# Training Sparse Neural Networks using Compressed Sensing


## Jonathan W. Siegel, Jianhong Chen, Jinchao Xu

## Department of Mathematics, The Pennsylvania State University


### Introduction

Pruning the weights of neural networks is an effective and widely-used technique for reducing model size and inference complexity. We develop and test a novel method based on compressed sensing which combines the pruning and training into a single step. Specifically, we utilize an **adaptively weighted $\ell^1$ penalty** on the weights during training, which we combine with momentum in order to train sparse neural networks. The adaptive weighting we introduce corresponds to a novel regularizer based on the logarithm of the absolute value of the weights. We perform a series of ablation studies demonstrating the improvement provided by the adaptive weighting and generalized RDA algorithm.
Furthermore, numerical experiments on the CIFAR-10, CIFAR-100, and ImageNet datasets demonstrate that our method

1) **trains sparser, more accurate networks than existing state-of-the-art methods. For example, we can use less than 1% of the parameters of VGG-19 to get 94.18% test accuracy;**
2) **can also be used effectively to obtain structured sparsity;**
3) **can be used to train sparse networks from scratch, i.e. from a random initialization, as opposed to initializing with a well-trained base model;**
4) **acts as an effective regularizer, improving generalization accuracy.**

## Methodology

For a neural network, we denote $\Theta$ as the collection of all parameters, $\mathcal{D}$ as the training dataset, and

$$L(\Theta) = \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} l(x, y, \Theta) \tag{1}$$

as the empirical loss function.

The lasso, which involves adding an $\ell^1$-norm regularization to the regression loss function, is a well-known and effective method for performing sparse regression and signal estimation in compressed sensing. In the context of neural network training, this corresponds to solving

$$\arg min_\Theta\, L(\Theta) + \lambda||\Theta||_1, \tag{2}$$

where $\lambda$ is a hyperparameter controlling the trade-off between sparsity and training loss. However, it doesn't generate sparse iterates since the soft-thresholding parameter is very small and constant for all network parameters.

It can be considerably improved by using an adaptive $\ell^1$ weight. We denote the groups of parameters $G_1, \dots, G_N$ where each group $G_i$ is either weights $W$ or bias $b$ from a convolutional or linear layer, or is shifts $\tilde{\beta}$ or scale parameters $\gamma$ from a batch normalization layer. Here $N$ is the total number of groups. Then we weight the $\ell^1$-norm on a parameter $\theta \in G_i$ with

$$\lambda(\beta+1)(\beta + \frac{|\theta|}{M_i})^{-1}, \tag{3}$$

where $\beta$ and $\lambda$ are hyperparameters and $M_i$ is the maximum absolute value of all parameters in $G_i$, i.e. $M_i = max_{\theta\in G_i}|\theta|$. In particular, we consider a running average of the absolute values of each parameter, computed recursively according to

$$\tilde{\theta}_n^i = \mu\tilde{\theta}_{n-1}^i + (1-\mu)|\theta_n^i|. \tag{4}$$

Here $\mu$ is a momentum parameter which effectively controls the number of iterations over which we average.

The momentum is also included in the algorithm by replacing the sampled gradient $\tilde{\nabla} L(\Theta)$ by an average over the past gradients.

$$v_n = \mu v_n + (1-\mu)\tilde{\nabla} L(\Theta_n). \tag{5}$$

We can generalize it to structured sparsity by simply replacing $M_i = max_{\theta\in G_i}|\theta|$ with $M_i = max_{k_j\in G_i}|k_j|$ where the enumeration is over all the kernels for kernel sparsity and channels for channel sparsity.

## Results

We provide experimental evidence demonstrating the effectiveness of our compressed sensing based approach to training sparse neural networks.

Table 1: Unstructured sparsity results on CIFAR-10.

| Model | Algorithm | Base Top1 | Sparse Top1 | Dense / Sparse Parameters | Compression Ratio | Non-Zero Fraction |
|---|---|---|---|---|---|---|
| **ResNet-18** | **CS** | **95.05** | **94.49** | **11.17M / 0.14M** | **81x** | **1.23** |
| ResNet-18 | RDA He et al. (2018) | - | 93.95 | 11.17M / 0.56M | 20x | 5.00 |
| **ResNet-20** | **CS** | **93.87** | **91.99** | **270K / 27K** | **10x** | **9.88** |
| ResNet-20 | Bayesian Deng et al. (2019) | 93.90 | 91.68 | 270K / 27K | 10x | 10.00 |
| **VGG-16** | **CS** | **93.79** | **94.13** | **14.73M / 0.18M** | **80x** | **1.25** |
| VGG-16 | Momentum Dettmers & Zettlemoyer (2019) | 93.41 | 93.31 | 14.73M / 0.74M | 20x | 5.00 |
| VGG-16 | Bayesian Louizos et al. (2017) | 91.60 | 91.00 | 14.73M / 0.81M | 18x | 5.50 |
| VGG-16 | Var Dropout Molchanov et al. (2017) | 92.70 | 92.70 | 14.73M / 0.31M | 48x | 2.08 |
| VGG-16 | Slimming Liu et al. (2017) | 93.66 | 93.41 | 14.73M / 0.65M | 22x | 4.40 |
| VGG-16 | DST Junjie et al. (2019) | 93.74 | 93.36 | 14.73M / 1.4M | 10x | 10.00 |
| VGG-16 | DST Junjie et al. (2019) | 93.74 | 93.00 | 14.73M / 0.74M | 20x | 5.00 |
| **VGG-19** | **CS** | **93.60** | **94.18** | **20.04M / 0.19M** | **104x** | **0.97** |
| VGG-19 | Pruning Han et al. (2015b) | 93.50 | 93.34 | 20.04M / 1.00M | 20x | 5.00 |
| VGG-19 | Scratch-B Liu et al. (2018) | 93.50 | 93.63 | 20.04M / 1.00M | 20x | 5.00 |

