, 0.5, 0.75)$, T$\ell_1(a = 0.5, 1.0, 10.0)$, MCP ($a = 5000, 10000, 15000$), and SCAD ($a = 5000, 10000, 15000$). The examined parameter values for these regularizers are chosen because they attain similar model accuracy as the baseline model without scaling factor regularization and they can prune a model by at least 40% of its channels. Lastly, we have the regularization parameter $\lambda = 10^{-4}$ for VGG-19 and DenseNet-40 and $\lambda = 5 \times 10^{-5}$ for ResNet-164. The regularization parameter is chosen by trying to balance between model accuracy and channel sparsity.

2) PRUNING THE NETWORK

After a model is trained, its channels are pruned globally. For example, we specify a channel pruning ratio to be 0.35 or a channel pruning percentage to be 35% and determine the 35th percentile among all magnitudes of the scaling factors of the model. The 35th percentile is set as the threshold. Any channels whose scaling factors are below the threshold in magnitude are pruned.

Since the channels are pruned globally, there is a threshold specific for each model: if the pruning ratio is above a certain value, a model becomes over-pruned. That is, the model cannot be used for inference because at least one of its layers has all of its channels removed.

3) RETRAINING THE NETWORK

We retrain the pruned model 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 compressed model's original accuracy prior to pruning.

4) PERFORMANCE METRICS

We compare the regularizers' performances based on test accuracy and compression of their respective models.

After pruning a network by its channels, we measure its compression by the remaining number of parameters and floating point operations (FLOPs). The number of parameters relates to the storage cost while the number of FLOPS relates to the computational cost. In our experiments, we report the following percentages:

$$\text{Percentage of parameters pruned} = \left(1 - \frac{\text{\# parameters remaining}}{\text{total \# network parameters}}\right) \times 100\%$$

and

$$\text{Percentage of FLOPs pruned} = \left(1 - \frac{\text{\# FLOPs remaining}}{\text{total \# network FLOPs}}\right) \times 100\%.$$

Since CNNs are highly nonconvex, each run of the same model and regularizer with the same hyperparameters will give a different result. Hence, we train each model of one regularizer five times and compute the mean. Therefore, the mean test accuracies and mean ratios/percentages of parameters/FLOPs pruned are computed from five runs each.

C. CHANNEL PRUNING RESULTS

1) VGG-19

VGG-19 has about 20 million parameters and 7.97×10^8 FLOPs. Table 2 shows the relationships between channel pruning ratios and mean percentages of parameters/FLOPs pruned. Figure 3 shows the effect of channel pruning on mean test accuracies.

On CIFAR 10, according to Table 2a, most of the nonconvex regularizers prune more parameters than ℓ_1 up to channel pruning ratio 0.50. Although more parameters are pruned, MCP and SCAD require more FLOPs in general compared to ℓ_1. On the other hand, ℓ_p and Tℓ_1 outperform ℓ_1 with respect to percentages of parameters/FLOPs pruned for channel pruning ratio at least 0.60. Additionally, the models trained with $\ell_{1/2}$ and $\ell_{3/4}$ can have at least 80% of its channels pruned and still be used for inference even though their test accuracies are low. However, their test accuracies can be improved if the models were retrained. According to Figure 3, $\ell_{3/4}$, Tℓ_1, MCP, and SCAD are more robust than ℓ_1 to channel pruning since their accuracies drop at higher channel pruning ratios. Although both $\ell_{1/2}$ and $\ell_{1/4}$ compress the model significantly compared to other regularizers, they are very sensitive to channel pruning.

On CIFAR 100, according to Table 2b, ℓ_p and T$\ell_1(a = 0.5, 1.0)$ require less parameters and FLOPs compared to ℓ_1 when the channel pruning ratios are at least 0.40. MCP and SCAD have comparable number of parameters and FLOPs pruned as ℓ_1. Figure 3 shows that Tℓ_1 is robust against channel pruning, especially when $a = 0.5$. At channel pruning ratio 0.6, the accuracy for T$\ell_1(a = 0.5)$ does not drop as much compared to other values of a and also other nonconvex regularizers. For the other regularizers, ℓ_1 is outperformed by $\ell_{3/4}$, $\ell_{1/2}$, MCP, and SCAD ($a = 10000, 15000$). Like for CIFAR 10, models trained with either $\ell_{1/2}$ or $\ell_{1/4}$ are still sensitive to channel pruning.

Lastly, for SVHN, according to Table 2c, MCP and SCAD generally outperform ℓ_1 in parameter pruning percentages for channel pruning ratios up to 0.60, but they do not save more on FLOPs. However, FLOPs are reduced more by ℓ_p and Tℓ_1 in general across all channel pruning ratios. By Figure 3, $\ell_{3/4}$, $\ell_{1/2}$, and Tℓ_1 have higher test accuracies than ℓ_1 when the channel pruning ratio is at 0.85.

In general, nonconvex regularizers save more on parameters, FLOPs, or both. It is important to note that Tℓ_1, especially $a = 0.5$, helps preserve model accuracy against channel pruning, and $\ell_{1/4}$ is very sensitive to channel pruning.

2) DENSENET-40

DenseNet-40 has about 1 million parameters and 5.33×10^8 FLOPs. Table 3 shows the relationships between channel pruning ratios and mean percentages of parameters/FLOPs pruned. Figure 4 shows the effect of channel pruning on mean test accuracies.

On CIFAR 10, by Table 3a, ℓ_p and Tℓ_1 compress the model more in terms of number of parameters and FLOPs than ℓ_1 after channel pruning across the various levels of channel pruning ratios. In general, MCP and SCAD require slightly more FLOPS than ℓ_1, but they require similar number of parameters as ℓ_1. According to Figure 4, $\ell_p(p = 1/2, 3/4)$ and Tℓ_1 are more robust to channel pruning than ℓ_1 since their accuracies drop at higher channel pruning ratios, while MCP and SCAD are worse.

For CIFAR 100, Table 3b demonstrates that ℓ_p and Tℓ_1 generally reduce more parameters and FLOPs required than ℓ_1 after channel pruning. At channel pruning ratios 0.60 and above, MCP and SCAD reduce only more FLOPs than ℓ_1. In addition, models with MCP and SCAD regularization remain usable for inference after 90% of their channels are pruned, unlike models with ℓ_1 regularization. However, their test accuracies are unacceptable so that the models will need to be retrained to recover its original accuracies. According to Figure 4, $\ell_p(p = 1/2, 3/4)$, Tℓ_1, and MCP ($a = 15000$) are more robust to channel pruning than ℓ_1 because their test accuracies drop at higher channel pruning ratios than ℓ_1's.

For SVHN, Table 3c shows that ℓ_p and Tℓ_1 have larger parameter/FLOPs pruning percentages than ℓ_1 across different levels of the channel pruning ratios. In general, MCP also saves more on parameters and FLOPs for channel pruning ratio up to 0.50. After 0.50, MCP saves more on only FLOPs. SCAD also generally saves more on FLOPs than ℓ_1. According to Figure 4, the test accuracy remains nearly constant for channel pruning ratio up to 0.90 for all regularizers except for $\ell_{1/4}$ and $\ell_{1/2}$. We also observe that across different channel pruning ratios, Tℓ_1 has slightly worse test accuracy than ℓ_1 while MCP and SCAD mostly have better test accuracies than ℓ_1.

In summary, we observe that ℓ_p and Tℓ_1 reduce more parameters and FLOPs required than ℓ_1 after channel pruning, while MCP and SCAD save more on only FLOPs specifically for CIFAR 100 and SVHN. Like for VGG-19, T$\ell_1(a = 0.5)$ is the most robust against channel pruning, whereas $\ell_{1/4}$ is the most sensitive to it.

3) RESNET-164

ResNet-164 has about 1.70 million parameters and requires 5.00×10^8 FLOPs. Table 4 records the mean percentages of parameters/FLOPs pruned for different channel pruning ratios. Figure 5 shows the effect of channel pruning on the test accuracies of the regularized models.

TABLE 3. Effect of channel pruning on the mean pruned parameter / FLOPs percentages (%) on DenseNet-40 trained on (a) CIFAR 10, (b) CIFAR 100, and (c) SVHN. The mean is computed from five runs for each regularizer. For each channel pruning ratio, **bold** indicates outperforming ℓ_1; * indicates best value; and NA indicates at least one of the five models is over-pruned.

(a)	CIFAR 10								
Channel Pruning Ratio	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90
ℓ_1	9.22% / 8.40%	18.35% / 16.63%	27.57% / 24.91%	36.73% / 33.02%	45.95% / 41.49%	55.15% / 49.75%	64.38% / 58.10%	73.75% / 68.18%	83.76% / 79.75%
$\ell_{3/4}$	**9.32% / 8.53%**	**18.64% / 16.79%**	**27.87% / 25.62%**	**37.14% / 34.10%**	**46.42% / 42.85%**	**55.62% / 51.27%**	**64.90% / 59.71%**	**74.25% / 68.63%**	**84.02% / 80.07%**
$\ell_{1/2}$	**9.33% / 8.65%**	**18.59% / 17.08%**	**27.97% / 25.96%**	**37.26% / 34.71%**	**46.62% / 43.33%**	**55.88% / 51.85%**	**65.12% / 60.32%**	**74.47% / 69.14%**	**84.36% / 80.13%**
$\ell_{1/4}$	**9.35%* / 8.83%***	**18.71% / 17.63%***	**28.13%* / 26.39%***	**37.52%* / 35.27%***	**47.05%* / 44.74%***	**56.69%* / 54.33%***	**66.56%* / 64.34%***	**77.02%* / 75.42%***	NA
T$\ell_1(a = 10.0)$	9.20% / 8.31%	18.34% / **16.83%**	**27.59% / 25.32%**	**36.82% / 33.65%**	**46.08% / 41.97%**	**55.27% / 50.17%**	**64.54% / 58.19%**	**73.89%** / 68.01%	**83.89%** / 79.72%
T$\ell_1(a = 1.0)$	**9.35%* / 8.67%**	**18.63% / 17.09%**	**27.85% / 25.39%**	**37.17% / 34.04%**	**46.41% / 42.32%**	**55.73% / 50.93%**	**65.14% / 59.70%**	**74.46% / 68.57%**	**84.23% / 80.19%**
T$\ell_1(a = 0.5)$	**9.35%* / 8.45%**	**18.72%* / 16.99%**	**28.08% / 25.82%**	**37.39% / 34.47%**	**46.73% / 43.13%**	**56.16% / 52.18%**	**65.49% / 60.60%**	**74.88% / 69.28%**	**84.45% / 80.70%**
MCP($a = 15000$)	9.19% / 8.01%	**18.37%** / 16.21%	**27.59%** / 24.47%	**36.79%** / 32.96%	**45.97%** / 40.97%	55.15% / 49.24%	64.35% / 57.64%	**73.77%** / 68.13%	83.72% / 79.37%
MCP($a = 10000$)	**9.29%** / 8.23%	**18.45%** / 16.28%	**27.71%** / 24.60%	**36.93% / 33.05%**	**46.07%** / 41.26%	**55.22%** / 49.32%	**64.40%** / 57.57%	**73.91% / 68.23%**	**83.85%** / 79.39%
MCP($a = 5000$)	9.17% / 8.19%	18.25% / 16.11%	27.45% / 24.25%	36.57% / 32.22%	45.75% / 40.39%	54.94% / 48.56%	64.13% / 56.70%	73.75% / 67.59%	**83.92%** / 79.17%
SCAD($a = 15000$)	9.21% / 8.11%	**18.36%** / 16.21%	27.54% / 24.43%	**36.75%** / 32.57%	45.94% / 40.90%	55.12% / 49.18%	**64.41%** / 57.68%	**73.85%** / 68.10%	**83.80%** / 79.42%
SCAD($a = 10000$)	9.18% / 8.16%	**18.36%** / 16.54%	**27.60%** / 24.83%	**36.77%** / 32.77%	45.94% / 41.05%	55.10% / 49.04%	64.30% / 57.21%	**73.79%** / 67.98%	83.75% / 79.27%
SCAD($a = 5000$)	9.06% / 7.78%	18.22% / 15.88%	27.40% / 23.97%	36.54% / 32.07%	45.66% / 39.87%	54.87% / 48.02%	64.08% / 56.01%	73.71% / 67.25%	**83.84%** / 78.76%

(b)	CIFAR 100								
Channel Pruning Ratio	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90
ℓ_1	9.18% / 7.46%	18.34% / 15.21%	27.53% / 22.91%	36.69% / 30.44%	45.84% / 37.84%	54.98% / 45.36%	64.12% / 54.09%	73.39% / 65.92%	
$\ell_{3/4}$	**9.19% / 8.20%**	**18.39% / 16.12%**	**27.57% / 24.04%**	**36.76% / 31.88%**	**45.95% / 39.91%**	**55.13% / 47.74%**	**64.33% / 55.84%**	**73.56% / 66.30%**	**83.34% / 79.89%**
$\ell_{1/2}$	**9.20% / 8.23%**	**18.41% / 16.41%**	**27.62% / 24.37%**	**36.85% / 32.34%**	**46.06% / 40.58%**	**55.26% / 48.91%**	**64.44% / 56.97%**	**73.67% / 66.98%**	NA
$\ell_{1/4}$	**9.26%* / 8.33%***	**18.53%* / 16.76%***	**27.85%* / 25.00%***	**37.17%* / 33.70%***	**46.51%* / 43.03%***	**55.94%* / 52.75%***	**65.73%* / 63.59%***	**76.28%* / 76.02%***	NA
T$\ell_1(a = 10.0)$	**9.19% / 7.80%**	**18.35%** / 15.19%	**27.55% / 23.01%**	**36.72% / 30.60%**	**45.92% / 38.42%**	**55.08% / 45.82%**	**64.24%** / 53.94%	**73.49%** / 65.90%	NA
T$\ell_1(a = 1.0)$	**9.26%* / 8.00%**	**18.46% / 15.92%**	**27.72% / 23.79%**	**36.91% / 31.49%**	**46.15% / 39.49%**	**55.35% / 47.34%**	**64.55% / 55.62%**	**73.78% / 66.24%**	**83.48% / 80.01%**
T$\ell_1(a = 0.5)$	**9.25% / 8.11%**	**18.49% / 15.98%**	**27.75% / 24.15%**	**36.98% / 32.22%**	**46.24% / 40.44%**	**55.46% / 48.33%**	**64.71% / 56.39%**	**73.92% / 66.20%**	**83.60%* / 80.15%***
MCP($a = 15000$)	**9.19% / 7.72%**	**18.35% / 15.52%**	27.52% / **23.29%**	36.67% / **30.99%**	45.81% / **38.42%**	**54.99% / 46.02%**	**64.14% / 55.17%**	**73.46% / 66.30%**	**83.35% / 79.72%**
MCP($a = 10000$)	9.16% / **7.50%**	18.31% / 15.09%	27.46% / 22.84%	36.61% / **30.61%**	45.79% / **38.36%**	54.94% / **45.92%**	64.10% / **55.76%**	73.37% / **66.94%**	**83.19% / 79.68%**
MCP($a = 5000$)	9.16% / **7.53%**	18.32% / 15.00%	27.46% / 22.51%	36.64% / 30.01%	45.78% / **37.87%**	54.93% / **46.00%**	64.12% / **56.81%**	**73.42% / 67.34%**	**83.46% / 79.52%**
SCAD($a = 15000$)	**9.19% / 7.85%**	**18.36% / 15.50%**	27.52% / **23.12%**	36.68% / **30.65%**	45.84% / **38.33%**	**54.99% / 46.03%**	**64.16% / 55.37%**	**73.45% / 66.81%**	**83.33% / 79.72%**
SCAD($a = 10000$)	9.15% / **7.72%**	18.30% / **15.46%**	27.47% / **23.15%**	36.63% / **30.66%**	45.76% / **38.43%**	54.94% / **46.14%**	**64.14% / 56.10%**	**73.44% / 67.22%**	**83.36% / 79.61%**
SCAD($a = 5000$)	9.15% / 7.37%	18.31% / 14.96%	27.44% / 22.27%	36.59% / 29.79%	45.76% / 37.50%	54.91% / **45.40%**	64.11% / **55.93%**	**73.42% / 67.19%**	**83.53% / 79.75%**

