Supplemental Note: Net-Trim Implementation for Convolutional Layers

Alireza Aghasi* Afshin Abdi† Justin Romberg†

Abstract

This short note presents the details of implementing the the Net-Trim algorithm for convolutional layers. Net-Trim requires a least-squares solve at each iteration and the main idea is to employ the conjugate gradient (CG) method in operator form. This way the least squares step can be efficiently carried out via a series of forward/adjoint operations.

1 Net-Trim in Operator Form

For $\boldsymbol{W} \in \mathbb{R}^{N\times M}$, $\boldsymbol{X}^{in} \in \mathbb{R}^{N\times P}$, and $\Omega \subseteq \{1,\cdots,M\}\times\{1,\cdots,P\}$ the Net-Trim central program in the matrix form is

$$\underset{\boldsymbol{W}}{\text{minimize}} \ \ \|\boldsymbol{W}\|_1 \quad \text{subject to} \quad \begin{cases} \left\|(\boldsymbol{W}^\top \boldsymbol{X}^{in} - \boldsymbol{X}^{out})_\Omega\right\|_F \le \epsilon \\ (\boldsymbol{W}^\top \boldsymbol{X}^{in})_{\Omega^c} \le \boldsymbol{V}_{\Omega^c} \end{cases}. \tag{1}$$

When $\boldsymbol{X}^{in} \in \mathbb{T}_{in}$, $\boldsymbol{W} \in \mathbb{T}_w$ and $\boldsymbol{X}^{out} \in \mathbb{T}_{out}$ are tensors, and Ω is a set of indices on the tensor elements, our central program takes the following form:

$$\underset{\boldsymbol{W}}{\text{minimize}} \ \ \|\boldsymbol{W}\|_1 \quad \text{subject to} \quad \begin{cases} \left\|(\mathcal{A}_{\boldsymbol{X}^{in}}(\boldsymbol{W}) - \boldsymbol{X}^{out})_\Omega\right\|_F \le \epsilon \\ (\mathcal{A}_{\boldsymbol{X}^{in}}(\boldsymbol{W}))_{\Omega^c} \le \boldsymbol{V}_{\Omega^c} \end{cases}, \tag{2}$$

where $\|\cdot\|_1$ and $\|\cdot\|_F$ naturally apply to the vectorized tensors. To clarify the notation, given a tensor $\boldsymbol{Z}$ and an index set Ω, $\boldsymbol{Z}_\Omega$ is a tensor of similar size as $\boldsymbol{Z}$, which takes identical values as $\boldsymbol{Z}$ on Ω and takes zero values on Ω^c.

The operator $\mathcal{A}_{\boldsymbol{X}^{in}} : \mathbb{T}_w \to \mathbb{T}_{out}$ is a linear operator that is parameterized by $\boldsymbol{X}^{in}$. For instance in convolutional layers it is a tensor convolution operator with one of the operands being $\boldsymbol{X}^{in}$. Throughout the text we assume that $\mathcal{A}^*_{\boldsymbol{X}^{in}} : \mathbb{T}_{out} \to \mathbb{T}_w$ is the adjoint operator. The adjoint operator needs to satisfy the following property:

$$\forall \boldsymbol{W} \in \mathbb{T}_w, \forall \boldsymbol{Z} \in \mathbb{T}_{out} : \quad \langle \mathcal{A}_{\boldsymbol{X}^{in}}(\boldsymbol{W}), \boldsymbol{Z}\rangle_{\mathbb{T}_{out}} = \langle \boldsymbol{W}, \mathcal{A}^*_{\boldsymbol{X}^{in}}(\boldsymbol{Z})\rangle_{\mathbb{T}_w}.$$

This is the fundamental property that we verify when forming the adjoint operator.

*(Corresponding Author) Robinson College of Business, Georgia State University, Atlanta, GA. Email: aaghasi@gsu.edu
†School of Electrical and Computer Engineering, Georgia Tech, Atlanta, GA. Emails: {abdi,jrom}@ece.gatech.edu.

