# SSGD: Sparsity-Promoting Stochastic Gradient Descent Algorithm for Unbiased DNN Pruning

Ching-Hua Lee*, Igor Fedorov†, Bhaskar D. Rao*, and Harinath Garudadri*

*Department of ECE, University of California, San Diego
†ARM ML Research

The 45th IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)
@ Barcelona, Spain
May 6, 2020

# Introduction

- While deep neural networks (DNNs) are powerful models for many engineering tasks, due to the lack of guidance on the network size, they are typically designed to be quite large.
- Over-parameterization of DNNs results in parameter redundancy, which, in turn, leads to inefficiency.
- Sparse signal recovery (SSR) techniques, on the other hand, find compact solutions to overcomplete linear problems.
- A logical step is to draw the connection between SSR and DNNs for learning sparse networks that require less computation and storage than the original, dense network.

# Problem Formulation

- We consider the empirical risk minimization problem:

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, D) = \frac{1}{N} \sum_{n=0}^{N-1} L\left(h(\mathbf{x}_n; \boldsymbol{\theta}), \mathbf{y}_n\right). \tag{1}$$

  - $\boldsymbol{\theta}$: the parameter set of the hypothesis $h(\cdot; \boldsymbol{\theta})$, e.g., a DNN
  - $D$=$\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=0}^{N-1}$: a dataset of $N$ input-output pairs
  - $J(\cdot, \cdot)$: the empirical risk objective function
  - $L(\cdot, \cdot)$: the loss function

* For a DNN with $M$ network parameters (weights and biases), we treat $\boldsymbol{\theta} = [\theta_0, \theta_1, ..., \theta_{M-1}]^T$ as a vector consisting of them all.

# Stochastic Gradient Descent (SGD) for DNN Training

- In DNN training, the SGD is widely used for learning the model parameters:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t, d_t). \tag{2}$$

  - $t$: the timestep
  - $\eta > 0$: the learning rate
  - $d_t$: a subset (mini-batch) of the training set $D$ given to the model at timestep $t$
  - $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t, \cdot)$: the gradient of the objective function w.r.t. $\boldsymbol{\theta}$ evaluated at timestep $t$

* We will omit the dependency of $J(\cdot, \cdot)$ on the data for brevity.

# Proposed Sparsity Regularization Framework

- A regularization term $G(\boldsymbol{\theta})$, serving as a *diversity measure* of $\boldsymbol{\theta}$, is added to the empirical risk $J(\boldsymbol{\theta})$ to promote sparse solutions $\boldsymbol{\theta}$:

$$\min_{\boldsymbol{\theta}} \quad J^G(\boldsymbol{\theta}) \triangleq J(\boldsymbol{\theta}) + \lambda G(\boldsymbol{\theta}), \tag{3}$$

  where $\lambda$ is the sparsity regularization coefficient.

- Commonly used in SSR is a *separable* diversity measure that has the form $G(\boldsymbol{\theta}) = \sum_{i=0}^{M-1} g(\theta_i)$, where $g(\cdot)$ has the following properties [Wipf and Nagarajan, 2010]:

  **Property 1:** $g(z)$ is symmetric, i.e., $g(z) = g(-z) = g(|z|)$;
  **Property 2:** $g(|z|)$ is monotonically increasing with $|z|$;
  **Property 3:** $g(0)$ is finite;
  **Property 4:** $g(z)$ is strictly concave in $|z|$ or $z^2$.

  Any function that holds the above properties is a candidate for effective SSR algorithms.