(c)	SVHN								
Channel Pruning Ratio	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90
ℓ_1	9.51% / 9.29%	19.12% / 18.86%	28.61% / 27.83%	38.25% / 37.38%	47.84% / 46.82%	57.48% / 55.88%	67.09% / 65.54%	76.67% / 75.30%	86.15% / 85.06%
$\ell_{3/4}$	**9.63% / 9.69%**	**19.27% / 19.42%**	**28.79% / 28.88%**	**38.42% / 38.25%**	**48.04% / 47.95%**	**57.69% / 57.71%**	**67.25% / 67.11%**	**76.79% / 76.51%**	**86.38% / 85.87%**
$\ell_{1/2}$	**9.62% / 9.38%**	**19.21% / 19.20%**	**28.81% / 28.68%**	**38.44% / 38.46%**	**48.08% / 47.85%**	**57.75% / 57.54%**	**67.43% / 67.40%**	**77.05% / 76.96%**	**86.68% / 86.41%**
$\ell_{1/4}$	**9.68%* / 9.88%***	**19.34% / 19.46%***	**29.05%* / 29.40%***	**38.74%* / 39.33%***	**48.42%* / 49.00%***	**58.12%* / 58.71%***	**67.92%* / 68.69%***	**77.81%* / 78.96%***	**87.81%* / 89.44%***
T$\ell_1(a = 10.0)$	**9.57% / 9.48%**	**19.13% / 19.05%**	**28.72% / 28.73%**	**38.36% / 38.13%**	**47.87% / 47.49%**	**57.51% / 56.91%**	67.06% / **66.32%**	76.64% / **75.74%**	**86.29% / 85.54%**
T$\ell_1(a = 1.0)$	**9.58% / 9.33%**	**19.24% / 19.26%**	**28.92% / 28.77%**	**38.58% / 38.59%**	**48.20% / 48.03%**	**57.82% / 57.66%**	**67.44% / 66.97%**	**77.01% / 76.50%**	**86.57% / 86.17%**
T$\ell_1(a = 0.5)$	**9.62%** / 9.29%	**19.19%** / 18.82%	**28.81% / 28.37%**	**38.51% / 37.98%**	**48.16% / 47.64%**	**57.83% / 57.76%**	**67.46% / 67.27%**	**77.03% / 77.01%**	**86.70% / 86.55%**
MCP($a = 15000$)	**9.65% / 9.52%**	**19.31% / 19.09%**	**28.89% / 28.73%**	**38.40% / 38.03%**	**47.88% / 47.53%**	57.44% / **56.81%**	67.05% / **66.62%**	76.60% / **76.05%**	NA
MCP($a = 10000$)	9.51% / **9.42%**	19.02% / **18.92%**	28.60% / **28.36%**	38.22% / **37.67%**	47.73% / **47.15%**	57.26% / **56.59%**	66.95% / **65.99%**	76.61% / **75.71%**	86.14% / **85.29%**
MCP($a = 5000$)	**9.55% / 9.44%**	**19.14% / 18.89%**	**28.70% / 28.36%**	38.25% / **37.66%**	**47.89% / 47.26%**	57.48% / **56.57%**	67.02% / **66.10%**	76.58% / **75.69%**	86.10% / 84.97%
SCAD($a = 15000$)	**9.55% / 9.31%**	19.09% / 18.75%	**28.71% / 28.52%**	**38.26% / 38.02%**	47.84% / **47.30%**	57.47% / **56.49%**	**67.13% / 66.27%**	76.57% / **75.62%**	NA
SCAD($a = 10000$)	**9.66% / 9.82%**	**19.37%* / 19.25%**	**28.88% / 28.99%**	**38.46% / 38.36%**	**48.06% / 47.87%**	**57.53% / 57.31%**	**67.13% / 66.95%**	76.61% / **76.55%**	NA
SCAD($a = 5000$)	**9.55% / 9.31%**	19.09% / 18.75%	**28.71% / 28.52%**	**38.26% / 38.02%**	47.84% / **47.30%**	57.47% / **56.49%**	**67.13% / 66.27%**	76.57% / **75.62%**	NA

On CIFAR 10, Table 4a shows a quite noticeable difference in the numbers of parameters and FLOPs pruned between ℓ_1 and ℓ_p or T$\ell_1(a = 0.5, 1.0)$. For example, $\ell_{1/2}$ saves at least 10% more weight parameters and at least 8% more FLOPs than ℓ_1 at channel pruning ratio 0.40 and above. On the other hand, SCAD and MCP are outperformed by ℓ_1 in percentages of parameters/FLOPs pruned. According to Figure 5, most of the regularizers do not suffer a significant drop in test accuracy when large number of channels are pruned.

On CIFAR 100, according to Table 4b, $\ell_p(p = 1/4, 1/2)$ and T$\ell_1(a = 0.5, 1.0)$ prune at least 3% more parameters and at least 1% more FLOPs than ℓ_1. However, MCP and SCAD are outperformed by ℓ_1 again for percentages of parameters and FLOPs pruned. In Figure 5, we observe that most of the regularizers are robust against channel pruning since the test accuracies do not drop severely at higher channel pruning ratios.

On SVHN, Table 4c reports that ℓ_p and Tℓ_1 save more parameters and FLOPs than ℓ_1 for channel pruning ratio at least 0.20, while that MCP and SCAD do not. Like for CIFAR 10 and CIFAR 100, most regularizers yield models whose test accuracies are robust against channel pruning according to Figure 5.

In general, the test accuracies of ResNet-164 models with any regularizers, except for $\ell_{1/4}$, are stable against channel pruning. In addition, ℓ_p and T$\ell_1(a = 0.5, 1.0)$ prune more parameters and FLOPs than ℓ_1. Overall, MCP and SCAD do not perform well on ResNet-164.

D. RETRAINING AFTER PRUNING

Because the test accuracy drops after channel pruning for VGG-19 and DenseNet-40 trained on CIFAR 10/100, we retrain the models without regularization on the scaling factors and examine whether or not the original test accuracy

TABLE 4. Effect of channel pruning on the mean pruned parameter / FLOPs percentages (%) on ResNet-164 trained on (a) CIFAR 10, (b) CIFAR 100, and (c) SVHN. The mean is computed from five runs for each regularizer. For each channel pruning ratio, **bold** indicates outperforming ℓ_1; * indicates best value; and NA indicates at least one of the five models is over-pruned.

(a)	CIFAR 10						
Channel Pruning Ratio	0.10	0.20	0.30	0.40	0.50	0.60	0.70
ℓ_1	8.57% / 8.37%	16.67% / 16.62%	24.44% / 24.29%	31.39% / 31.57%	38.50% / 38.31%	46.43% / 45.54%	NA
$\ell_{3/4}$	**10.87% / 10.07%**	**21.51% / 19.54%**	**31.23% / 28.32%**	**40.07% / 36.66%**	**47.86% / 43.96%**	**55.15% / 50.71%**	**62.88% / 58.36%**
$\ell_{1/2}$	**12.13% / 10.96%**	**22.88% / 21.41%**	**33.23% / 31.09%**	**42.54% / 39.81%**	**50.88% / 48.02%**	**58.17% / 55.12%**	**64.88% / 61.46%**
$\ell_{1/4}$	**14.26%* / 13.00%***	**26.44%* / 24.50%***	**37.64%* / 34.93%***	**47.82%* / 44.89%***	**57.10%* / 54.47%***	**65.58%* / 64.27%***	NA
T$\ell_1(a = 10.0)$	**8.99% / 8.56%**	**17.92% / 16.88%**	**25.80%** / 24.18%	**33.26%** / 31.14%	**40.50%** / 38.27%	**47.52%** / 44.88%	NA
T$\ell_1(a = 1.0)$	**11.99% / 10.74%**	**22.86% / 20.72%**	**33.08% / 29.61%**	**42.64% / 38.44%**	**51.03% / 46.02%**	**58.52% / 53.03%**	NA
T$\ell_1(a = 0.5)$	**12.51% / 11.29%**	**23.95% / 21.61%**	**34.43% / 31.01%**	**44.10% / 39.82%**	**52.93% / 47.65%**	**60.81% / 54.86%**	**67.15% / 61.20%**
MCP$(a = 15000)$	8.07% / 7.90%	16.00% / 15.54%	23.46% / 22.69%	30.50% / 29.08%	36.84% / 35.05%	**47.73% / 45.63%**	NA
MCP$(a = 10000)$	7.10% / 7.61%	13.90% / 14.43%	20.58% / 21.06%	26.87% / 27.32%	32.74% / 33.29%	NA	NA
MCP$(a = 5000)$	4.19% / 5.55%	8.64% / 10.97%	12.85% / 16.08%	17.06% / 21.00%	23.89% / 29.34%	NA	NA
SCAD$(a = 15000)$	7.71% / 7.65%	15.53% / 15.44%	22.58% / 22.83%	29.51% / 29.44%	36.44% / 35.82%	**47.47% / 46.15%**	NA
SCAD$(a = 10000)$	7.19% / 7.33%	13.99% / 14.30%	20.51% / 20.88%	26.71% / 27.26%	32.79% / 33.27%	NA	NA
SCAD$(a = 5000)$	4.62% / 5.68%	8.98% / 11.02%	13.24% / 16.23%	17.45% / 21.35%	23.94% / 29.47%	NA	NA

(b)	CIFAR 100						
Channel Pruning Ratio	0.10	0.20	0.30	0.40	0.50	0.60	0.70
ℓ_1	4.01% / 7.42%	7.98% / 14.48%	11.88% / 20.94%	15.72% / 26.88%	NA	NA	NA
$\ell_{3/4}$	**4.95% / 7.55%**	**9.84% / 14.90%**	**14.70% / 22.08%**	**19.15% / 28.15%**	**24.58% / 35.33%**	NA	NA
$\ell_{1/2}$	**5.72% / 8.53%**	**11.35% / 16.51%**	**16.66% / 23.82%**	**21.86% / 30.73%**	**26.64% / 36.87%**	NA	NA
$\ell_{1/4}$	**11.13%* / 11.46%***	**20.98%* / 21.85%***	**30.00%* / 31.48%***	**37.85%* / 41.10%***	NA	NA	NA
T$\ell_1(a = 10.0)$	**4.08%** / 7.07%	**8.29%** / 13.87%	**12.36%** / 20.03%	**16.36%** / 25.92%	NA	NA	NA
T$\ell_1(a = 1.0)$	**6.08% / 8.09%**	**11.96% / 15.67%**	**17.38% / 22.93%**	**22.99% / 29.81%**	**28.27% / 36.36%**	NA	NA
T$\ell_1(a = 0.5)$	**6.37% / 9.25%**	**12.68% / 17.30%**	**18.82% / 25.19%**	**24.87% / 31.89%**	**30.54%* / 38.63%***	NA	NA
MCP$(a = 15000)$	3.64% / 6.64%	7.25% / 12.89%	10.92% / 18.99%	15.22% / 25.05%	NA	NA	NA
MCP$(a = 10000)$	3.51% / 6.65%	7.01% / 12.53%	10.42% / 18.48%	14.45% / 24.81%	NA	NA	NA
MCP$(a = 5000)$	3.32% / 6.37%	6.52% / 12.13%	9.67% / 17.58%	NA	NA	NA	NA
SCAD$(a = 15000)$	3.62% / 6.56%	7.20% / 13.01%	10.88% / 19.36%	15.16% / 25.79%	NA	NA	NA
SCAD$(a = 10000)$	3.53% / 6.36%	6.99% / 12.61%	10.36% / 18.46%	14.54% / 25.33%	NA	NA	NA
SCAD$(a = 5000)$	3.31% / 5.92%	6.59% / 11.83%	9.77% / 17.33%	14.15% / 25.57%	NA	NA	NA

(c)	SVHN						
Channel Pruning Ratio	0.10	0.20	0.30	0.40	0.50	0.60	0.70
ℓ_1	12.32% / 17.02%*	22.70% / 29.19%	32.63% / 41.26%	41.88% / 52.39%	50.14% / 62.14%	NA	NA
$\ell_{3/4}$	**13.09%** / 15.50%	**25.49% / 29.87%**	**36.84% / 42.16%**	**47.02% / 53.46%**	**55.77% / 63.17%**	NA	NA
$\ell_{1/2}$	**13.80%** / 15.21%	**26.62% / 29.57%**	**38.20% / 42.21%**	**48.80% / 53.60%**	**58.45% / 63.91%***	**66.65% / 72.62%**	NA
$\ell_{1/4}$	**15.16%*** / 15.61%	**29.05%* / 29.73%**	**41.52%* / 42.43%**	**52.47%* / 53.73%***	**62.39%* / 63.68%**	**71.50%* / 72.79%***	NA
T$\ell_1(a = 10.0)$	12.13% / 16.66%	**23.13% / 30.10%***	**33.11% / 41.50%**	**42.70% / 52.97%**	**50.87% / 62.07%**	**58.16% / 69.81%**	NA
T$\ell_1(a = 1.0)$	**13.45%** / 15.39%	**25.82% / 29.90%**	**37.29% / 42.59%**	**47.70% / 53.87%**	**56.79% / 63.43%**	NA	NA
T$\ell_1(a = 0.5)$	**14.35%** / 15.83%	**26.94% / 29.53%**	**38.69% / 42.68%***	**48.83% / 53.70%**	**58.31% / 63.81%**	**66.44% / 72.28%**	NA
MCP$(a = 15000)$	12.07% / 15.25%	**23.19%** / 28.99%	**32.89%** / 40.96%	41.67% / 51.50%	49.89% / 60.89%	**57.23% / 68.84%**	NA
MCP$(a = 10000)$	11.39% / 15.19%	22.09% / 28.56%	32.33% / 40.67%	41.32% / 51.23%	49.08% / 60.14%	NA	NA
MCP$(a = 5000)$	9.90% / 13.98%	19.13% / 26.99%	27.85% / 38.51%	35.80% / 48.73%	43.23% / 57.77%	NA	NA
SCAD$(a = 15000)$	11.45% / 15.70%	22.01% / 28.82%	32.14% / 40.65%	41.05% / 51.61%	49.47% / 61.02%	**56.76% / 68.83%**	NA
SCAD$(a = 10000)$	12.30% / 16.86%	22.63% / 29.36%	32.39% / 40.89%	41.23% / 51.75%	NA	NA	NA
SCAD$(a = 5000)$	10.42% / 15.04%	19.82% / 27.80%	28.52% / 38.81%	36.76% / 49.44%	NA	NA	NA

is recovered. For brevity, we analyze ℓ_1 and the nonconvex regularizers whose possible channel pruning percentages are at least the same as ℓ_1's.