Program (2) can be equivalently cast as

$$\underset{\substack{\boldsymbol{W}^{(1)}\in\mathbb{T}_{out}\\ \boldsymbol{W}^{(2)},\boldsymbol{W}^{(3)}\in\mathbb{T}_{w}}}{\text{minimize}} \quad f_1\left(\boldsymbol{W}^{(1)}\right)+f_2\left(\boldsymbol{W}^{(2)}\right) \quad \text{subject to} \quad \begin{cases} \boldsymbol{W}^{(1)}=\mathcal{A}_{\boldsymbol{X}^{in}}\left(\boldsymbol{W}^{(3)}\right) \\ \boldsymbol{W}^{(2)}=\boldsymbol{W}^{(3)} \end{cases}, \tag{3}$$

where

$$f_1\left(\boldsymbol{W}\right)=\mathcal{I}_{\left\|\boldsymbol{W}_{\Omega}-\boldsymbol{X}_{\Omega}^{out}\right\|_F\leq\epsilon}\left(\boldsymbol{W}\right)+\mathcal{I}_{\boldsymbol{W}_{\Omega^c}\leq\boldsymbol{V}_{\Omega^c}}\left(\boldsymbol{W}\right), \text{ and } \ f_2\left(\boldsymbol{W}\right)=\left\|\boldsymbol{W}\right\|_1,$$

and $\mathcal{I}_C(\cdot)$ represents the indicator function of the set C:

$$\mathcal{I}_C(\boldsymbol{W})=\begin{cases} 0 & \boldsymbol{W}\in C \\ +\infty & \boldsymbol{W}\notin C \end{cases}.$$

For the convex program (3), the ADMM update for each variable at the k-th iteration follows the standard forms

$$\boldsymbol{W}_{k+1}^{(1)}=\underset{\boldsymbol{W}}{\arg\min}\ f_1\left(\boldsymbol{W}\right)+\frac{\rho}{2}\left\|\boldsymbol{W}+\boldsymbol{U}_k^{(1)}-\mathcal{A}_{\boldsymbol{X}^{in}}\left(\boldsymbol{W}_k^{(3)}\right)\right\|_F^2, \tag{4}$$

$$\boldsymbol{W}_{k+1}^{(2)}=\underset{\boldsymbol{W}}{\arg\min}\ f_2\left(\boldsymbol{W}\right)+\frac{\rho}{2}\left\|\boldsymbol{W}+\boldsymbol{U}_k^{(2)}-\boldsymbol{W}_k^{(3)}\right\|_F^2, \tag{5}$$

$$\boldsymbol{W}_{k+1}^{(3)}=\underset{\boldsymbol{W}}{\arg\min}\ \frac{\rho}{2}\left\|\boldsymbol{W}_{k+1}^{(1)}+\boldsymbol{U}_k^{(1)}-\mathcal{A}_{\boldsymbol{X}^{in}}\left(\boldsymbol{W}\right)\right\|_F^2+\frac{\rho}{2}\left\|\boldsymbol{W}_{k+1}^{(2)}+\boldsymbol{U}_k^{(2)}-\boldsymbol{W}\right\|_F^2, \tag{6}$$

and the dual updates are performed via

$$\boldsymbol{U}_{k+1}^{(1)}=\boldsymbol{U}_k^{(1)}+\boldsymbol{W}_{k+1}^{(1)}-\mathcal{A}_{\boldsymbol{X}^{in}}\left(\boldsymbol{W}_k^{(3)}\right), \quad \boldsymbol{U}_{k+1}^{(2)}=\boldsymbol{U}_k^{(2)}+\boldsymbol{W}_{k+1}^{(2)}-\boldsymbol{W}_{k+1}^{(3)}.$$

Obtaining closed-form expressions for the programs (4) and (5) is identical to the Net-Trim derivation in ([AAR18]) and we bring them here for completeness. For $\boldsymbol{W}^{(1)}$ update we have

$$\left(\boldsymbol{W}_{k+1}^{(1)}\right)_{\Omega}=\underset{\boldsymbol{W}_{\Omega}:\ \left\|\boldsymbol{W}_{\Omega}-\boldsymbol{X}_{\Omega}^{out}\right\|_F\leq\epsilon}{\arg\min}\ \frac{\rho}{2}\left\|\boldsymbol{W}_{\Omega}-\left(\mathcal{A}_{\boldsymbol{X}^{in}}\left(\boldsymbol{W}_k^{(3)}\right)-\boldsymbol{U}_k^{(1)}\right)_{\Omega}\right\|_F^2,$$

and the closed form expression is obtained using the fact that

$$\underset{\boldsymbol{W}_{\Omega}:\ \left\|\boldsymbol{W}_{\Omega}-\boldsymbol{Z}_{\Omega}\right\|_F\leq\epsilon}{\arg\min}\ \frac{\rho}{2}\left\|\boldsymbol{W}_{\Omega}-\boldsymbol{Y}_{\Omega}\right\|_F^2=\begin{cases} \boldsymbol{Y}_{\Omega} & \text{if } \left\|\boldsymbol{Y}_{\Omega}-\boldsymbol{Z}_{\Omega}\right\|_F\leq\epsilon \\ \boldsymbol{Z}_{\Omega}+\epsilon\frac{\boldsymbol{Y}_{\Omega}-\boldsymbol{Z}_{\Omega}}{\left\|\boldsymbol{Y}_{\Omega}-\boldsymbol{Z}_{\Omega}\right\|_F} & \text{else} \end{cases}.$$