# Iterative Reweighted $\ell_2$ and $\ell_1$ Frameworks

- The iterative reweighted $\ell_2$ [Gorodnitsky and Rao, 1997, Chartrand and Yin, 2008] and $\ell_1$ [Candès et al., 2008] frameworks respectively suggest the following problems be solved at timestep (iteration) $t$:

$$\begin{aligned} \min_{\boldsymbol{\theta}} \quad & J_t^{\ell_2}(\boldsymbol{\theta}) \triangleq J(\boldsymbol{\theta}) + \lambda \left\| \boldsymbol{\Omega}_t^{-1} \boldsymbol{\theta} \right\|_2^2 \\ \min_{\boldsymbol{\theta}} \quad & J_t^{\ell_1}(\boldsymbol{\theta}) \triangleq J(\boldsymbol{\theta}) + \lambda \left\| \boldsymbol{\Omega}_t^{-1} \boldsymbol{\theta} \right\|_1 \end{aligned}, \tag{4}$$

  where $\boldsymbol{\Omega}_t = \text{diag}\{\omega_{i,t}\}$ is positive definite and each $\omega_{i,t}$ is a function of $\theta_{i,t}$, whose form depends on the choices of the reweighting framework and the diversity measure $G(\cdot)$.

- The reweighting methods belong to the majorization-minimization (MM) [Sun et al., 2017] family of algorithms: both $J_t^{\ell_2}(\boldsymbol{\theta})$ and $J_t^{\ell_1}(\boldsymbol{\theta})$ serve as a majorizer of $J^G(\boldsymbol{\theta})$. By iteratively minimizing the majorizers we can approach a solution to $\min_{\boldsymbol{\theta}} J^G(\boldsymbol{\theta})$.

# Examples of $G(\boldsymbol{\theta})$ and the Corresponding $\boldsymbol{\Omega}_t$

Table: Example diversity measures $G(\boldsymbol{\theta})$ and the corresponding $\boldsymbol{\Omega}_t$ updates.

| **Diversity measure** | $G(\boldsymbol{\theta})$ **function** $g(\theta_i) =$ | **Parameter range** | **Reweighting framework** | $\boldsymbol{\Omega}_t$ **update** $\omega_{i,t} =$ |
|---|---|---|---|---|
| $p$-norm-like | $\lvert\theta_i\rvert^p$ | $0 < p \leq 2$ | reweighted $\ell_2$ | $\left(\frac{2}{p}\lvert\theta_{i,t}\rvert^{2-p}\right)^{\frac{1}{2}}$ |
| $p$-norm-like | $\lvert\theta_i\rvert^p$ | $0 < p \leq 1$ | reweighted $\ell_1$ | $\frac{1}{p}\lvert\theta_{i,t}\rvert^{1-p}$ |
| log-sum | $\log(\theta_i^2 + \epsilon)$ | $\epsilon > 0$ | reweighted $\ell_2$ | $\left(\theta_{i,t}^2 + \epsilon\right)^{\frac{1}{2}}$ |
| log-sum | $\log(\lvert\theta_i\rvert + \epsilon)$ | $\epsilon > 0$ | reweighted $\ell_1$ | $\lvert\theta_{i,t}\rvert + \epsilon$ |

* In general, a smaller $p$ or $\epsilon$ promotes stronger sparsity.
* Practically, for the $p$-norm-like cases using $g(\theta_i) = \lvert\theta_i\rvert^p$, a small constant $c > 0$ is employed for avoiding possible algorithm stagnation and instability; i.e., substituting $\lvert\theta_{i,t}\rvert + c$ for $\lvert\theta_{i,t}\rvert$ in the $\omega_{i,t}$ updates.

# Affine Scaling Transformation (AST)

- Before proceeding, we propose the following reparameterization in terms of the (affinely) scaled variable $\mathbf{q}$:

$$\mathbf{q} \triangleq \boldsymbol{\Omega}_t^{-1}\boldsymbol{\theta}, \tag{5}$$

  in which $\boldsymbol{\Omega}_t$ is used as the *scaling matrix*.

- It can be interpreted as the AST commonly employed by the interior point approach to solving optimization problems [Nash and Sofer, 1996].

- It is well-known in the context of SSR that AST-based methods converge to sparse solutions [Rao and Kreutz-Delgado, 1999, Gorodnitsky and Rao, 1997].