1) VGG-19

The results for VGG-19 on CIFAR 10/100 are presented in Table 5. Generally, we observe that the test accuracy after retraining is better than the original test accuracy before channel pruning and retraining. For CIFAR 10, after the models are retrained with 70% of their channels pruned, only ℓ_1 $\ell_{3/4}$, T$\ell_1(a = 1.0, 10.0)$, MCP $(a = 10000, 15000)$, and SCAD $(a = 10000, 15000)$ exceed the baseline test accuracy of 93.83%. Among the nonconvex regularizers, $\ell_{3/4}$ and T$\ell_1(a = 1.0, 10.0)$ yield more compressed models in terms of both parameters and FLOPS but have slightly lower test accuracies than ℓ_1. On the other hand, MCP $(a = 15000, 10000)$ and SCAD $(a = 15000)$ are slightly less compressed than ℓ_1 but have better test accuracies. When 75% of the channels are pruned, their retrained test accuracies decrease slightly due to compressing the models further. Among the nonconvex regularizers, the test accuracy for SCAD $(a = 15000)$ is better than the baseline. Moreover, SCAD $(a = 15000)$ with 75% of its channels pruned requires less parameters and FLOPS than ℓ_1 with 70% of its channels pruned. For $\ell_{1/4}$, when 90% of the channels are pruned, at least 98% of parameters and FLOPs are pruned, but the test accuracy after retraining is 81.57%. For CIFAR 100, with 45% of the channels pruned, all of the regularizers except for $\ell_{1/4}$ and T$\ell_1(a = 0.5)$ attain better test accuracies than the baseline accuracy of 72.73%. Similar to CIFAR 10, $\ell_{3/4}$, $\ell_{1/2}$, and T$\ell_1(a = 1.0, 10.0)$ have slightly lower test accuracies than ℓ_1 but have better compression. MCP and SCAD have better test accuracies than ℓ_1 with similar parameter and FLOP compression. When more channels are pruned, most

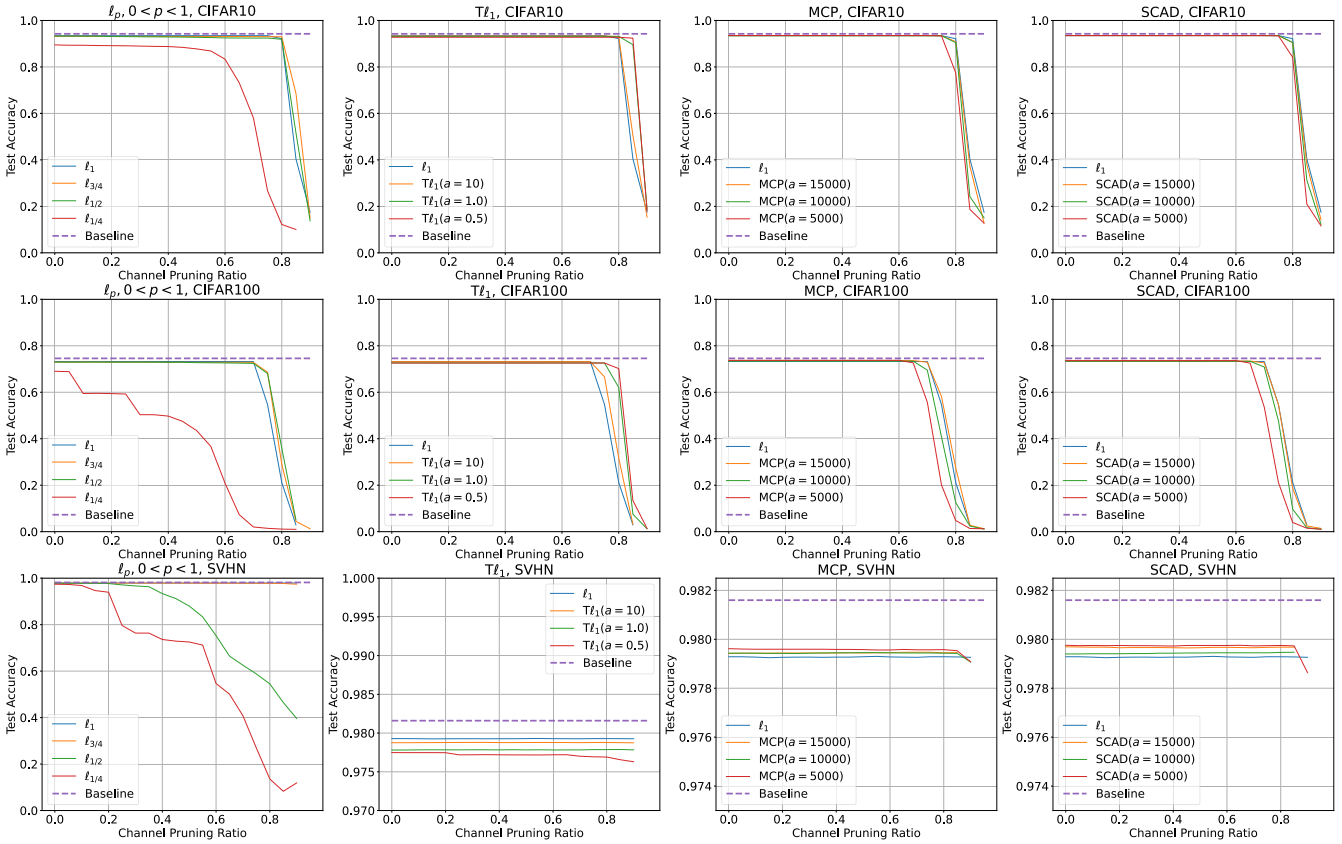


FIGURE 4. Effect of channel pruning on the mean test accuracy of five runs of DenseNet-40 on CIFAR 10/100 and SVHN. Baseline refers to the mean test accuracy of the unregularized model that is not pruned. Baseline accuracies are 94.25% for CIFAR 10, 74.58% for CIFAR 100, and 98.16% for SVHN.

of the regularizers suffer a slight decrease in retrained test accuracies. Only $\ell_{3/4}$ with 55% channels pruned and T$\ell_1(a = 0.5, 1.0)$ with 60% channels pruned experience a modest improvement in test accuracy, but their test accuracies exceed the baseline test accuracy and ℓ_1's test accuracy with 55% channels pruned.

Overall, for $\ell_p(p = 1/2, 3/4)$ and T$\ell_1(a = 0.5, 1.0)$, the retrained models, despite being more compressed than their ℓ_1 counterparts, have slightly lower test accuracies. However, MCP and SCAD have similar compression as ℓ_1 but with better test accuracies after retraining.

2) DENSENET-40

Table 6 reports the results for DenseNet-40 on CIFAR 10/100. Overall, the baseline accuracy is better than all of the retrained test accuracies, but the differences are at most 3.07% for CIFAR 10 and at most 6.82% for CIFAR 100. For CIFAR 10, when 82.5% of the channels are pruned, only MCP ($a = 10000$) and SCAD ($a = 10000$) have better test accuracies than ℓ_1 with similar compression in parameters and FLOPs. For $\ell_p(p = 1/2, 3/4)$ and T$\ell_1(a = 1.0)$, their retrained test accuracies are only slightly lower by at most 0.20%, but this is at the cost of better compression. When 90% of the channels are pruned, the retrained test accuracies decrease slightly more into the range of 91%-92%. Only $\ell_{3/4}$ and T$\ell_1(a = 0.5, 1.0)$ have better test accuracies than ℓ_1 with much better compression. For CIFAR 100, when 75% of the channels are pruned, $\ell_{3/4}$, T$\ell_1(a = 10.0)$, MCP, and SCAD have at least the same test accuracies as ℓ_1 with better compression in parameters and FLOPs. However, increasing the channel pruning percentage to 90% causes their retrained test accuracies to deteriorate. As a result, none of the models is able to exceed the test accuracy of the ℓ_1-regularized models retrained with 85% of their channels pruned. For $\ell_{1/2}$ and T$\ell_1(a = 10.0)$, when 85% of the channels are pruned, their test accuracies exceed ℓ_1.

In general, pruning channels for DenseNet at the highest percentage possible can be detrimental to the retrained test accuracy. When channels are pruned at intermediate levels, the nonconvex regularizers can have better retrained test accuracies and/or better compression than ℓ_1.

E. SCALING FACTOR ANALYSIS

In order to better understand how ℓ_1 and the nonconvex regularizers affect the scaling factors γ, we plot histograms of the counts of the $\log_{10}(|\gamma|)$ averaged from the five models trained for each model and regularizer. Figures 6-14 provide the histograms while Table 7 records the average number of scaling factors whose magnitudes are less than 10^{-6} and more than 10^{-6}. The value 10^{-6} is chosen because generally,

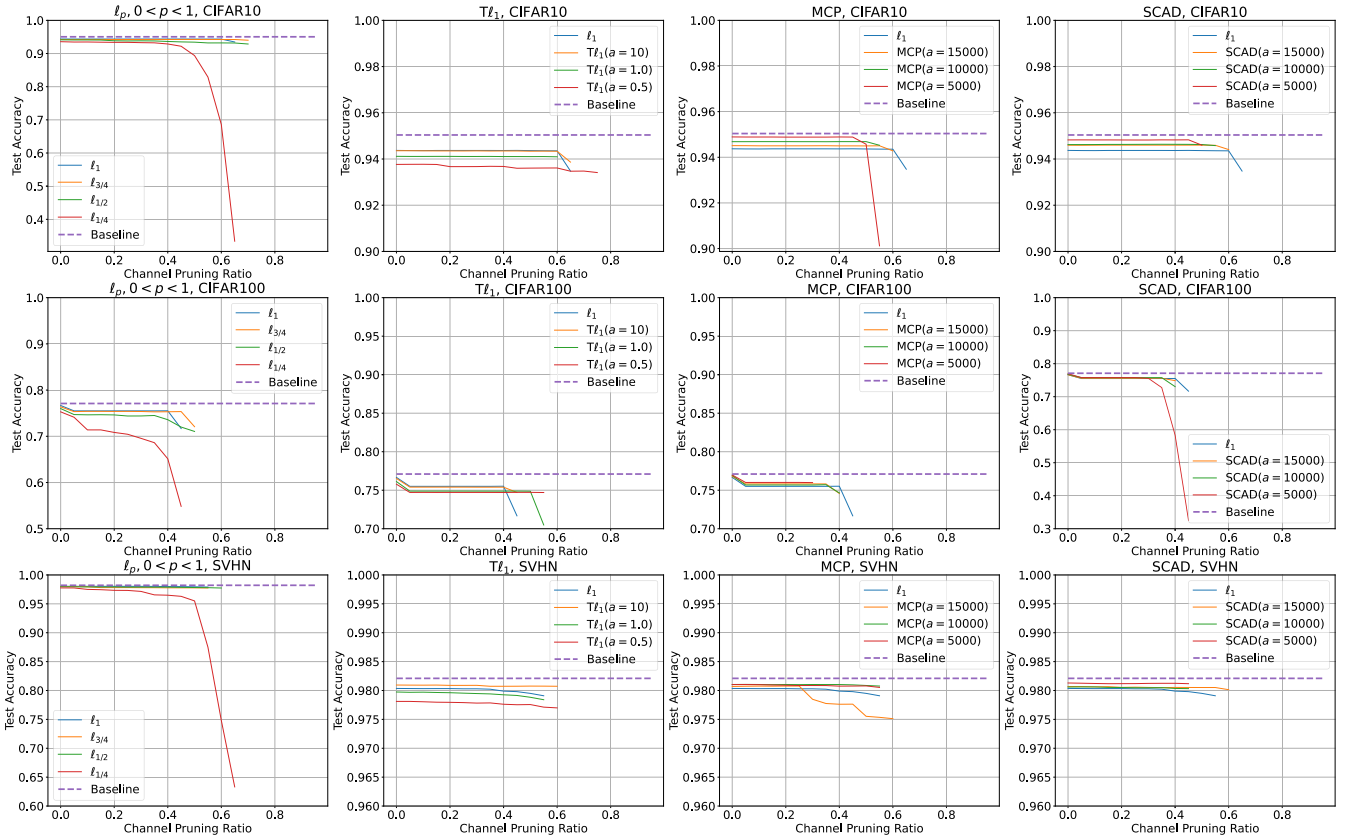


FIGURE 5. **Effect of channel pruning on the mean test accuracy of five runs of ResNet-164 on CIFAR 10/100 and SVHN. Baseline refers to the mean test accuracy of the unregularized model that is not pruned. Baseline accuracies are 95.04% for CIFAR 10, 77.10% for CIFAR 100, and 98.21% for SVHN.**

any value below it has negligible effect on the numerical computation [89].

For CIFAR 10, Figures 6-8 show the histograms while Table 7a provides the average counts of the scaling factors based on their magnitudes. For all three networks, we observe the following phenomena. MCP and SCAD have similar scaling factor distributions as ℓ_1 across all given values of a. Moreover, MCP, SCAD, and ℓ_1 have similar number of scaling factors whose magnitudes are less than 10^{-6} as verified by Table 7a. This may explain why their compression rates are similar to ℓ_1 in our earlier analyses. For ℓ_p, we see that $\ell_{3/4}$ has most of its scaling factors within the interval $(10^{-6}, 10^{-5})$. As p decreases, the values of the scaling factors tend farther away from 0. In fact, majority of the scaling factors for $\ell_{1/2}$ and $\ell_{1/4}$ are at least 10^{-6} in magnitude. Specifically for $\ell_{1/4}$, most of the scaling factors have absolute values at least 0.10. Hence, we can see why $\ell_{1/4}$ is sensitive to channel pruning. Lastly, for Tℓ_1, more scaling factors decrease towards 0 in magnitude as a decreases. Moreover, we observe that most of the scaling factors are accumulated within the interval $(10^{-7}, 10^{-6})$. Because Tℓ_1 causes more scaling factors to decrease towards 0 in magnitude, this might explain why Tℓ_1 is robust against channel pruning.

