Novosibirsk State University
*THE REAL SCIENCE

Gradient Centralization: A New Optimization

Presented by: Alix Bernard

Authors of the paper:
Hongwei Yong
Jianqiang Huang
Xiansheng Hua
Lei Zhang

Publication date: April 8, 2020

March 16, 2021

Overview

Introduction

Neural Network optimizers have two goals:

- speed up training process
- improve model generalization capabilities

Divers optimization algorithms exist such as **Stochastic Gradient Descent with Momentum** (SGDM) and **Adam**.
Normalization of the network activations: **Batch Normalization** (BN), Group Normalization (GN).
Normalization of the weights: **Weight Standardization** (WS), Weight Normalization (WN).

Introduction

The paper referenced [1] introduce the new technique **Gradient Centralization** (GC), it operates on the gradient of weight vectors by centralizing the gradient vector to have zero mean. It can be implemented in gradient based optimization algorithms with one line of code.
This technique demonstrates a shortened training process, improved generalization performance, and compatibility for fine-tuning pre-trained models.

Novosibirsk State University
*THE REAL SCIENCE

Introduction

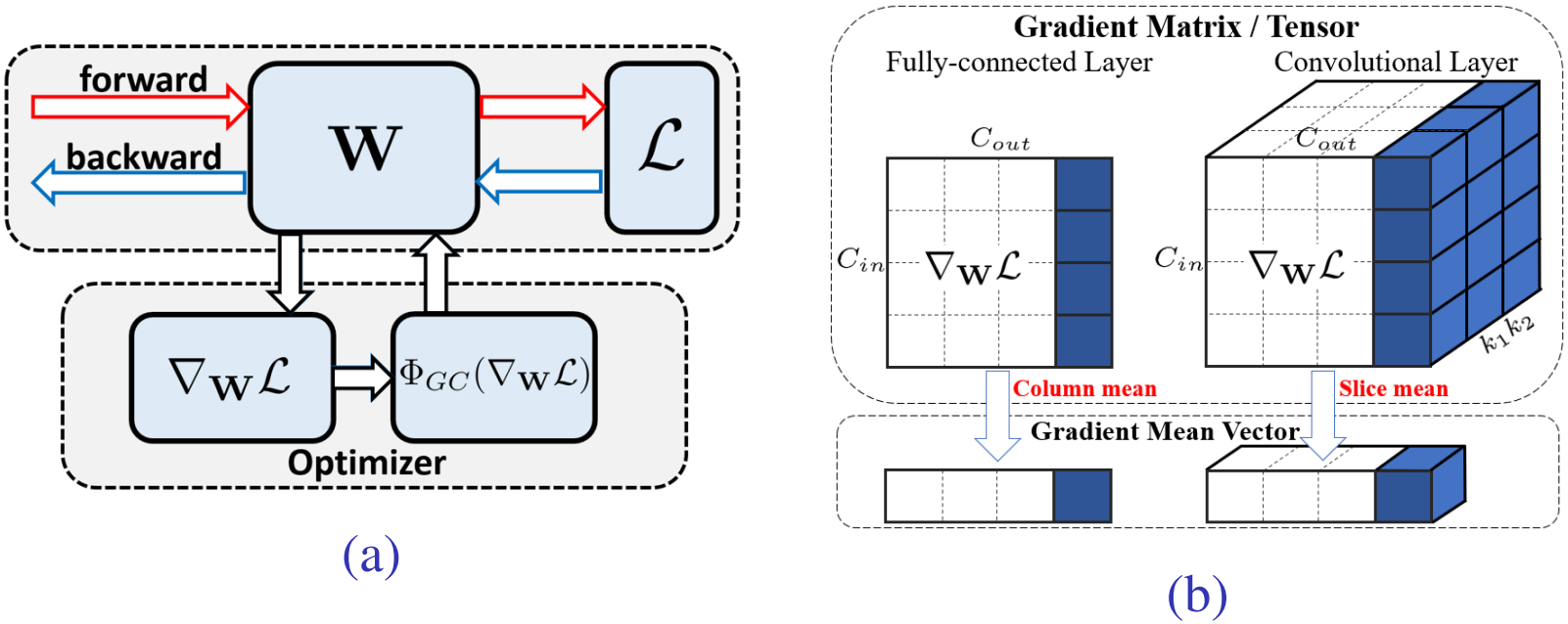

Fig. 1. (a) Sketch map for using gradient centralization (GC). W is the weight matrix, $\mathcal{L}$ is the loss function, $\nabla\mathcal{L}$ is the gradient of weight, and $\Phi_{GC}(\nabla W\mathcal{L})$ is the centralized gradient. It is very simple to embed GC into existing network optimizers by replacing $\nabla W\mathcal{L}$ with $\Phi_{GC}(\nabla W\mathcal{L})$. (b) Illustration of the GC operation on gradient matrix/tensor of weights in the fully-connected layer (left) and convolutional layer (right). GC computes the column/slice mean of gradient matrix/tensor and centralizes each column/slice to have zero mean.

Gradient Centralization

Using Z-score standardization to normalize the gradient, as in BN and WS, was found not to improve the stability of training, as opposed to computing the mean of gradient vectors and centralizing them to have zero mean.
The GC operator denoted as Φ_{GC} is define as:

$$\Phi_{GC}(\nabla w_i \mathcal{L}) = \nabla w_i \mathcal{L} - \mu_{\nabla w_i \mathcal{L}} \tag{1}$$

where $\mathcal{L}$ is the loss function, w_i is a weight vector whose gradient is $\nabla w_i \mathcal{L}$, and $\mu_{\nabla w_i \mathcal{L}} = \frac{1}{M} \sum_{j=1}^{M} \nabla w_i \mathcal{L}$.