# Reparameterize the Problems by AST

- We apply the AST to reparameterize $J_t^{\ell_2}(\boldsymbol{\theta})$ and $J_t^{\ell_1}(\boldsymbol{\theta})$ and perform minimization in the $\mathbf{q}$ domain, that is:

$$\begin{aligned} \min_{\mathbf{q}} \quad & J_t^{\ell_2}(\boldsymbol{\Omega}_t\mathbf{q}) = J(\boldsymbol{\Omega}_t\mathbf{q}) + \lambda\|\mathbf{q}\|_2^2 \\ \min_{\mathbf{q}} \quad & J_t^{\ell_1}(\boldsymbol{\Omega}_t\mathbf{q}) = J(\boldsymbol{\Omega}_t\mathbf{q}) + \lambda\|\mathbf{q}\|_1 \end{aligned}, \tag{6}$$

  for the $\ell_2$ and $\ell_1$ cases, respectively.

- To proceed, define the *a posteriori* AST variable at timestep $t$:

$$\mathbf{q}_{t|t} \triangleq \boldsymbol{\Omega}_t^{-1}\boldsymbol{\theta}_t \tag{7}$$

  and the *a priori* AST variable at timestep $t$:

$$\mathbf{q}_{t+1|t} \triangleq \boldsymbol{\Omega}_t^{-1}\boldsymbol{\theta}_{t+1}. \tag{8}$$

# The AST-Based Update Rules

- We formulate a recursive update by using SGD in the $\mathbf{q}$ domain:

$$\begin{aligned} \mathbf{q}_{t+1|t} &\leftarrow \mathbf{q}_{t|t} - \eta \nabla_{\mathbf{q}} J^{\ell_2}(\boldsymbol{\Omega}_t \mathbf{q}_{t|t}) \\ \mathbf{q}_{t+1|t} &\leftarrow \mathbf{q}_{t|t} - \eta \nabla_{\mathbf{q}} J^{\ell_1}(\boldsymbol{\Omega}_t \mathbf{q}_{t|t}) \end{aligned}, \tag{9}$$

  for the $\ell_2$ and $\ell_1$ cases, respectively.

- Using the chain rule and the AST relationships, the above can be shown to be equivalent to the following update rules in the $\boldsymbol{\theta}$ domain:

$$\begin{aligned} \boldsymbol{\theta}_{t+1} &\leftarrow \boldsymbol{\theta}_t - \eta \boldsymbol{\Omega}_t^2 \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t) - \lambda \eta 2 \boldsymbol{\theta}_t \\ \boldsymbol{\theta}_{t+1} &\leftarrow \boldsymbol{\theta}_t - \eta \boldsymbol{\Omega}_t^2 \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t) - \lambda \eta \boldsymbol{\Omega}_t \mathrm{sgn}(\boldsymbol{\theta}_t) \end{aligned}, \tag{10}$$

  for the $\ell_2$ and $\ell_1$ cases, respectively.

# The SSGD Algorithm Adopting $\lambda = 0$

- Setting $\lambda = 0$ in both the $\ell_2$ and $\ell_1$ update rules leads to:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta \boldsymbol{\Omega}_t^2 \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t). \tag{11}$$

  It is the Sparsity-promoting Stochastic Gradient Descent (SSGD).

- The term $\boldsymbol{\Omega}_t^2$ in (11) provides a weighting factor $\omega_{i,t}^2$ to the learning rate $\eta$ for updating the corresponding parameter $\theta_{i,t}$. Typically, $\omega_{i,t}^2$ is a function of $\left|\theta_{i,t}\right|$.

- It is similar to the proportionate normalized least mean square (PNLMS) algorithm [Duttweiler, 2000]. Instead of speeding up convergence like that in the PNLMS, the purpose of SSGD is to promote sparse solutions.

# Unbiasedness of SSGD

- By adopting $\lambda = 0$, the underlying optimization problem of SSGD is:

$$\min_{\mathbf{q}} J(\boldsymbol{\Omega}_t \mathbf{q}), \tag{12}$$

  which actually transforms the original problem $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ by applying a change of coordinates via the AST.