For CIFAR 100, Figures 9-11 show the histograms of the scaling factors while Table 7b records the average counts by magnitudes. Because CIFAR 100 is a more difficult classification dataset compared to CIFAR 10, most of the scaling factors appear to be within the interval $(10^{-1}, 1)$. However, DenseNet-40 shows bimodal distributions for Tℓ_1, MCP, and SCAD. Table 7b shows that more than half of the scaling factors are less than 10^{-6} in magnitudes in DenseNet-40 for most regularizers, but they are more than 10^{-6} in magnitudes in VGG-19 and ResNet-164 for all regularizers. The distributions of the scaling factors convey why DenseNet-40 can be pruned at higher channel pruning ratios than VGG-19 and ResNet-164, as indicated by the middle rows of Figures 3-5. Across the three networks, MCP and SCAD have similar distributions with ℓ_1. For $\ell_{3/4}$, a considerable amount of scaling factors are within the interval $(10^{-6}, 10^{-5})$, but as p decreases, the magnitudes of most scaling factors increase. Hence, less than a few hundred scaling factors are below 10^{-6} in magnitudes. As a result, models regularized with $\ell_{1/2}$ and $\ell_{1/4}$ become more sensitive to channel pruning as demonstrated earlier. For T$\ell_1(a = 10.0)$, its distribution of scaling factors is similar to ℓ_1. However, when $a = 0.5, 1.0$, more scaling factors have magnitudes less than 10^{-5}, which demonstrates Tℓ_1's robustness to channel pruning when a is small enough.

Figures 12-14 and Table 7c provide statistics about SVHN. For all three networks, T$\ell_1(a = 10.0)$, SCAD, and MCP

TABLE 5. Results from five retrained VGG-19 on CIFAR 10/100 after pruning. Baseline refers to the VGG-19 model trained without regularization on the scaling factors.

	Number of Parameters/FLOPs	Percentage of Parameters/FLOPs Pruned (%)	Mean Test Accuracy before Retraining (%)	Mean Test Accuracy after Retraining (%)
Baseline	20.04M/7.97 $\times 10^8$	0.00/0.00	93.83	N/A
ℓ_1 (0% Pruned)	20.04M/7.97 $\times 10^8$	0.00/0.00	93.63	N/A
ℓ_1 (70% Pruned)	2.24M/3.83 $\times 10^8$	88.81/51.93	28.28	93.91
$\ell_{3/4}$ (0% Pruned)	20.04M/7.97 $\times 10^8$	0.00/0.00	93.53	N/A
$\ell_{3/4}$ (70% Pruned)	2.07M/3.59 $\times 10^8$	89.69/54.96	88.87	93.90
$\ell_{3/4}$ (75% Pruned)	1.79M/3.43 $\times 10^8$	91.06/57.00	16.18	93.79
$\ell_{1/2}$ (0% Pruned)	20.04M/7.97 $\times 10^8$	0.00/0.00	93.57	N/A
$\ell_{1/2}$ (70% Pruned)	2.00M/3.50 $\times 10^8$	90.01/56.12	40.07	93.77
$\ell_{1/2}$ (75% Pruned)	1.66M/3.25 $\times 10^8$	91.70/59.20	13.65	93.82
$\ell_{1/4}$ (0% Pruned)	20.04M/7.97 $\times 10^8$	0.00/0.00	86.97	N/A
$\ell_{1/4}$ (70% Pruned)	1.58M/1.44 $\times 10^8$	92.14/81.89	47.59	92.15
$\ell_{1/4}$ (90% Pruned)	0.19M/0.13 $\times 10^8$	99.05/98.32	10.00	81.57
T$\ell_1(a = 10.0)$ (0% Pruned)	20.04M/7.97 $\times 10^8$	0.00/0.00	93.64	N/A
T$\ell_1(a = 10.0)$ (70% Pruned)	2.19M/3.77 $\times 10^8$	89.06/52.75	47.70	93.86
T$\ell_1(a = 10.0)$ (75% Pruned)	1.84M/3.49 $\times 10^8$	90.82/56.19	10.00	93.72
T$\ell_1(a = 1.0)$ (0% Pruned)	20.04M/7.97 $\times 10^8$	0.00/0.00	93.55	N/A
T$\ell_1(a = 1.0)$ (70% Pruned)	1.93M/3.39 $\times 10^8$	90.35/57.43	93.54	93.86
T$\ell_1(a = 1.0)$ (75% Pruned)	1.66M/3.24 $\times 10^8$	91.71/59.29	86.83	93.82
T$\ell_1(a = 0.5)$ (0% Pruned)	20.04M/7.97 $\times 10^8$	0.00/0.00	93.15	N/A
T$\ell_1(a = 0.5)$ (70% Pruned)	1.83M/3.20 $\times 10^8$	90.88/59.84	93.14	93.75
T$\ell_1(a = 0.5)$ (75% Pruned)	1.53M/3.05 $\times 10^8$	92.38/61.74	92.38	93.77
MCP ($a = 15000$) (0% Pruned)	20.04M/7.97 $\times 10^8$	0.00/0.00	93.65	N/A
MCP ($a = 15000$) (70% Pruned)	2.29M/3.93 $\times 10^8$	88.58/50.69	47.18	93.97
MCP ($a = 15000$) (75% Pruned)	1.89M/3.58 $\times 10^8$	90.58/55.04	10.00	93.68
MCP ($a = 10000$) (0% Pruned)	20.04M/7.97 $\times 10^8$	0.00/0.00	93.69	N/A
MCP ($a = 10000$) (70% Pruned)	2.28M/3.95 $\times 10^8$	88.63/50.49	40.24	94.12
MCP ($a = 10000$) (75% Pruned)	1.89M/3.62 $\times 10^8$	90.56/54.54	10.00	93.73
SCAD ($a = 15000$) (0% Pruned)	20.04M/7.97 $\times 10^8$	0.00/0.00	93.64	N/A
SCAD ($a = 15000$) (70% Pruned)	2.26M/3.93 $\times 10^8$	88.71/50.70	52.72	93.94
SCAD ($a = 15000$) (75% Pruned)	1.87M/3.59 $\times 10^8$	90.65/54.97	10.00	93.91
SCAD ($a = 10000$) (0% Pruned)	20.04M/7.97 $\times 10^8$	0.00/0.00	93.60	N/A
SCAD ($a = 10000$) (70% Pruned)	2.29M/3.95 $\times 10^8$	88.57/50.43	55.25	93.88

(a) CIFAR 10

	Number of Parameters/FLOPs	Percentage of Parameters/FLOPs Pruned (%)	Mean Test Accuracy before Retraining (%)	Mean Test Accuracy after Retraining (%)
Baseline	20.08M/7.97 $\times 10^8$	0.00/0.00	72.73	N/A
ℓ_1 (0% Pruned)	20.08M/7.97 $\times 10^8$	0.00/0.00	72.57	N/A
ℓ_1 (45% Pruned)	5.67M/5.26 $\times 10^8$	71.78/34.00	51.16	73.44
ℓ_1 (55% Pruned)	4.31M/4.89 $\times 10^8$	78.53/38.66	1.00	72.98
$\ell_{3/4}$ (0% Pruned)	20.08M/7.97 $\times 10^8$	0.00/0.00	72.14	N/A
$\ell_{3/4}$ (45% Pruned)	5.49M/5.04 $\times 10^8$	72.68/36.75	71.76	73.24
$\ell_{3/4}$ (55% Pruned)	4.10M/4.76 $\times 10^8$	79.59/40.28	3.40	73.26
$\ell_{1/2}$ (0% Pruned)	20.08M/7.97 $\times 10^8$	0.00/0.00	72.06	N/A
$\ell_{1/2}$ (45% Pruned)	5.38M/5.03 $\times 10^8$	73.21/36.95	71.27	73.34
$\ell_{1/2}$ (60% Pruned)	3.40M/4.48 $\times 10^8$	83.07/43.82	1.08	71.59
$\ell_{1/4}$ (0% Pruned)	20.08M/7.97 $\times 10^8$	0.00/0.00	70.95	N/A
$\ell_{1/4}$ (45% Pruned)	5.30M/4.76 $\times 10^8$	73.59/40.26	22.70	72.50
$\ell_{1/4}$ (80% Pruned)	0.69M/1.05 $\times 10^8$	96.54/86.86	1.00	46.97
T$\ell_1(a = 10.0)$ (0% Pruned)	20.08M/7.97 $\times 10^8$	0.00/0.00	72.36	N/A
T$\ell_1(a = 10.0)$ (45% Pruned)	5.53M/5.18 $\times 10^8$	72.45/34.95	69.35	73.39
T$\ell_1(a = 10.0)$ (55% Pruned)	4.21M/4.85 $\times 10^8$	79.05/39.19	1.46	73.17
T$\ell_1(a = 1.0)$ (0% Pruned)	20.08M/7.97 $\times 10^8$	0.00/0.00	72.07	N/A
T$\ell_1(a = 1.0)$ (45% Pruned)	5.39M/4.87 $\times 10^8$	73.16/38.89	72.07	73.03
T$\ell_1(a = 1.0)$ (60% Pruned)	3.43M/4.47 $\times 10^8$	82.90/43.94	1.84	73.06
T$\ell_1(a = 0.5)$ (0% Pruned)	20.08M/7.97 $\times 10^8$	0.00/0.00	71.63	N/A
T$\ell_1(a = 0.5)$ (45% Pruned)	5.29M/4.74 $\times 10^8$	73.66/40.48	71.63	72.69
T$\ell_1(a = 0.5)$ (60% Pruned)	3.19M/4.21 $\times 10^8$	84.09/47.15	66.50	72.81
MCP ($a = 15000$) (0% Pruned)	20.08M/7.97 $\times 10^8$	0.00/0.00	72.26	N/A
MCP ($a = 15000$) (45% Pruned)	5.66M/5.27 $\times 10^8$	71.82/33.87	66.14	73.68
MCP ($a = 15000$) (55% Pruned)	4.30M/4.92 $\times 10^8$	78.58/38.21	1.00	72.94
SCAD ($a = 15000$) (0% Pruned)	20.08M/7.97 $\times 10^8$	0.00/0.00	72.50	N/A
SCAD ($a = 15000$) (45% Pruned)	5.64M/5.26 $\times 10^8$	71.89/33.99	65.72	73.61
SCAD ($a = 15000$) (55% Pruned)	4.32M/4.90 $\times 10^8$	78.48/38.49	1.00	72.67
SCAD ($a = 10000$) (0% Pruned)	20.08M/7.97 $\times 10^8$	0.00/0.00	72.33	N/A
SCAD ($a = 10000$) (45% Pruned)	5.72M/5.32 $\times 10^8$	71.50/33.21	64.98	73.52
SCAD ($a = 10000$) (55% Pruned)	4.37M/4.94 $\times 10^8$	78.22/37.99	1.00	71.98

(b) CIFAR 100

TABLE 6. Results from five 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/FLOPs	Percentage of Parameters/FLOPs Pruned (%)	Mean Test Accuracy before Retraining (%)	Mean Test Accuracy after Retraining (%)
Baseline	$1.02\text{M}/5.33 \times 10^8$	0.00/0.00	94.25	N/A
ℓ_1 (0 % Pruned)	$1.02\text{M}/5.33 \times 10^8$	0.00/0.00	93.46	N/A
ℓ_1 (82.5% Pruned)	$0.24\text{M}/1.54 \times 10^8$	76.21/71.20	78.27	93.46
ℓ_1 (90% Pruned)	$0.17\text{M}/1.08 \times 10^8$	83.76/79.75	17.47	91.42
$\ell_{3/4}$ (0% Pruned)	$1.02\text{M}/5.33 \times 10^8$	0.00/0.00	93.19	N/A
$\ell_{3/4}$ (82.5% Pruned)	$0.24\text{M}/1.53 \times 10^8$	76.57/71.34	90.17	93.33
$\ell_{3/4}$ (90% Pruned)	$0.16\text{M}/1.06 \times 10^8$	84.02/80.07	15.06	91.54
$\ell_{1/2}$ (0% Pruned)	$1.02\text{M}/5.33 \times 10^8$	0.00/0.00	93.28	N/A
$\ell_{1/2}$ (82.5% Pruned)	$0.25\text{M}/1.51 \times 10^8$	76.84/71.76	83.17	93.43
$\ell_{1/2}$ (90% Pruned)	$0.16\text{M}/1.06 \times 10^8$	84.36/80.13	13.76	91.31
$\ell_{1/4}$ (0% Pruned)	$1.02\text{M}/5.33 \times 10^8$	0.00/0.00	89.48	N/A
$\ell_{1/4}$ (82.5% Pruned)	$0.21\text{M}/1.14 \times 10^8$	79.81/78.63	11.29	91.68
$\ell_{1/4}$ (85% Pruned)	$0.18\text{M}/0.98 \times 10^8$	82.57/81.64	10.05	91.44
T$\ell_1(a = 10.0)$ (0% Pruned)	$1.02\text{M}/5.33 \times 10^8$	0.00/0.00	93.30	N/A
T$\ell_1(a = 10.0)$ (82.5% Pruned)	$0.24\text{M}/1.54 \times 10^8$	76.33/71.10	83.24	93.38
T$\ell_1(a = 10.0)$ (90% Pruned)	$0.16\text{M}/1.08 \times 10^8$	83.89/79.72	15.35	91.37
T$\ell_1(a = 1.0)$ (0% Pruned)	$1.02\text{M}/5.33 \times 10^8$	0.00/0.00	93.16	N/A
T$\ell_1(a = 1.0)$ (82.5% Pruned)	$0.24\text{M}/1.53 \times 10^8$	76.80/71.35	93.17	93.26
T$\ell_1(a = 1.0)$ (90% Pruned)	$0.16\text{M}/1.06 \times 10^8$	84.23/80.19	18.91	91.70
T$\ell_1(a = 0.5)$ (0% Pruned)	$1.02\text{M}/5.33 \times 10^8$	0.00/0.00	92.78	N/A
T$\ell_1(a = 0.5)$ (82.5% Pruned)	$0.23\text{M}/1.50 \times 10^8$	77.21/71.83	92.74	93.05
T$\ell_1(a = 0.5)$ (90% Pruned)	$0.16\text{M}/1.03 \times 10^8$	84.45/80.70	18.12	91.69
MCP($a = 15000$) (0% Pruned)	$1.02\text{M}/5.33 \times 10^8$	0.00/0.00	93.48	N/A
MCP($a = 15000$)(82.5% Pruned)	$0.24\text{M}/1.55 \times 10^8$	76.23/71.00	92.74	93.44
MCP($a = 15000$) (90% Pruned)	$0.17\text{M}/1.10 \times 10^8$	83.72/79.37	12.92	91.31
MCP($a = 10000$) (0% Pruned)	$1.02\text{M}/5.33 \times 10^8$	0.00/0.00	93.41	N/A
MCP($a = 10000$) (82.5% Pruned)	$0.24\text{M}/1.53 \times 10^8$	76.37/71.23	67.36	93.53
MCP($a = 10000$) (90% Pruned)	$0.16\text{M}/1.10 \times 10^8$	83.85/79.39	15.08	91.24
SCAD($a = 15000$) (0% Pruned)	$1.02\text{M}/5.33 \times 10^8$	0.00/0.00	93.48	N/A
SCAD($a = 15000$) (82.5% Pruned)	$0.24\text{M}/1.54 \times 10^8$	76.28/71.02	71.33	93.42
SCAD($a = 15000$) (90% Pruned)	$0.17\text{M}/1.10 \times 10^8$	83.80/79.42	14.21	91.26
SCAD($a = 10000$) (0% Pruned)	$1.02\text{M}/5.33 \times 10^8$	0.00/0.00	93.52	N/A
SCAD($a = 10000$) (82.5% Pruned)	$0.24\text{M}/1.55 \times 10^8$	76.25/70.93	71.49	93.49
SCAD($a = 10000$) (90% Pruned)	$0.17\text{M}/1.10 \times 10^8$	83.75/79.27	12.27	91.18