Also

$$\left(\boldsymbol{W}_{k+1}^{(1)}\right)_{\Omega^c}=\underset{\boldsymbol{W}_{\Omega^c}:\ \boldsymbol{W}_{\Omega^c}\leq\boldsymbol{V}_{\Omega^c}}{\arg\min}\ \frac{\rho}{2}\left\|\boldsymbol{W}_{\Omega^c}-\left(\mathcal{A}_{\boldsymbol{X}^{in}}\left(\boldsymbol{W}_k^{(3)}\right)-\boldsymbol{U}_k^{(1)}\right)_{\Omega^c}\right\|_F^2,$$

to obtain which we use the fact that

$$\underset{\boldsymbol{W}_{\Omega^c}:\, \boldsymbol{W}_{\Omega^c} \leq \boldsymbol{V}_{\Omega^c}}{\arg\min} \ \frac{\rho}{2} \left\| \boldsymbol{W}_{\Omega^c} - \boldsymbol{Y}_{\Omega^c} \right\|_F^2 = \boldsymbol{Y}_{\Omega^c} - \left(\boldsymbol{Y}_{\Omega^c} - \boldsymbol{V}_{\Omega^c} \right)^+ .$$

Finally, the solution to (5) is the standard soft thresholding operator (e.g., see §4.4.3 of [BPC$^+$11]), which reduces the update to

$$\boldsymbol{W}_{k+1}^{(2)} = S_{1/\rho}\left(\boldsymbol{W}_k^{(3)} - \boldsymbol{U}_k^{(2)} \right), \ \text{where} \ \ S_c(w) = \begin{cases} w - c & w > c \\ 0 & |w| \leq c \\ w + c & w < -c \end{cases} .$$

Note that when $S_{1/\rho}$ applies to a tensor, it applies to each element individually. After combining the steps above, we propose Algorithm 1 as a computational scheme to address the Net-Trim in operator form.

Algorithm 1: Implementation of the Net-Trim In Operator Form

Input: $\boldsymbol{X}^{in} \in \mathbb{T}_{in}$, $\boldsymbol{X}^{out} \in \mathbb{T}_{out}$, Ω, $\boldsymbol{V}_\Omega$, ϵ, ρ
initialize $\boldsymbol{U}^{(1)} \in \mathbb{T}_{out}, \boldsymbol{U}^{(2)} \in \mathbb{T}_w$ and $\boldsymbol{W}^{(3)} \in \mathbb{T}_w$ `% all initializations can be with 0`
while *not converged* **do**
- $\boldsymbol{Y} \leftarrow \mathcal{A}_{\boldsymbol{X}^{in}}\left(\boldsymbol{W}^{(3)} \right) - \boldsymbol{U}^{(1)}$
- **if** $\left\| \boldsymbol{Y}_\Omega - \boldsymbol{X}_\Omega^{out} \right\|_F \leq \epsilon$ **then**
  - $\boldsymbol{W}_\Omega^{(1)} \leftarrow \boldsymbol{Y}_\Omega$
- **else**
  - $\boldsymbol{W}_\Omega^{(1)} \leftarrow \boldsymbol{X}_\Omega^{out} + \epsilon \left\| \boldsymbol{Y}_\Omega - \boldsymbol{X}_\Omega^{out} \right\|_F^{-1} \left(\boldsymbol{Y}_\Omega - \boldsymbol{X}_\Omega^{out} \right)$
- **end**
- $\boldsymbol{W}_{\Omega^c}^{(1)} \leftarrow \boldsymbol{Y}_{\Omega^c} - \left(\boldsymbol{Y}_{\Omega^c} - \boldsymbol{V}_{\Omega^c} \right)^+$
- $\boldsymbol{W}^{(2)} \leftarrow S_{1/\rho}\left(\boldsymbol{W}^{(3)} - \boldsymbol{U}^{(2)} \right)$ `% applies to each element of the tensor`
- $\boldsymbol{W}^{(3)} \leftarrow \arg\min_{\boldsymbol{W}} \ \frac{1}{2} \left\| \mathcal{A}_{\boldsymbol{X}^{in}}(\boldsymbol{W}) - \left(\boldsymbol{W}^{(1)} + \boldsymbol{U}^{(1)} \right) \right\|_F^2 + \frac{1}{2} \left\| \boldsymbol{W} - \left(\boldsymbol{W}^{(2)} + \boldsymbol{U}^{(2)} \right) \right\|_F^2$
- $\boldsymbol{U}^{(1)} \leftarrow \boldsymbol{U}^{(1)} + \boldsymbol{W}^{(1)} - \mathcal{A}_{\boldsymbol{X}^{in}}\left(\boldsymbol{W}^{(3)} \right)$
- $\boldsymbol{U}^{(2)} \leftarrow \boldsymbol{U}^{(2)} + \boldsymbol{W}^{(2)} - \boldsymbol{W}^{(3)}$