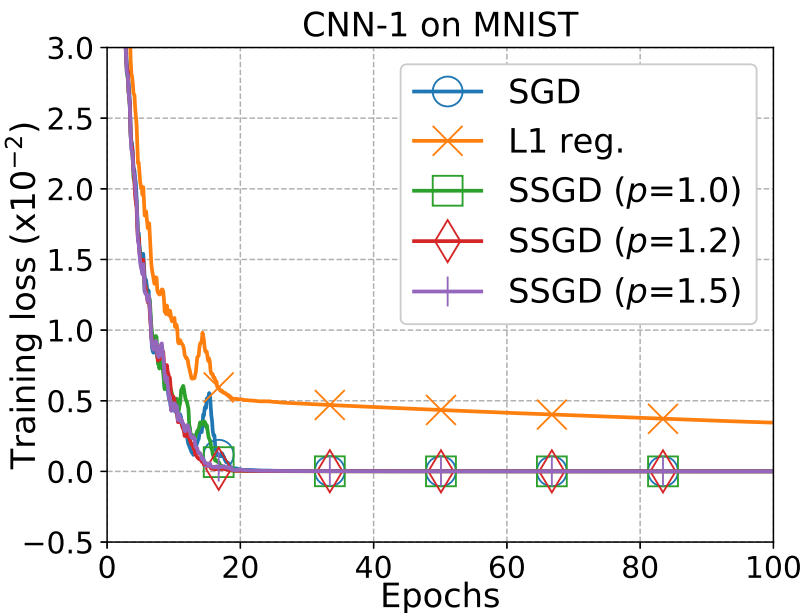
- Since $\boldsymbol{\Omega}_t$ is invertible, the problem of finding the $\boldsymbol{\theta}$ which minimizes $J(\boldsymbol{\theta})$ is equivalent to finding the $\mathbf{q}$ which minimizes $J(\boldsymbol{\Omega}_t \mathbf{q})$. Therefore, the solution of (12) is guaranteed to also be a solution of $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$, which is not true for $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) + \lambda G(\boldsymbol{\theta})$ with $\lambda > 0$. Therefore, the optimization (12) is *unbiased*.

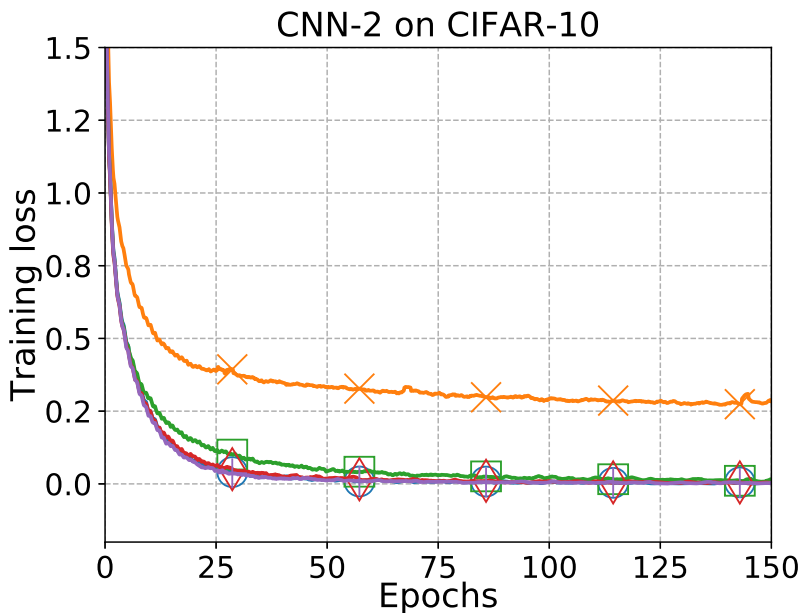
- Due to the sparsity-promoting ability of $\boldsymbol{\Omega}_t$, if there are multiple solutions to $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$, then iteratively solving $\min_{\mathbf{q}} J(\boldsymbol{\Omega}_t \mathbf{q})$ will tend to produce sparse choices of $\boldsymbol{\theta}$.