(a) CIFAR 10

	Number of Parameters/FLOPs	Percentage of Parameters/FLOPs Pruned (%)	Mean Test Accuracy before Retraining (%)	Mean Test Accuracy after Retraining (%)
Baseline	$1.06\text{M}/5.33 \times 10^8$	0.00/0.00	74.58	N/A
ℓ_1 (0% Pruned)	$1.06\text{M}/5.33 \times 10^8$	0.00/0.00	73.24	N/A
ℓ_1 (75% Pruned)	$0.35\text{M}/2.14 \times 10^8$	68.73/59.89	54.68	73.73
ℓ_1 (85% Pruned)	$0.23\text{M}/1.46 \times 10^8$	78.08/72.60	2.94	72.40
$\ell_{3/4}$ (0% Pruned)	$1.06\text{M}/5.33 \times 10^8$	0.00/0.00	72.97	N/A
$\ell_{3/4}$ (75% Pruned)	$0.33\text{M}/2.11 \times 10^8$	68.93/60.40	68.60	73.75
$\ell_{3/4}$ (90% Pruned)	$0.18\text{M}/1.07 \times 10^8$	83.34/79.89	1.23	69.33
$\ell_{1/2}$ (0% Pruned)	$1.06\text{M}/5.33 \times 10^8$	0.00/0.00	72.98	N/A
$\ell_{1/2}$ (75% Pruned)	$0.33\text{M}/2.06 \times 10^8$	69.03/61.41	68.05	73.39
$\ell_{1/2}$ (85% Pruned)	$0.23\text{M}/1.42 \times 10^8$	78.42/73.43	5.05	72.52
$\ell_{1/4}$ (0% Pruned)	$1.06\text{M}/5.33 \times 10^8$	0.00/0.00	69.02	N/A
$\ell_{1/4}$ (75% Pruned)	$0.31\text{M}/1.62 \times 10^8$	70.81/69.59	1.45	71.62
$\ell_{1/4}$ (85% Pruned)	$0.19\text{M}/0.88 \times 10^8$	82.28/83.54	1.00	67.76
T$\ell_1(a = 10.0)$ (0% Pruned)	$1.06\text{M}/5.33 \times 10^8$	0.00/0.00	73.18	N/A
T$\ell_1(a = 10.0)$ (75% Pruned)	$0.33\text{M}/2.12 \times 10^8$	68.84/60.18	66.62	73.78
T$\ell_1(a = 10.0)$ (85% Pruned)	$0.23\text{M}/1.47 \times 10^8$	78.21/72.37	3.17	72.69
T$\ell_1(a = 1.0)$ (0% Pruned)	$1.06\text{M}/5.33 \times 10^8$	0.00/0.00	72.63	N/A
T$\ell_1(a = 1.0)$ (75% Pruned)	$0.33\text{M}/2.12 \times 10^8$	69.16/60.24	72.60	73.42
T$\ell_1(a = 1.0)$ (90% Pruned)	$0.18\text{M}/1.07 \times 10^8$	83.48/80.01	1.24	69.98
T$\ell_1(a = 0.5)$ (0% Pruned)	$1.06\text{M}/5.33 \times 10^8$	0.00/0.00	72.57	N/A
T$\ell_1(a = 0.5)$ (75% Pruned)	$0.33\text{M}/2.10 \times 10^8$	69.33 /60.56	72.59	73.23
T$\ell_1(a = 0.5)$ (90% Pruned)	$0.17\text{M}/1.06 \times 10^8$	83.61/80.16	1.37	70.16
MCP($a = 15000$) (0% Pruned)	$1.06\text{M}/5.33 \times 10^8$	0.00/0.00	73.64	N/A
MCP($a = 15000$)(75% Pruned)	$0.33\text{M}/2.10 \times 10^8$	68.80/60.61	58.12	73.73
MCP($a = 15000$) (90% Pruned)	$0.18\text{M}/1.08 \times 10^8$	83.35/79.73	1.27	69.94
MCP($a = 10000$) (0% Pruned)	$1.06\text{M}/5.33 \times 10^8$	0.00/0.00	73.40	N/A
MCP($a = 10000$) (75% Pruned)	$0.33\text{M}/2.06 \times 10^8$	68.73/61.36	40.76	73.95
MCP($a = 10000$) (90% Pruned)	$0.18\text{M}/1.08 \times 10^8$	83.19/79.68	1.10	69.10
SCAD($a = 15000$) (0% Pruned)	$1.06\text{M}/5.33 \times 10^8$	0.00/0.00	73.41	N/A
SCAD($a = 15000$) (75% Pruned)	$0.33\text{M}/2.09 \times 10^8$	68.79/60.83	54.71	73.97
SCAD($a = 15000$) (90% Pruned)	$0.18\text{M}/1.08 \times 10^8$	83.33/79.72	1.42	69.87
SCAD($a = 10000$) (0% Pruned)	$1.06\text{M}/5.33 \times 10^8$	0.00/0.00	73.37	N/A
SCAD($a = 10000$) (75% Pruned)	$0.33\text{M}/2.04 \times 10^8$	68.80/61.66	47.70	73.75
SCAD($a = 10000$) (90% Pruned)	$0.18\text{M}/1.09 \times 10^8$	83.36/79.61	1.08	69.73

(b) CIFAR 100

TABLE 7. Counts of scaling factors that are averaged across five runs per model and regularizer.

	VGG-19		DenseNet-40		ResNet-164	
	$\lvert\gamma\rvert \leq 10^{-6}$	$\lvert\gamma\rvert > 10^{-6}$	$\lvert\gamma\rvert \leq 10^{-6}$	$\lvert\gamma\rvert > 10^{-6}$	$\lvert\gamma\rvert \leq 10^{-6}$	$\lvert\gamma\rvert > 10^{-6}$
ℓ_1	3483	2021	7309.2	2050.8	7321.4	4790.6
$\ell_{3/4}$	3244.8	2259.2	6261.4	3098.6	7944.2	4167.8
$\ell_{1/2}$	263.4	5240.6	490.6	8869.4	898	11214
$\ell_{1/4}$	3	5501	4	9356	11.6	12100.4
T$\ell_1(a = 10.0)$	3559.6	1944.4	7372.8	1987.2	7466.6	4645.4
T$\ell_1(a = 1.0)$	4021.2	1482.8	7731.4	1628.6	8757.2	3354.8
T$\ell_1(a = 0.5)$	4216	1288	7839	1521	9192	2920
MCP ($a = 15000$)	3472.4	2031.6	7180.6	2179.4	6805.8	5306.2
MCP ($a = 10000$)	3485	2019	7123.6	2236.4	6438.4	5673.6
MCP ($a = 5000$)	3440.4	2063.6	6880.2	2479.8	5542.6	6569.4
SCAD ($a = 15000$)	3492.6	2011.4	7204.4	2155.6	6818.4	5293.6
SCAD ($a = 10000$)	3460.2	2043.8	7121.4	2238.6	6484.6	5627.4
SCAD ($a = 5000$)	3518.2	1985.8	6947.8	2412.2	5514.6	6597.4

(a) CIFAR 10

	VGG-19		DenseNet-40		ResNet-164	
	$\lvert\gamma\rvert \leq 10^{-6}$	$\lvert\gamma\rvert > 10^{-6}$	$\lvert\gamma\rvert \leq 10^{-6}$	$\lvert\gamma\rvert > 10^{-6}$	$\lvert\gamma\rvert \leq 10^{-6}$	$\lvert\gamma\rvert > 10^{-6}$
ℓ_1	1417.2	4086.8	6382	2978	5030.4	7081.6
$\ell_{3/4}$	1895.8	3608.2	2208.6	7151.4	5584.6	6527.4
$\ell_{1/2}$	151.6	5352.4	94.4	9265.6	430	11682
$\ell_{1/4}$	1.6	5502.4	6	9354	6.6	12105.4
T$\ell_1(a = 10.0)$	1629.6	3874.4	6555.4	2804.6	5192.8	6919.2
T$\ell_1(a = 1.0)$	2555.6	2948.4	6919.8	2440.2	6250	5862
T$\ell_1(a = 0.5)$	2802	2702	6889.6	2470.4	6739	5373
MCP ($a = 15000$)	1495	4009	6192.2	3167.8	4521.8	7590.2
MCP ($a = 10000$)	1440.4	4063.6	6055.6	3304.4	4191.8	7920.2
MCP ($a = 5000$)	1378	4126	5627.4	3732.6	3541.8	8570.2
SCAD ($a = 15000$)	1514.4	3989.6	6190.4	3169.6	4481.6	7630.4
SCAD ($a = 10000$)	1481.6	4022.4	6034.6	3325.4	4211.6	7900.4
SCAD ($a = 5000$)	1262	4242	5595.6	3764.4	3484.6	8627.4

(b) CIFAR 100

	CIFAR 10		CIFAR 100		SVHN	
	$\lvert\gamma\rvert \leq 10^{-6}$	$\lvert\gamma\rvert > 10^{-6}$	$\lvert\gamma\rvert \leq 10^{-6}$	$\lvert\gamma\rvert > 10^{-6}$	$\lvert\gamma\rvert \leq 10^{-6}$	$\lvert\gamma\rvert > 10^{-6}$
ℓ_1	4447.6	1056.4	8447.4	912.6	10058.8	2053.2
$\ell_{3/4}$	3862.2	1641.8	7079	2281	10130.4	1981.6
$\ell_{1/2}$	292	5212	543.4	8816.6	1070.4	11041.6
$\ell_{1/4}$	3.4	5500.6	7.2	9352.8	12.6	12099.4
T$\ell_1(a = 10.0)$	4505.4	998.6	8497.4	862.6	10184.8	1927.2
T$\ell_1(a = 1.0)$	4796.8	707.2	8674	686	10813.2	1298.8
T$\ell_1(a = 0.5)$	4874	630	8746.4	613.6	11002.4	1109.6
MCP ($a = 15000$)	4365.6	1138.4	8419.8	940.2	9930.4	2181.6
MCP ($a = 10000$)	4356.6	1147.4	8390.4	969.6	9841	2271
MCP ($a = 5000$)	4242.6	1261.4	8330	1030	9333.6	2778.4
SCAD ($a = 15000$)	4378.2	1125.8	8405.6	954.4	9894.8	2217.2
SCAD ($a = 10000$)	4361.4	1142.6	8407	953	9858.8	2253.2
SCAD ($a = 5000$)	4244.4	1259.6	8330.2	1029.8	9353.8	2758.2

(c) SVHN

have similar distributions as ℓ_1. Similar to CIFAR 10 and 100, $\ell_{3/4}$ has most of its scaling factors to be in the interval $(10^{-6}, 10^{-5})$, but as p decreases for ℓ_p, the magnitudes of the scaling factors increase, resulting in at least 90% of the scaling factors to be at least 10^{-6} in magnitudes as shown in Table 7c. For T$\ell_1(a = 0.5, 1.0)$ on the other hand, most of the scaling factors are in the interval $(10^{-7}, 10^{-6})$.

F. COMPARISON WITH VARIATIONAL CNN PRUNING

We have shown that network slimming with nonconvex regularizers can outperform the original with ℓ_1 regularization. Now we compare our proposed method with variational CNN pruning (VCP) proposed in [44], a Bayesian version of network slimming. VCP is designed to be robust against channel pruning, so we compare it with T$\ell_1(a = 0.5, 1.0)$, which is proven to also be robust against channel pruning in our earlier analyses. The comparisons between the two methods are shown in Table 8, using results from DenseNet-40 and ResNet-164 trained on CIFAR 10/100.

For DenseNet-40 trained on CIFAR 10, Tℓ_1 has a minimally better accuracy with less parameters pruned than VCP with 60% channels pruned. However, we can increase the percentage of channels pruned to 80% for Tℓ_1 so that the number of parameters are reduced while maintaining the same accuracy. On CIFAR 100, with similar percentages of channels pruned, Tℓ_1 has a much better accuracy than VCP

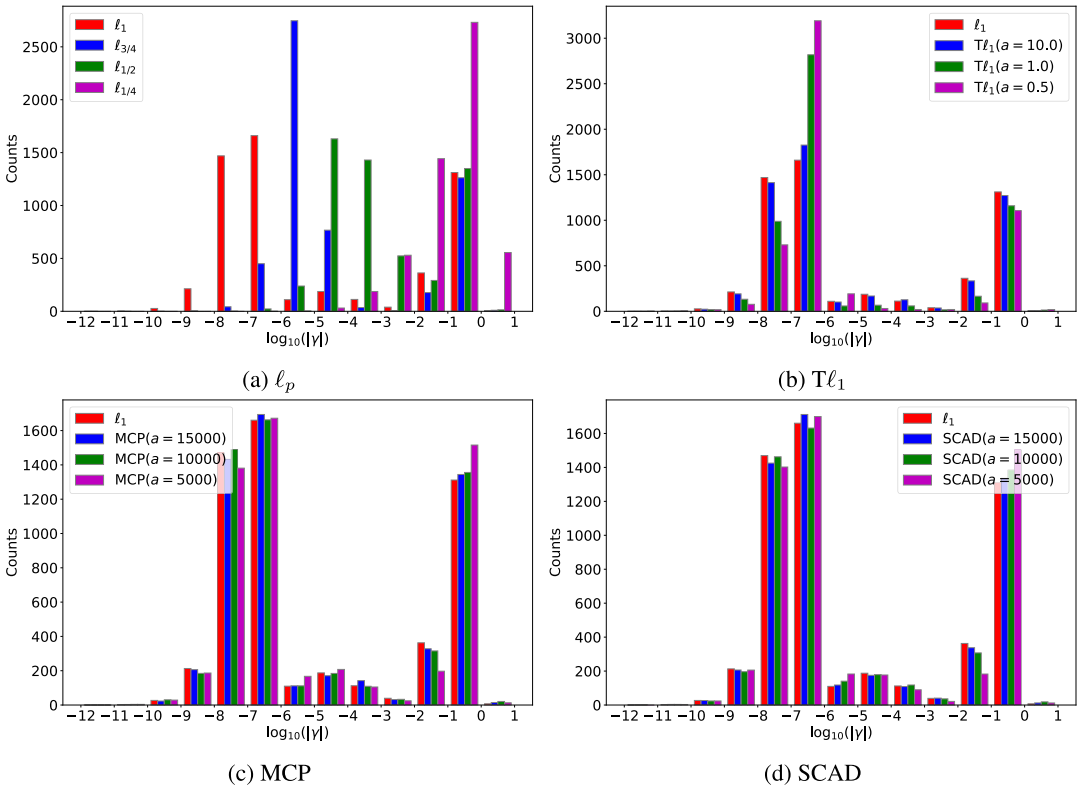


FIGURE 6. Histogram of scaling factors γ in VGG-19 trained on CIFAR 10. The x-axis is $\log_{10}(|\gamma|)$.