end
Output: $\boldsymbol{W}^{(3)}$

The only undiscussed part in Algorithm 1 is the update for $\boldsymbol{W}^{(3)}$ which we address using an operator form of the conjugate gradient algorithm explained in the next section.

2 Least Squares Update Using an Operator Conjugate Gradient

In this section we address the minimization

$$\underset{\boldsymbol{W} \in \mathbb{T}_w}{\text{minimize}} \ \frac{1}{2} \left\| \mathcal{A}_{\boldsymbol{X}^{in}}(\boldsymbol{W}) - \boldsymbol{B} \right\|_F^2 + \frac{1}{2} \left\| \boldsymbol{W} - \boldsymbol{C} \right\|_F^2 , \tag{7}$$

where $\mathcal{A}_{\boldsymbol{X}^{in}} : \mathbb{T}_w \to \mathbb{T}_{out}$, and $\boldsymbol{B} \in \mathbb{T}_{out}$ and $\boldsymbol{C} \in \mathbb{T}_w$ are tensors. The minimizer to (7) can be found by taking a derivative and setting it to zero, i.e.,

$$\mathcal{A}^*_{\boldsymbol{X}^{in}} \left(\mathcal{A}_{\boldsymbol{X}^{in}} \left(\boldsymbol{W} \right) - \boldsymbol{B} \right) + \boldsymbol{W} - \boldsymbol{C} = \boldsymbol{0}, \tag{8}$$

or

$$\mathcal{A}^*_{\boldsymbol{X}^{in}} \left(\mathcal{A}_{\boldsymbol{X}^{in}} \left(\boldsymbol{W} \right) \right) + \boldsymbol{W} = \mathcal{A}^*_{\boldsymbol{X}^{in}} \left(\boldsymbol{B} \right) + \boldsymbol{C}. \tag{9}$$

Solving (9) for $\boldsymbol{W}$ is efficiently possible via the method of conjugate gradient. The following algorithm outlines the process of solving (9), which is a variant of the original CG algorithm (e.g., see [CGL]) modified to address (9).

Algorithm 2: Least Squares Update in the Net-Trim Using Conjugate Gradient

initialize $\boldsymbol{W}_0 = \boldsymbol{0}$; $\boldsymbol{R}_0 = \mathcal{A}^*_{\boldsymbol{X}^{in}} \left(\boldsymbol{B} \right) + \boldsymbol{C}$; $\boldsymbol{P}_0 = \boldsymbol{R}_0$

for $k = 1, \ldots, K_{\max}$ **do**

- $\boldsymbol{T}_{k-1} = \mathcal{A}_{\boldsymbol{X}^{in}} \left(\boldsymbol{P}_{k-1} \right)$
- $\alpha_k = \frac{\|\boldsymbol{R}_{k-1}\|_F^2}{\|\boldsymbol{T}_{k-1}\|_F^2 + \|\boldsymbol{P}_{k-1}\|_F^2}$
- $\boldsymbol{W}_k = \boldsymbol{W}_{k-1} + \alpha_k \boldsymbol{P}_{k-1}$
- $\boldsymbol{R}_k = \boldsymbol{R}_{k-1} - \alpha_k \left(\mathcal{A}^*_{\boldsymbol{X}^{in}} \left(\boldsymbol{T}_{k-1} \right) + \boldsymbol{P}_{k-1} \right)$
- $\beta_k = \frac{\|\boldsymbol{R}_k\|_F^2}{\|\boldsymbol{R}_{k-1}\|_F^2}$
- $\boldsymbol{P}_k = \boldsymbol{R}_k + \beta_k \boldsymbol{P}_{k-1}$

end

Output: $\boldsymbol{W}_{K_{\max}}$

References

[AAR18] Alireza Aghasi, Afshin Abdi, and Justin Romberg. Fast convex pruning of deep neural networks. *arXiv preprint arXiv:1806.06457*, 2018.

[BPC+11] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[CGL] Conjugate Gradient Algorithm, howpublished = `https://math.aalto.fi/opetus/inv/cgalgorithm.pdf`, note = Accessed: 2018-07-18.