# Applying SSGD to DNN Compression

- [Han et al., 2015] have proposed a 3-stage compression scheme:
  - i) learning important connections (e.g., using $\ell_1$ regularization for sparsity)
  - ii) pruning unimportant parameters by hard thresholding based on magnitude
  - iii) fine-tuning the remaining ones to regain accuracy
- We adopt the same scheme, using SSGD in stage i).

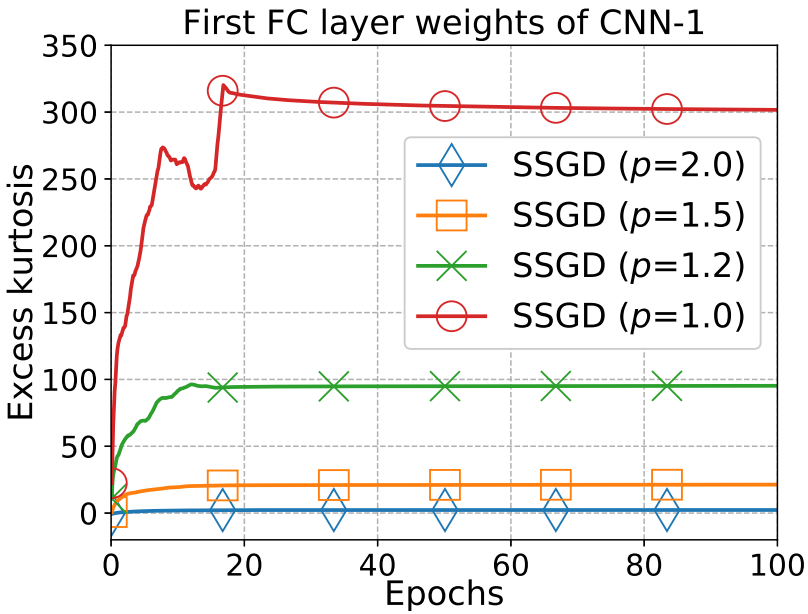
# Training Performance

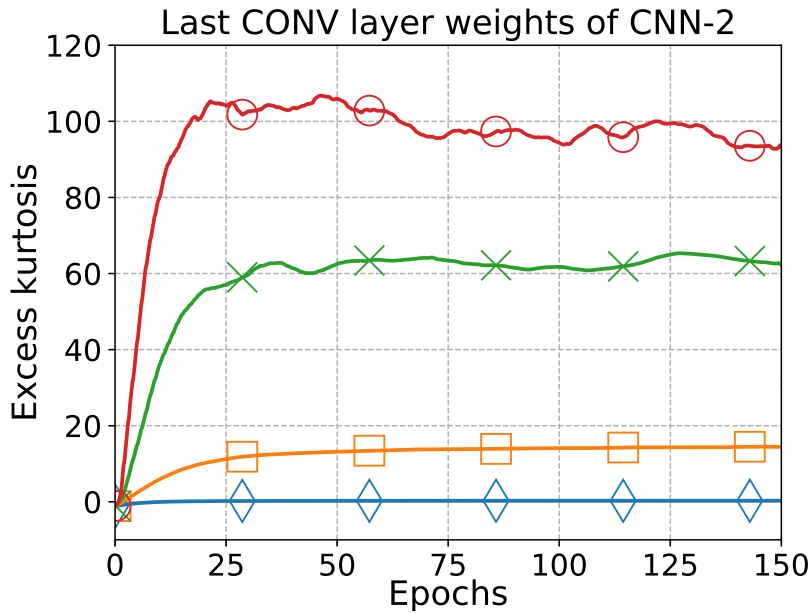



- SSGD is able to converge toward the same loss as SGD, supporting the argument that SSGD is *unbiased* as it finds solutions to $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$.
- The $\ell_1$ regularized case, however, ends up at a higher loss due to the bias introduced by a nonzero $\lambda$.