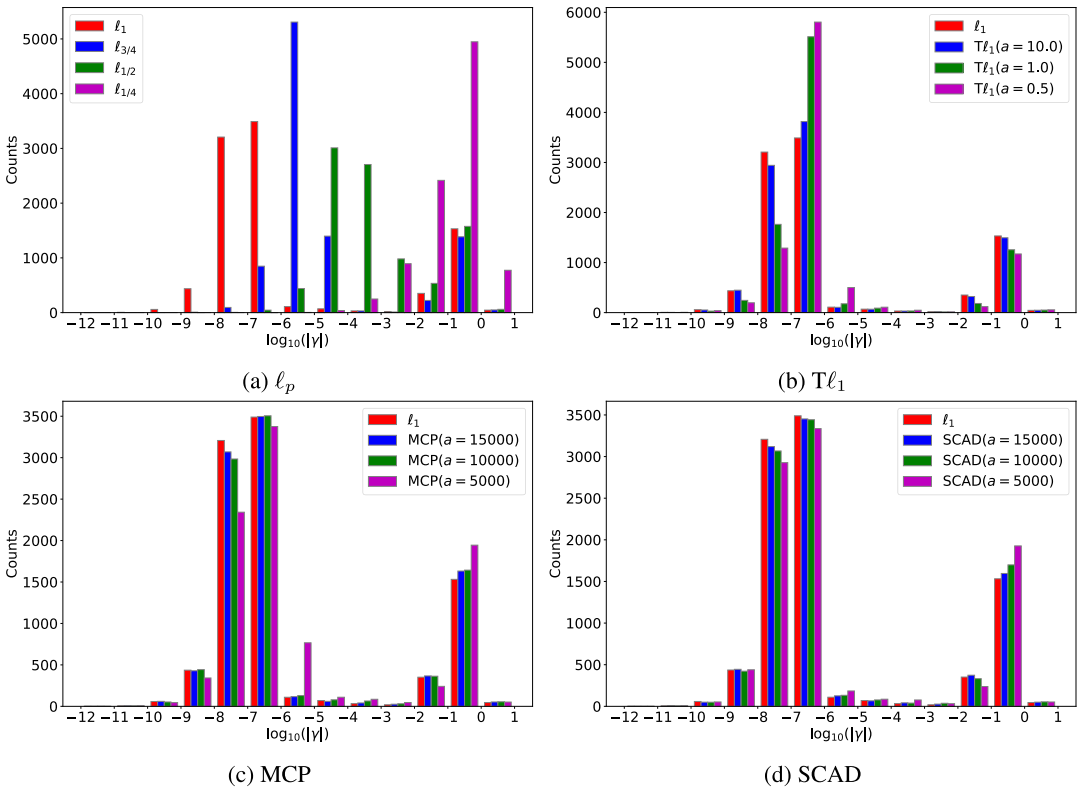


FIGURE 7. Histogram of scaling factors γ in DenseNet-40 trained on CIFAR 10. The x-axis is $\log_{10}(|\gamma|)$.

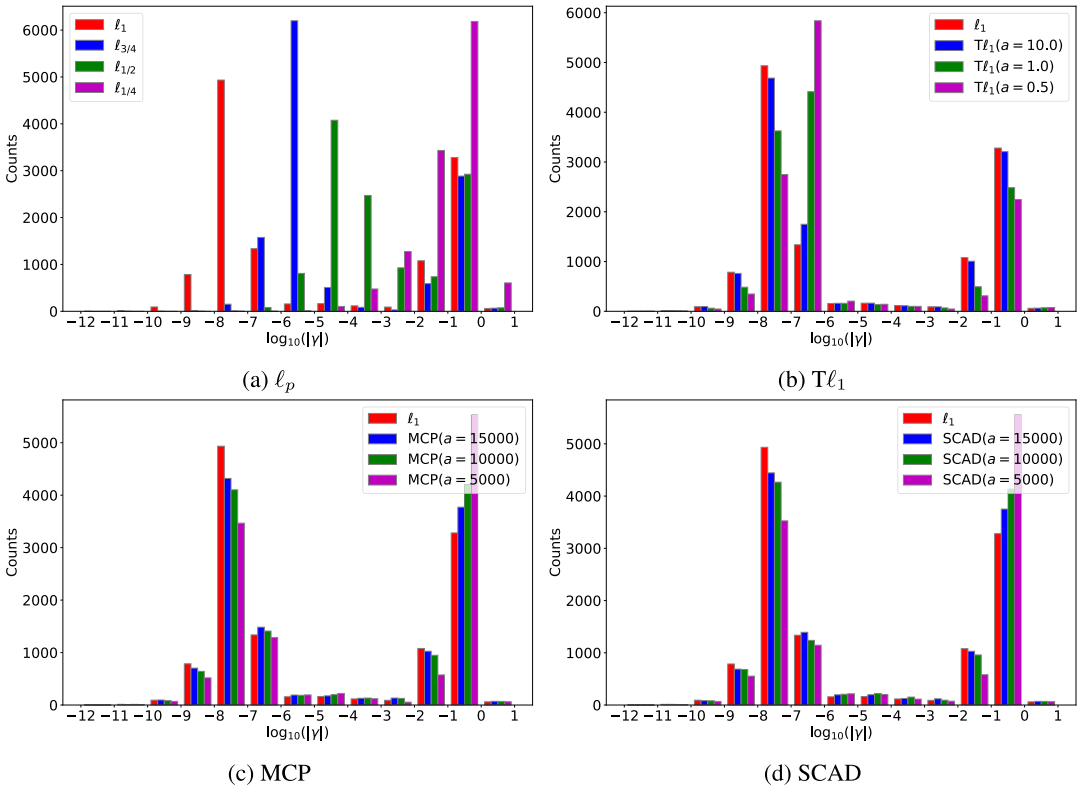


FIGURE 8. Histogram of scaling factors γ in ResNet-164 trained on CIFAR 10. The x-axis is $\log_{10}(|\gamma|)$.

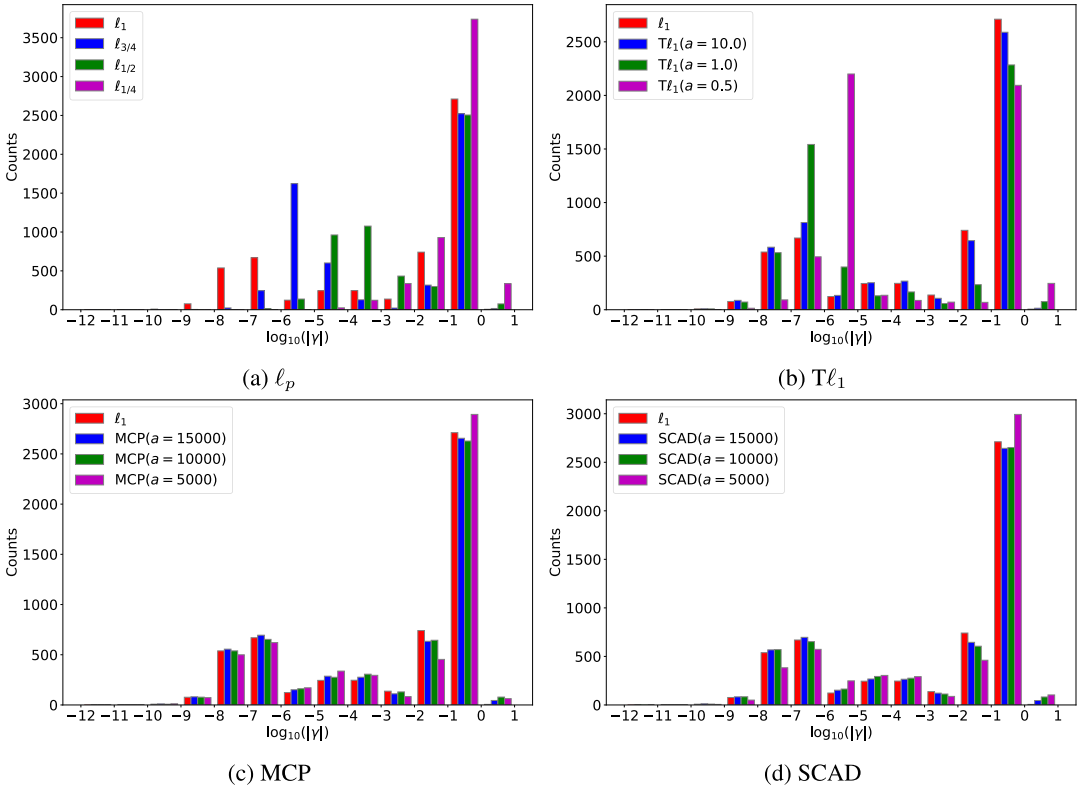
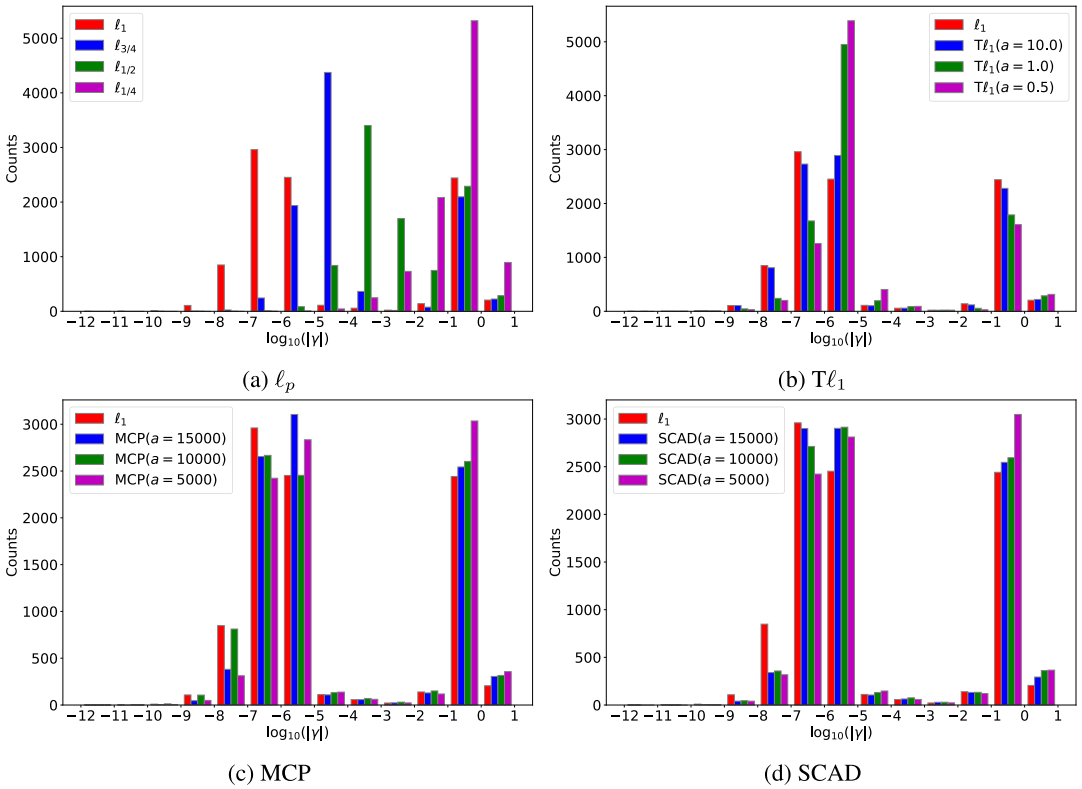
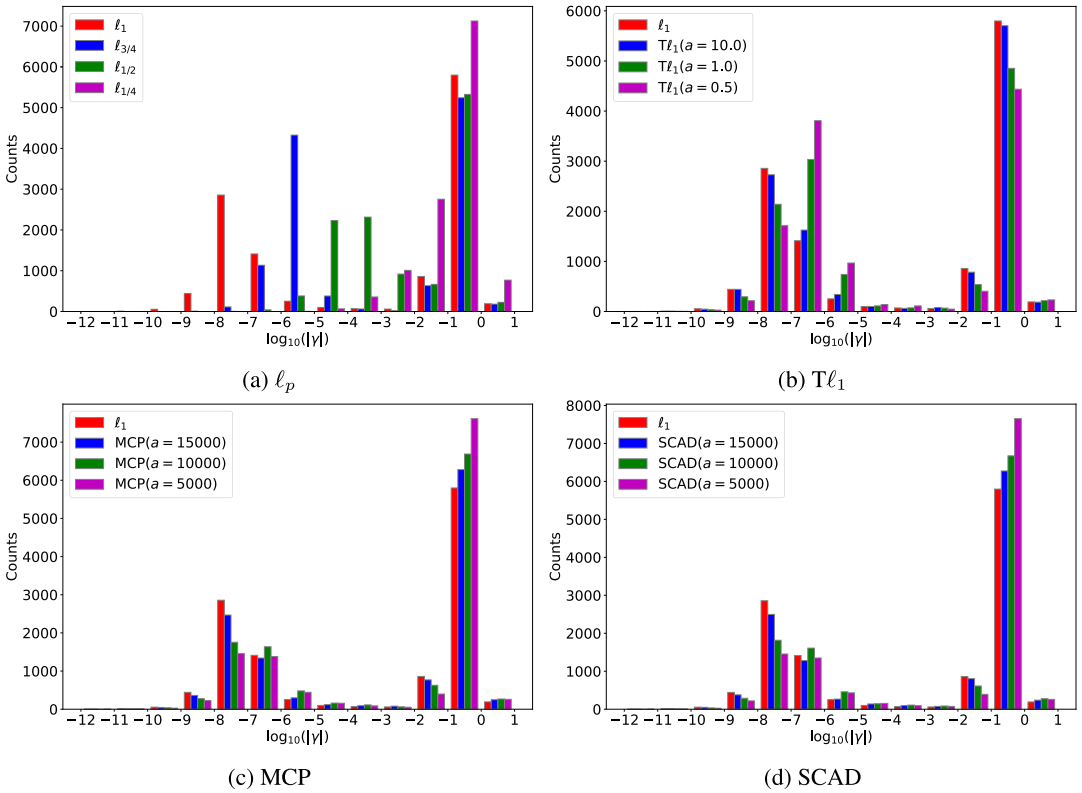


FIGURE 9. Histogram of scaling factors γ in VGG-19 trained on CIFAR 100. The x-axis is $\log_{10}(|\gamma|)$.