Table 2: Unstructured sparsity results on CIFAR-100.

| Model | Algorithm | Base Top1 | Sparse Top1 | Dense / Sparse Parameters | Compression Ratio | Non-Zero Fraction |
|---|---|---|---|---|---|---|
| **VGG-19** | **CS** | **73.83** | **75.93** | **20.09M / 0.46M** | **43x** | **2.30** |
| VGG-19 | Pruning Han et al. (2015b) | 71.70 | 70.22 | 20.09M / 1.00M | 20x | 5.00 |
| VGG-19 | Scratch-B Liu et al. (2018) | 71.70 | 72.08 | 20.09M / 1.00M | 20x | 5.00 |

Table 3: Results for structured (kernel) pruning on CIFAR-10.

| Model | Algorithm | Base Top1 | Sparse Top1 | Dense / Sparse Parameters | Compression Ratio | Non-Zero Fraction |
|---|---|---|---|---|---|---|
| **VGG-16** | **CS** | **93.79** | **94.24** | **14.73M / 0.57M** | **26x** | **3.84** |
| VGG-16 | Filter pruning Li et al. (2016) | 93.25 | 93.40 | 14.73M / 5.30M | 3x | 36.00 |
| VGG-16 | Scratch-B Liu et al. (2018) | 93.63 | 93.78 | 14.73M / 5.30M | 3x | 36.00 |

Table 4: Results for structured (channel) pruning on CIFAR-10.

| Model | Algorithm | Base Top1 | Sparse Top1 | Dense / Sparse Parameters | Compression Ratio | Non-Zero Parameters Fraction | Non-Zero Channels Fraction |
|---|---|---|---|---|---|---|---|
| **VGG-19** | **CS** | **93.60** | **93.87** | **20.04M / 1.35M** | **15x** | **6.73** | **22.34** |
| VGG-19 | Slimming Liu et al. (2017) | 93.66 | 93.80 | 20.04M / 2.30M | 9x | 11.5 | 30.00 |
| VGG-19 | Scratch-B Liu et al. (2018) | 93.53 | 93.81 | 20.04M / - | - | - | 30.00 |
| **DenseNet-40** | **CS** | **94.19** | **94.10** | **1.09M / 428K** | **3x** | **39.35** | **36.66** |
| DenseNet-40 | Scratch-B Liu et al. (2018) | 94.10 | 93.85 | 1.09M / - | - | - | 40.00 |

Table 5: Results for structured (channel) pruning on CIFAR-100.

| Model | Algorithm | Base Top1 | Sparse Top1 | Dense / Sparse Parameters | Compression Ratio | Non-Zero Parameters Fraction | Non-Zero Channels Fraction |
|---|---|---|---|---|---|---|---|
| **VGG-19** | **CS** | **73.83** | **74.64** | **20.10M / 4.06M** | **5x** | **20.20** | **44.72** |
| VGG-19 | Slimming Liu et al. (2017) | 73.26 | 73.48 | 20.08M / 5.00M | 4x | 24.9 | 50.00 |
| VGG-19 | Scratch-B Liu et al. (2018) | 72.63 | 73.08 | 20.04M / - | - | - | 50.00 |
| **DenseNet-40** | **CS** | **74.54** | **73.95** | **1.11M / 495K** | **2x** | **43.90** | **39.32** |
| DenseNet-40 | Scratch-B Liu et al. (2018) | 73.82 | 72.91 | 1.11M / - | - | - | 40.00 |

## Conclusion

1) We design an adaptively weighted $\ell^1$-regularization scheme which works well for training sparse neural networks. We connect this regularization scheme with a novel logarithmic regularizer, and also show how to adapt it to obtain structured sparsity.
2) In order to use this scheme to train sparse neural networks, we propose the use of a modified version of the regularized dual averaging (RDA) method which incorporates momentum.
3) We run a series of tests showing the effects of both the weighted $\ell^1$-norm and the RDA algorithm and its variants. In addition, we test the lottery ticket hypothesis on the final sparse structures obtained.
4) Experimental results indicate that, on a variety of datasets and architectures, our method trains networks which generalize better and are significantly sparser than existing state-of-the-art methods.

## Reference


Siegel, Jonathan W., Jianhong Chen, and Jinchao Xu. "Training Sparse Neural Networks using Compressed Sensing." arXiv preprint arXiv:2008.09661 (2020).