# Sparsity-Promoting Ability

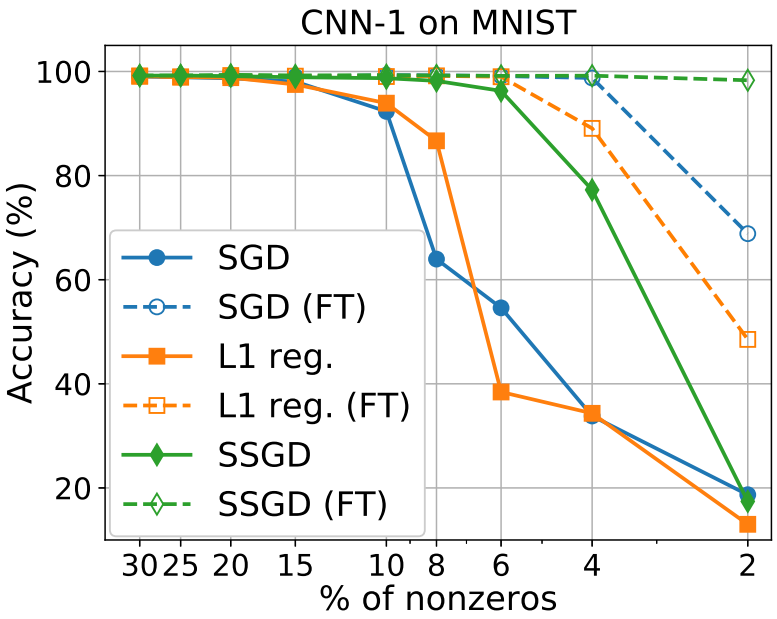


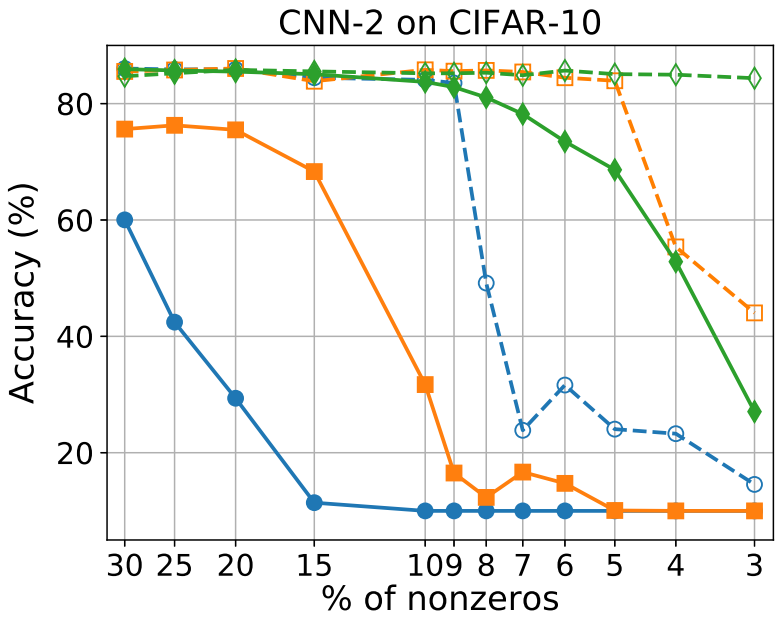




* Excess kurtosis serves as a measure of sparsity (the higher, the sparser). A Gaussian distribution has an excess kurtosis of 0.

- SSGD with a smaller $p$ results in greater sparsity. Note that when $p = 2$, SSGD reduces to the regular SGD, resulting in near 0 excess kurtosis.

# Pruning Results




* 'FT' stands for 'fine-tuned.'

- Training with SSGD for stage i), the test accuracy remains the highest after hard thresholding in stage ii) (solid lines), and regains most accuracy after fine-tuning in stage iii) (dashed lines).