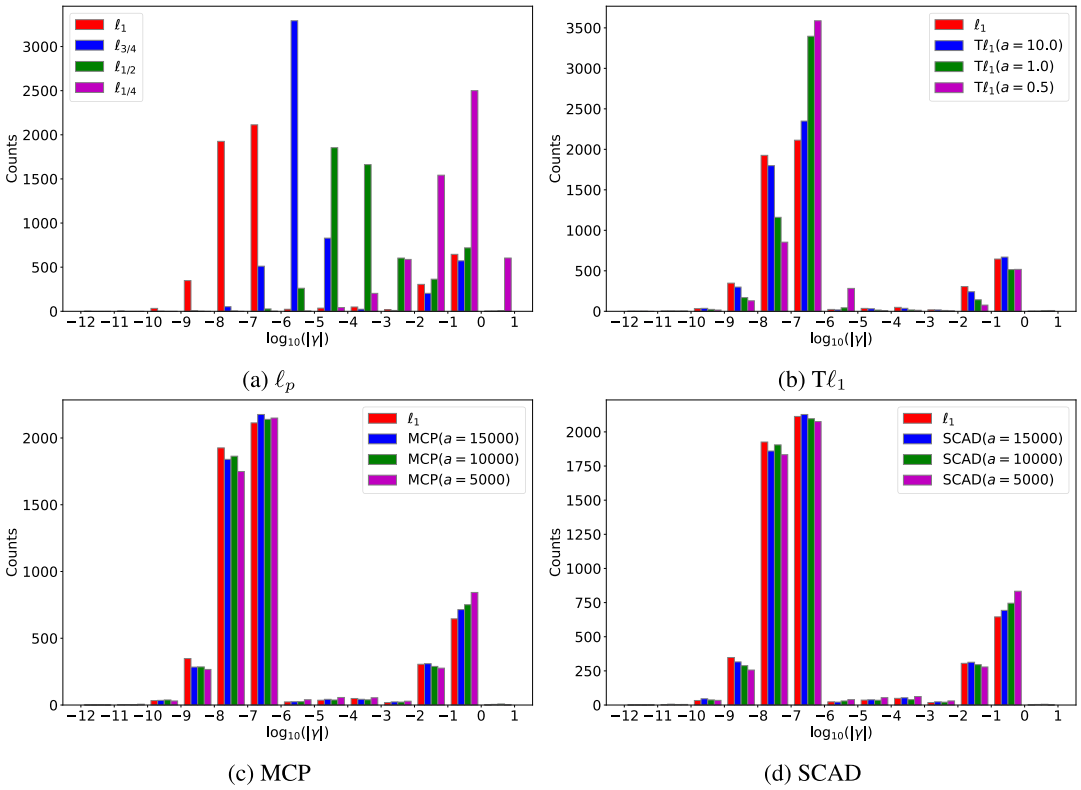
(a) ℓ_p (b) Tℓ_1 (c) MCP (d) SCAD

FIGURE 10. Histogram of scaling factors γ in DenseNet-40 trained on CIFAR 100. The x-axis is $\log_{10}(|\gamma|)$.



(a) ℓ_p (b) Tℓ_1 (c) MCP (d) SCAD

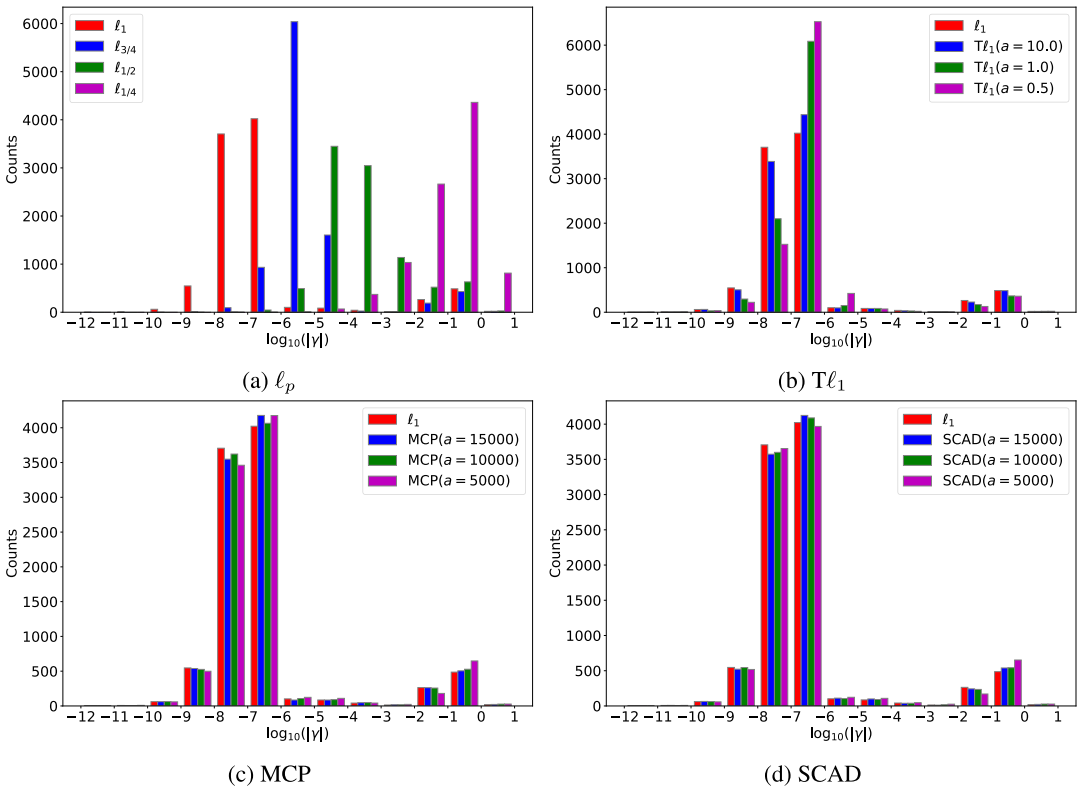
FIGURE 11. Histogram of scaling factors γ in ResNet-164 trained on CIFAR 100. The x-axis is $\log_{10}(|\gamma|)$.



(a) ℓ_p (b) Tℓ_1

(c) MCP (d) SCAD

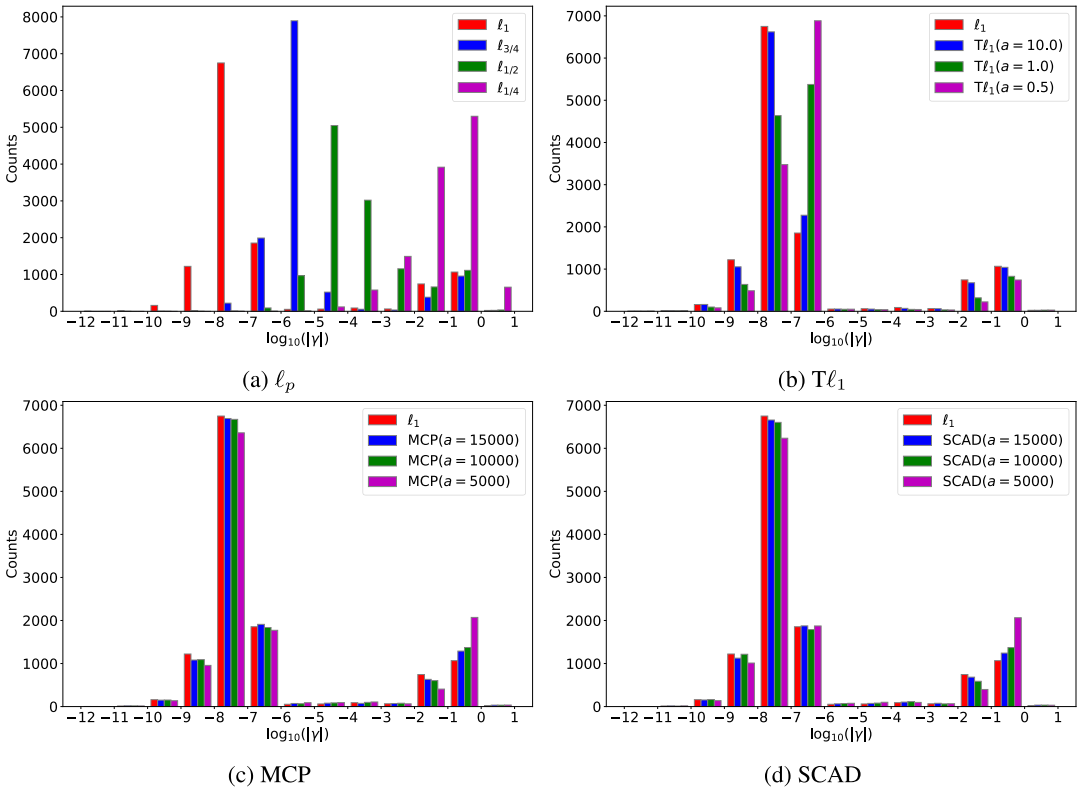
FIGURE 12. Histogram of scaling factors γ in VGG-19 trained on SVHN. The x-axis is $\log_{10}(|\gamma|)$.



(a) ℓ_p (b) Tℓ_1

(c) MCP (d) SCAD

FIGURE 13. Histogram of scaling factors γ in DenseNet-40 trained on SVHN. The x-axis is $\log_{10}(|\gamma|)$.



(a) ℓ_p (b) Tℓ_1

(c) MCP (d) SCAD

FIGURE 14. Histogram of scaling factors γ in ResNet-164 trained on SVHN. The x-axis is $\log_{10}(|\gamma|)$.

TABLE 8. Comparisons between network slimming with T$\ell_1(a = 0.5, 1.0)$ and variational channel pruning. The results are immediately obtained after channel pruning.

Model	Dataset	Method	Test Accuracy	Percentage of Channels Pruned	Percentage of Parameters Pruned
DenseNet-40	CIFAR 10	VCP [44]	93.16%	60%	59.67%
		T$\ell_1(a = 1.0)$	93.17%	60%	55.73%
		T$\ell_1(a = 1.0)$	93.17%	80%	74.46%
		T$\ell_1(a = 0.5)$	92.78%	60%	56.16%
		T$\ell_1(a = 0.5)$	92.78%	80%	74.88%
	CIFAR 100	VCP [44]	72.19%	37%	37.73%
		T$\ell_1(a = 1.0)$	72.63%	40%	36.91%
		T$\ell_1(a = 1.0)$	72.63%	60%	55.35%
		T$\ell_1(a = 0.5)$	72.57%	40%	36.98%
		T$\ell_1(a = 0.5)$	72.58%	60%	55.46%
ResNet-164	CIFAR 10	VCP [44]	93.16%	74%	56.70%
		T$\ell_1(a = 0.5)$	93.41%	75%	70.39%
	CIFAR 100	VCP [44]	73.76%	47%	17.59%
		T$\ell_1(a = 1.0)$	74.89%	45%	25.56%
		T$\ell_1(a = 0.5)$	74.72%	45%	27.74%

but again with less parameters pruned. Nevertheless, we can increase the percentage of channels pruned to 60% and the accuracy will remain the same with more parameters pruned.

On ResNet-164, with similar percentages of channels pruned, Tℓ_1 outperforms VCP by a large margin for both test accuracy and percentage of parameters pruned. For CIFAR 10, only T$\ell_1(a = 0.5)$ is able to have 75% of the channels pruned, and it saves more parameters by almost 24% with test accuracy better by 0.25%. For CIFAR 100%, with 2% less channels pruned, Tℓ_1 prunes at least 7.97% more parameters than VCP while having better accuracy of at least 0.96%.

Overall, network slimming with Tℓ_1 is competitive against the latest variant of network slimming.

V. CONCLUSION

We improve network slimming by replacing the ℓ_1 regularizer with a sparse, nonconvex regularizer for penalizing the scaling factors in the batch normalization layers. In particular, we investigate $\ell_p(0 < p < 1)$, Tℓ_1, MCP, and SCAD. We apply the proposed methods onto VGG-19, DenseNet-40, and ResNet-164 trained on CIFAR 10/100 and SVHN. We observe that ℓ_p and Tℓ_1 save more on parameters and FLOPs than ℓ_1 with a slight decrease in test accuracy.

In addition, Tℓ_1, especially $a = 0.5$, preserves model accuracy against channel pruning. Network slimming with Tℓ_1 is competitive against VCP, another network slimming variant robust against channel pruning. To attain better accuracy than ℓ_1 while having similar compression, MCP and SCAD perform the best job after their models are pruned and retrained, especially for VGG-19 and DenseNet-40.

For future directions, we plan to develop an optimization algorithm based on the relaxed variable splitting method [90] in order to use other nonconvex regularizers such as $\ell_1 - \alpha\ell_2$ [56], [91]. Additionally, we aim to generalize nonconvex network slimming to layer normalization [92] and group normalization [93]. Last, we will adapt nonconvex network slimming to the state-of-the-art compact CNNs, such as MobileNetv2 [94] and ShuffleNet [95].

ACKNOWLEDGMENT

The authors would like to thank the associate editor and the two anonymous referees for their careful reading and helpful feedback, which improved the presentation of the paper.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

[4] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2017.

[5] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 3431–3440.

[6] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.* Cham, Switzerland: Springer, 2015, pp. 234–241.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.

[8] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 7310–7311.

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[10] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1269–1277.

[11] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," 2014, *arXiv:1405.3866*. [Online]. Available: http://arxiv.org/abs/1405.3866

[12] W. Wen, C. Xu, C. Wu, Y. Wang, Y. Chen, and H. Li, "Coordinating filters for faster deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 658–666.

[13] Y. Xu, Y. Li, S. Zhang, W. Wen, B. Wang, Y. Qi, Y. Chen, W. Lin, and H. Xiong, "Trained rank pruning for efficient deep neural networks," 2018, *arXiv:1812.02402*. [Online]. Available: http://arxiv.org/abs/1812.02402

[14] Y. Xu, Y. Li, S. Zhang, W. Wen, B. Wang, Y. Qi, Y. Chen, W. Lin, and H. Xiong, "TRP: Trained rank pruning for efficient deep neural networks," in *Proc. Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 1–7.

[15] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2285–2294.

[16] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3123–3131.

[17] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," 2016, *arXiv:1605.04711*. [Online]. Available: http://arxiv.org/abs/1605.04711

[18] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," 2016, *arXiv:1612.01064*. [Online]. Available: http://arxiv.org/abs/1612.01064

[19] P. Yin, S. Zhang, J. Lyu, S. Osher, Y. Qi, and J. Xin, "BinaryRelax: A relaxation approach for training deep neural networks with quantized weights," *SIAM J. Imag. Sci.*, vol. 11, no. 4, pp. 2205–2223, Jan. 2018.

[20] A. Aghasi, A. Abdi, N. Nguyen, and J. Romberg, "Net-trim: Convex pruning of deep neural networks with performance guarantee," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3177–3186.

[21] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1135–1143.

[22] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient ConvNets," 2016, *arXiv:1608.08710*. [Online]. Available: http://arxiv.org/abs/1608.08710

[23] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," 2016, *arXiv:1607.03250*. [Online]. Available: http://arxiv.org/abs/1607.03250

[24] J. M. Alvarez and M. Salzmann, "Learning the number of neurons in deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2270–2278.

[25] S. Changpinyo, M. Sandler, and A. Zhmoginov, "The power of sparsity in convolutional neural networks," 2017, *arXiv:1702.06257*. [Online]. Available: http://arxiv.org/abs/1702.06257

[26] S. Scardapane, D. Comminiello, A. Hussain, and A. Uncini, "Group sparse regularization for deep neural networks," *Neurocomputing*, vol. 241, pp. 81–89, Jun. 2017.