# Comparison with Existing Pruning Approaches

Table: Comparison of sparsification results.

| Model | Method | Accuracy | % of nonzeros |
|---|---|---|---|
| MLP | Original | 98.62% | 100.0 |
| | Net-Trim [Aghasi et al., 2017] | 97.70% | 30.5 |
| | Iter. Reweight. [Jiang et al., 2019] | 97.46% | 14.8 |
| | Proposed | 98.39% | 3.7 |
| CNN-3 | Original | 77.44% | 100.0 |
| | Net-Trim [Aghasi et al., 2017] | 75.92% | 17.8 |
| | Iter. Reweight. [Jiang et al., 2019] | 74.17% | 7.9 |
| | Proposed | 74.54% | 5.1 |

* MLP: multi-layer perceptron
* Both MLP and CNN-3 are models used in [Jiang et al., 2019] on MNIST and CIFAR-10, respectively

# Conclusion

- We proposed SSGD, an unbiased learning algorithm for DNN pruning.
- The SSGD is based on the iterative reweighting frameworks and AST reparameterization, allowing the adoption of a zero regularization coefficient $\lambda$ while incorporating sparsity.
- The sparsification ability of SSGD has been demonstrated on image classification tasks and shown to outperform existing methods on the MNIST and CIFAR-10 datasets.

# Reference I


[Aghasi et al., 2017] Aghasi, A., Abdi, A., Nguyen, N., and Romberg, J. (2017).

Net-Trim: Convex pruning of deep neural networks with performance guarantee.

In *Adv. Neural Inform. Process. Syst. (NIPS)*, pages 3177–3186.

[Candès et al., 2008] Candès, E. J., Wakin, M. B., and Boyd, S. P. (2008).

Enhancing sparsity by reweighted $\ell_1$ minimization.

*J. Fourier Anal. Appl.*, 14(5):877–905.

[Chartrand and Yin, 2008] Chartrand, R. and Yin, W. (2008).

Iteratively reweighted algorithms for compressive sensing.

In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, pages 3869–3872.

[Duttweiler, 2000] Duttweiler, D. L. (2000).

Proportionate normalized least-mean-squares adaptation in echo cancelers.

*IEEE Trans. Speech Audio Process.*, 8(5):508–518.

# Reference II


[Gorodnitsky and Rao, 1997] Gorodnitsky, I. F. and Rao, B. D. (1997).

Sparse signal reconstruction from limited data using FOCUSS: A re-weighted minimum norm algorithm.

*IEEE Trans. Signal Process.*, 45(3):600–616.

[Han et al., 2015] Han, S., Pool, J., Tran, J., and Dally, W. J. (2015).

Learning both weights and connections for efficient neural networks.

In *Adv. Neural Inform. Process. Syst. (NIPS)*, pages 1135–1143.

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015).

Batch normalization: Accelerating deep network training by reducing internal covariate shift.

In *Proc. Int. Conf. Mach. Learn. (ICML)*, pages 448–456.

[Jiang et al., 2019] Jiang, T., Yang, X., Shi, Y., and Wang, H. (2019).

Layer-wise deep neural network pruning via iteratively reweighted optimization.

In *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Process. (ICASSP)*, pages 5606–5610.

# Reference III

[Krizhevsky and Hinton, 2009] Krizhevsky, A. and Hinton, G. (2009).
Learning multiple layers of features from tiny images.
Technical report, Univ. Toronto.

[LeCun et al., 1998] LeCun, Y., Cortes, C., and Burges, E. J. (1998).
*The MNIST Database of Handwritten Digits*.

[Nash and Sofer, 1996] Nash, S. G. and Sofer, A. (1996).
*Linear and Nonlinear Programming*.
McGraw-Hill Inc.

[Rao and Kreutz-Delgado, 1999] Rao, B. D. and Kreutz-Delgado, K. (1999).
An affine scaling methodology for best basis selection.
*IEEE Trans. Signal Process.*, 47(1):187–200.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014).
Dropout: A simple way to prevent neural networks from overfitting.
*J. Mach. Learn. Res.*, 15:1929–1958.

# Reference IV

[Sun et al., 2017] Sun, Y., Babu, P., and Palomar, D. P. (2017).

Majorization-minimization algorithms in signal processing, communications, and machine learning.

*IEEE Trans. Signal Process.*, 65(3):794–816.

[Wipf and Nagarajan, 2010] Wipf, D. and Nagarajan, S. (2010).

Iterative reweighted $\ell_1$ and $\ell_2$ methods for finding sparse solutions.

*IEEE J. Sel. Top. Signal Process.*, 4(2):317–329.

# Backup Slides

# Practical Considerations

- In practice, we find that normalizing the $\boldsymbol{\Omega}_t^2$ term helps stabilize SSGD. We thus propose the practical SSGD update rule:

$$\boxed{\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta \mathbf{S}_t \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t),} \tag{13}$$

where $\mathbf{S}_t = \text{diag}\{s_{i,t}\}$, referred to as the *sparsity-promoting matrix*, is the normalized version of $\boldsymbol{\Omega}_t^2$:

$$\boxed{s_{i,t} = \frac{\omega_{i,t}^2}{\frac{1}{|\mathcal{I}^{(k)}|} \sum_{j \in \mathcal{I}^{(k)}} \omega_{j,t}^2}, \quad \text{for } i \in \mathcal{I}^{(k)},} \tag{14}$$

where $\mathcal{I}^{(k)}$ denotes the index set of parameters of layer $k$, $\theta_{i,t}$ is in layer $k$, and $|\mathcal{I}^{(k)}|$ is the cardinality of $\mathcal{I}^{(k)}$.

# The SSGD Algorithm

- Algorithm 1 summarizes SSGD which can be implemented using standard deep learning libraries without much effort.

**Algorithm 1:** The proposed SSGD algorithm for learning sparse DNN connections. $\boldsymbol{\omega}_t$ and $\mathbf{s}_t$ denote the vectors consisting of the diagonal elements of $\boldsymbol{\Omega}_t$ and $\mathbf{S}_t$, respectively. $\odot$ denotes element-wise multiplication.

**Input:** $\eta$: learning rate, $d_t$: training data at timestep $t$, and the choice of the diversity measure
**Output:** $\boldsymbol{\theta}_t$: estimated model parameters
Initialize: $\boldsymbol{\theta}_0$
**for** $t = 0, 1, 2...$ **do**
    Compute scaling factors: $\boldsymbol{\omega}_t$ according to the specified diversity measure
    Compute sparsity-promoting factors: $\mathbf{s}_t$ by normalizing $\boldsymbol{\omega}_t$
    Update parameters: $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta \cdot \mathbf{s}_t \odot \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t, d_t)$
**end for**

# Applying SSGD to DNN Compression

- [Han et al., 2015] have proposed a 3-stage compression scheme:
  - i) learning important connections (e.g., using $\ell_1$ regularization for sparsity)
  - ii) pruning unimportant parameters by hard thresholding based on magnitude
  - iii) fine-tuning the remaining ones fine-tuning the remaining ones to regain accuracy

  * We adopt the same scheme, using SSGD in stage i).

- In [Han et al., 2015], it is observed that, $\ell_1$ regularization leads to sparser networks after stage i), but the network loses significant accuracy after stage ii), and is not able to recover from this accuracy drop even after stage iii).

- The authors posit that the discrepancy between using $\ell_1$ regularization during stage i) and not using it during stage iii) leads to poor performance. SSGD circumvents such issues because it finds (sparse) solutions to $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ directly.

# Experimental Setup

- *CNN-1 on MNIST database [LeCun et al., 1998]*: We define a model that has 2 CONV layers followed by 3 FC layers for this task. Max pooling is performed. A ReLU activation is adopted for all layers and we use cross-entropy for $J(\boldsymbol{\theta})$.

- *CNN-2 on CIFAR-10 database [Krizhevsky and Hinton, 2009]*: We define a more complicated model with 6 CONV layers followed by 3 FC layers for this task. Max pooling is performed. Batch normalization [Ioffe and Szegedy, 2015] and dropout [Srivastava et al., 2014] are appropriately used for showing their compatibility with SSGD. ReLU activation is adopted for all layers and we use cross-entropy for $J(\boldsymbol{\theta})$.

* CNN: convolutional neural network  CONV: convolutional
FC: fully-connected  ReLU: rectified linear unit