[27] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2074–2082.

[28] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 2736–2744.

[29] J. Zhang, W. Wang, C. Lu, J. Wang, and A. K. Sangaiah, "Lightweight deep network for traffic sign classification," *Ann. Telecommun.*, vol. 75, nos. 7–8, pp. 369–379, Aug. 2020.

[30] H. Ma, T. Celik, and H.-C. Li, "Lightweight attention convolutional neural network through network slimming for robust facial expression recognition," *Signal, Image Video Process.*, pp. 1–9, Apr. 2021.

[31] W. He, M. Wu, M. Liang, and S.-K. Lam, "CAP: Context-aware pruning for semantic segmentation," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis.*, Jan. 2021, pp. 960–969.

[32] R. Chartrand, "Exact reconstruction of sparse signals via nonconvex minimization," *IEEE Signal Process. Lett.*, vol. 14, no. 10, pp. 707–710, Oct. 2007.

[33] R. Chartrand and W. Yin, "Iteratively reweighted algorithms for compressive sensing," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Mar. 2008, pp. 3869–3872.

[34] Z. Xu, X. Chang, F. Xu, and H. Zhang, "$L_{1/2}$ regularization: A thresholding representation theory and a fast solver," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 7, pp. 1013–1027, Jul. 2012.

[35] S. Zhang and J. Xin, "Minimization of transformed L_1 penalty: Closed form representation and iterative thresholding algorithms," *Commun. Math. Sci.*, vol. 15, no. 2, pp. 511–537, 2017.

[36] S. Zhang and J. Xin, "Minimization of transformed L_1 penalty: Theory, difference of convex function algorithm, and robust application in compressed sensing," *Math. Program.*, vol. 169, no. 1, pp. 307–336, 2018.

[37] C.-H. Zhang, "Nearly unbiased variable selection under minimax concave penalty," *Ann. Statist.*, vol. 38, no. 2, pp. 894–942, 2010.

[38] J. Fan and R. Li, "Variable selection via nonconcave penalized likelihood and its Oracle properties," *J. Amer. Statist. Assoc.*, vol. 96, no. 456, pp. 1348–1360, 2001.

[39] N. Z. Shor, *Minimization Methods for Non-Differentiable Functions*, vol. 3. Berlin, Germany: Springer, 2012.

[40] K. Bui, F. Park, S. Zhang, Y. Qi, and J. Xin, "Nonconvex regularization for network slimming: Compressing CNNs even more," in *Proc. Int. Symp. Vis. Comput.* Cham, Switzerland: Springer, 2020, pp. 39–53.

[41] C. Tai, T. Xiao, Y. Zhang, and X. Wang, "Convolutional neural networks with low-rank regularization," 2015, *arXiv:1511.06067*. [Online]. Available: http://arxiv.org/abs/1511.06067

[42] Y. Bai, Y.-X. Wang, and E. Liberty, "ProxQuant: Quantized neural networks via proximal operators," 2018, *arXiv:1810.00861*. [Online]. Available: http://arxiv.org/abs/1810.00861

[43] A. Aghasi, A. Abdi, and J. Romberg, "Fast convex pruning of deep neural networks," *SIAM J. Math. Data Sci.*, vol. 2, no. 1, pp. 158–188, Jan. 2020.

[44] C. Zhao, B. Ni, J. Zhang, Q. Zhao, W. Zhang, and Q. Tian, "Variational convolutional neural network pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 2780–2789.

[45] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *J. Roy. Statist. Soc. B, Statist. Methodol.*, vol. 68, no. 1, pp. 49–67, 2006.

[46] F. Xue and J. Xin, "Learning sparse neural networks via ℓ_0 and Tℓ_1 by a relaxed variable splitting method with application to multi-scale curve classification," in *Proc. World Congr. Global Optim.* Cham, Switzerland: Springer, 2019, pp. 800–809.

[47] R. Ma, J. Miao, L. Niu, and P. Zhang, "Transformed ℓ_1 regularization for learning sparse deep neural networks," *Neural Netw.*, vol. 119, pp. 286–298, Nov. 2019.

[48] M. K. Pandit, R. Naaz, and M. A. Chishti, "Learning sparse neural networks using non-convex regularization," *IEEE Trans. Emerg. Topics Comput. Intell.*, early access, Mar. 8, 2021, doi: 10.1109/TETCI.2021.3058672.

[49] K. Bui, F. Park, S. Zhang, Y. Qi, and J. Xin, "Structured sparsity of convolutional neural networks via nonconvex sparse group regularization," *Frontiers Appl. Math. Statist.*, vol. 6, p. 62, Feb. 2021.

[50] Y. Li, S. Gu, C. Mayer, L. Van Gool, and R. Timofte, "Group sparsity: The Hinge between filter pruning and decomposition for network compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020.

[51] E. J. Candés, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Commun. Pure Appl. Math.*, vol. 59, no. 8, pp. 1207–1223, 2006.

[52] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inf. Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.

[53] W. Yin, S. Osher, D. Goldfarb, and J. Darbon, "Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing," *SIAM J. Imag. Sci.*, vol. 1, no. 1, pp. 143–168, 2008.

[54] H. Jung, J. C. Ye, and E. Y. Kim, "Improved *k–t* BLAST and *k–t* SENSE using FOCUSS," *Phys. Med. Biol.*, vol. 52, no. 11, p. 3201, 2007.

[55] M. Lustig, D. Donoho, and J. M. Pauly, "Sparse MRI: The application of compressed sensing for rapid MR imaging," *Magn. Reson. Med., Off. J. Int. Soc. Magn. Reson. Med.*, vol. 58, no. 6, pp. 1182–1195, 2007.

[56] Y. Lou, P. Yin, Q. He, and J. Xin, "Computing sparse representation in a highly coherent dictionary based on difference of L_1 and L_2," *J. Sci. Comput.*, vol. 64, no. 1, pp. 178–196, 2015.

[57] Y. Lou, S. Osher, and J. Xin, "Computational aspects of constrained L_1-L_2 minimization for compressive sensing," in *Modelling, Computation and Optimization in Information Systems and Management Sciences*. Cham, Switzerland: Springer, 2015, pp. 169–180.

[58] R. Chartrand and V. Staneva, "Restricted isometry properties and nonconvex compressive sensing," *Inverse Problems*, vol. 24, no. 3, 2008, Art. no. 035020.

[59] Z. Xu, H. Guo, Y. Wang, and Z. Hai, "Representative of $L_{1/2}$ regularization among L_q $(0 \leq q \leq 1)$ regularizations: An experimental study based on phase diagram," *Acta Autom. Sinica*, vol. 38, no. 7, pp. 1225–1228, 2012.

[60] D. Krishnan and R. Fergus, "Fast image deconvolution using hyper-Laplacian priors," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1033–1041.

[61] W. Cao, J. Sun, and Z. Xu, "Fast image deconvolution using closed-form thresholding formulas of $L_q(q = 1/2, 2/3)$ regularization," *J. Vis. Commun. Image Represent.*, vol. 24, no. 1, pp. 31–41, 2013.

[62] Y. Qian, S. Jia, J. Zhou, and A. Robles-Kelly, "Hyperspectral unmixing via $L_{1/2}$ sparsity-constrained nonnegative matrix factorization," *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 11, pp. 4282–4297, Nov. 2011.

[63] C. Miao and H. Yu, "A general-thresholding solution for $l_p(0 < p < 1)$ regularized CT reconstruction," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 5455–5468, Dec. 2015.

[64] Y. Li, C. Wu, and Y. Duan, "The TVp regularized Mumford–Shah model for image labeling and segmentation," *IEEE Trans. Image Process.*, vol. 29, pp. 7061–7075, 2020.

[65] T. Wu, J. Shao, X. Gu, M. K. Ng, and T. Zeng, "Two-stage image segmentation based on nonconvex $\ell_2 - \ell_p$ approximation and thresholding," *Appl. Math. Comput.*, vol. 403, Aug. 2021, Art. no. 126168.

[66] J. Fan and H. Peng, "Nonconcave penalized likelihood with a diverging number of parameters," *Ann. Statist.*, vol. 32, no. 3, pp. 928–961, Jun. 2004.

[67] J. Lv and Y. Fan, "A unified approach to model selection and sparse recovery using regularized least squares," *Ann. Statist.*, vol. 37, no. 6A, pp. 3498–3528, 2009.

[68] S. Zhang, P. Yin, and J. Xin, "Transformed Schatten-1 iterative thresholding algorithms for low rank matrix completion," *Commun. Math. Sci.*, vol. 15, no. 3, pp. 839–862, 2017.

[69] J. You, Y. Jiao, X. Lu, and T. Zeng, "A nonconvex model with minimax concave penalty for image restoration," *J. Sci. Comput.*, vol. 78, no. 2, pp. 1063–1086, Feb. 2019.

[70] Z.-F. Jin, Z. Wan, Y. Jiao, and X. Lu, "An alternating direction method with continuation for nonconvex low rank minimization," *J. Sci. Comput.*, vol. 66, no. 2, pp. 849–869, 2016.

[71] A. Mehranian, H. S. Rad, A. Rahmim, M. R. Ay, and H. Zaidi, "Smoothly clipped absolute deviation (SCAD) regularization for compressed sensing MRI using an augmented Lagrangian scheme," *Magn. Reson. Imag.*, vol. 31, no. 8, pp. 1399–1411, Oct. 2013.

[72] P. Breheny and J. Huang, "Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection," *Ann. Appl. Statist.*, vol. 5, no. 1, p. 232, 2011.

[73] L. Wang, G. Chen, and H. Li, "Group SCAD regression analysis for microarray time course gene expression data," *Bioinformatics*, vol. 23, no. 12, pp. 1486–1494, Jun. 2007.

[74] G. Gu, S. Jiang, and J. Yang, "A TVSCAD approach for image deblurring with impulsive noise," *Inverse Problems*, vol. 33, no. 12, Dec. 2017, Art. no. 125008.

[75] A. Antoniadis and J. Fan, "Regularization of wavelet approximations," *J. Amer. Stat. Assoc.*, vol. 96, no. 455, pp. 939–967, Sep. 2001.

[76] M. Ahn, J.-S. Pang, and J. Xin, "Difference-of-convex learning: Directional stationarity, optimality, and sparsity," *SIAM J. Optim.*, vol. 27, no. 3, pp. 1637–1665, Jan. 2017.

[77] F. Wen, L. Chu, P. Liu, and R. C. Qiu, "A survey on nonconvex regularization-based sparse and low-rank recovery in signal processing, statistics, and machine learning," *IEEE Access*, vol. 6, pp. 69883–69906, 2018.

[78] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[79] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 2818–2826.

[80] J.-P. Penot, *Calculus Without Derivatives*, vol. 266. New York, NY, USA: Springer, 2012.

[81] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.

[82] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 646–661.

[83] M. Lin, Q. Chen, and S. Yan, "Network in network," 2013, *arXiv:1312.4400*. [Online]. Available: http://arxiv.org/abs/1312.4400

[84] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1319–1327.

[85] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS Workshop Deep Learn. Unsupervised Feature Learn.*, 2011, pp. 1–9.

[86] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 4700–4708.

[87] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1139–1147.

[88] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1026–1034.
[89] C. C. Aggarwal, *Neural Networks and Deep Learning*. Cham, Switzerland: Springer, 2018.
[90] T. Dinh and J. Xin, "Convergence of a relaxed variable splitting method for learning sparse neural networks via ℓ_1,ℓ_0, and transformed-ℓ_1 penalties," in *Proc. SAI Intell. Syst. Conf.* Cham, Switzerland: Springer, 2020, pp. 360–374.
[91] P. Yin, Y. Lou, Q. He, and J. Xin, "Minimization of ℓ_{1-2} for compressed sensing," *SIAM J. Sci. Comput.*, vol. 37, no. 1, pp. A536–A563, 2015.
[92] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*. [Online]. Available: http://arxiv.org/abs/1607.06450
[93] Y. Wu and K. He, "Group normalization," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 3–19.
[94] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.
[95] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6848–6856.

KEVIN BUI received the B.S. degree in mathematics of computation from the University of California at Los Angeles, in 2015, the M.S. degree in industrial engineering and management sciences from Northwestern University, in 2016, and the M.S. degree in mathematics from the University of California at Irvine, in 2021, where he is currently pursuing the Ph.D. degree in nonconvex optimization for image processing and deep learning. From 2016 to 2018, he was a Research Technologist specializing in healthcare engineering at the Center for Engineering Health, Northwestern University. His research interests include healthcare engineering, image processing, and deep learning.

FREDRICK PARK received the Ph.D. degree in applied mathematics from the University of California at Los Angeles, in 2006. His thesis work is in image processing, specializing in total variation and duality for blind image deconvolution, staircase reduction, and texture extraction. He worked as a Postdoctoral Scholar with the University of Michigan, Ann Arbor, and University of California at Irvine. His postdoctoral work included shape prior segmentation, shape modeling, and 3-D point cloud surface reconstruction. He is currently an Associate Professor of mathematics at Whittier College and a Lecturer at Paul Merage School of Business, University of California at Irvine. His current research interests include computer vision and machine learning.

SHUAI ZHANG received the Ph.D. degree in mathematics from UC Irvine. His research interests include numerical optimization, computer vision, and neural networks, especially on edge AI.

YINGYONG QI received the Ph.D. degree in speech and hearing sciences from Ohio State University, in 1989, and the Ph.D. degree in electrical and computer engineering from the University of Arizona, in 1993. He held a faculty position at the University of Arizona, from 1989 to 1999. He was a Visiting Scientist at the Research Laboratory of Electronics, Massachusetts Institute of Technology, from 1995 to 1996, and a Visiting Scientist at the Visual Computing Laboratory of Hewlett Packard, Palo Alto, in 1998. He is currently a Senior Director of technology at Qualcomm and a Researcher of the Department of Mathematics, University of California at Irvine. He has published over 100 scientific articles and U.S. patents during his tenure at university and industry. His research interests include speech processing, computer vision, and machine learning. He received Klatt Memorial Award in Speech Science from the Acoustical Society of America, in 1991, the First Award from the National Institute of Health, in 1992, and the AASFAA Outstanding Faculty Award from the University of Arizona, in 1998. More recently, he led a team winning the 3rd Place of IEEE Low Power Image Recognition Competition, sponsored by Google at CVPR 2019.

JACK XIN received the Ph.D. degree in mathematics from New York University's Courant Institute of Mathematical Sciences, in 1990. He was a Faculty at the University of Arizona, from 1991 to 1999, and the University of Texas at Austin, from 1999 to 2005. He is currently the Chancellor's Professor of mathematics at UC Irvine. His research interests include applied analysis and computational methods, and their applications in multi-scale problems and data science. He is a fellow of Guggenheim Foundation, American Mathematical Society, American Association for the Advancement of Science, and the Society of Industrial and Applied Mathematics. He was a recipient of Qualcomm Faculty Award (2019–2021).