Gradient Centralization

Algorithm 1 SGDM with Gradient Centralization

Input: Weight vector $\mathbf{w}^0$, step size α, momentum factor β, $\mathbf{m}^0$

Training step:

1: **for** $t = 1, ...T$ **do**
2: $\mathbf{g}^t = \nabla_{\mathbf{w}^t}\mathcal{L}$
3: $\widehat{\mathbf{g}}^t = \Phi_{GC}(\mathbf{g}^t)$
4: $\mathbf{m}^t = \beta\mathbf{m}^{t-1} + (1-\beta)\widehat{\mathbf{g}}^t$
5: $\mathbf{w}^{t+1} = \mathbf{w}^t - \alpha\mathbf{m}_t$
6: **end for**

Algorithm 2 Adam with Gradient Centralization

Input: Weight vector $\mathbf{w}^0$, step size α, β_1, β_2, ϵ, $\mathbf{m}^0$,$\mathbf{v}^0$

Training step:

1: **for** $t = 1, ...T$ **do**
2: $\mathbf{g}^t = \nabla_{\mathbf{w}^t}\mathcal{L}$
3: $\widehat{\mathbf{g}}^t = \Phi_{GC}(\mathbf{g}^t)$
4: $\mathbf{m}^t = \beta_1\mathbf{m}^{t-1} + (1-\beta_1)\widehat{\mathbf{g}}^t$
5: $\mathbf{v}^t = \beta_2\mathbf{v}^{t-1} + (1-\beta_2)\widehat{\mathbf{g}}^t \odot \widehat{\mathbf{g}}^t$
6: $\widehat{\mathbf{m}}^t = \mathbf{m}^t/(1-(\beta_1)^t)$
7: $\widehat{\mathbf{v}}^t = \mathbf{v}^t/(1-(\beta_2)^t)$
8: $\mathbf{w}^{t+1} = \mathbf{w}^t - \alpha\frac{\widehat{\mathbf{m}}^t}{\sqrt{\widehat{\mathbf{v}}^t}+\epsilon}$
9: **end for**

Embedding of GC in the SGDM and Adam algorithms

Gradient Centralization

GC can also be viewed as a projected gradient descent. Let's rewrite equation (1) as a matrix formulation:

$$\Phi_{GC}(\nabla_w \mathcal{L}) = \mathbf{P}\nabla_w \mathcal{L}, \qquad \mathbf{P} = I - ee^T \tag{2}$$

where $e = \frac{1}{\sqrt{M}}1$ denotes an M–dimensional unit vector and I the identity matrix of size $M \times M$. Then $\mathbf{P}$ is the projection matrix on the hyperplane, in weight space, determined by $e^T(w - w^t) = 0$. [1]

[1] WS uses the constraints $e^T w = 0$ but the initial weights may not satisfy this constraint therefore limiting its practical applications.

Gradient Centralization

The objective function can be written as:

$$\min_{w} \mathcal{L}(w), \qquad s.t. \qquad e^T(w - w^0) = 0 \tag{3}$$

This regularization constraint reduces the possibility of over-fitting.

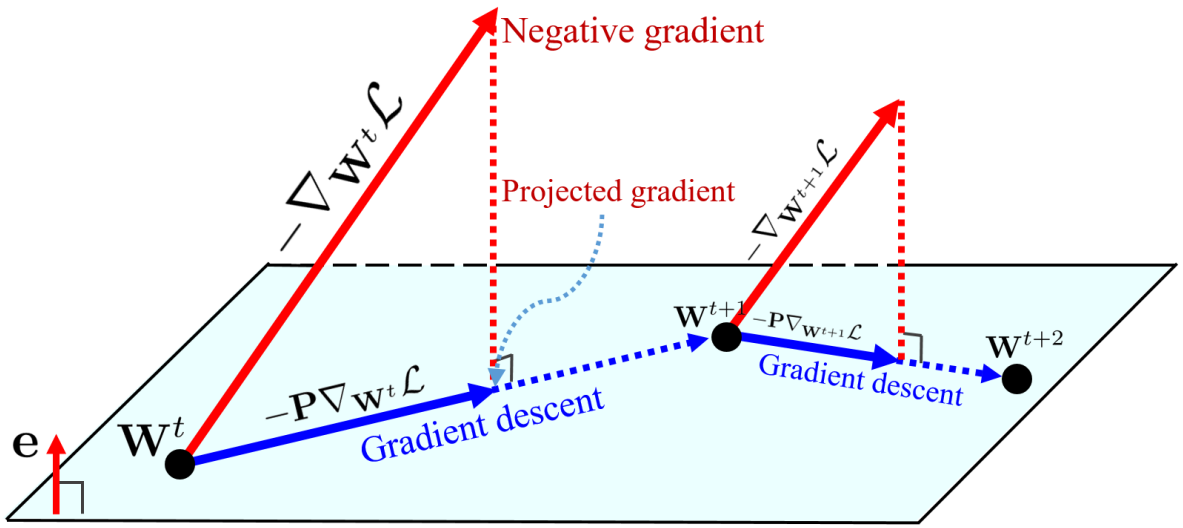

Fig. 2. The geometrical interpretation of GC. The gradient is projected on a hyperplane $e^T(w - w^t) = 0$, where the projected gradient is used to update the weight.

Novosibirsk State University
*THE REAL SCIENCE

Results

Experimental results have been obtained on the following datasets:

- Mini-ImageNet
- CIFAR100
- ImageNet
- Fine-grained image classification datasets (FGVC Aircraft, Stanford Cars, Stanford Dogs, CUB-200-2011)
- Object detection and segmentation dataset (COCO)

Novosibirsk State University
*THE REAL SCIENCE

Mini-ImageNet

In **Fig. 3.** we can see that the red and blue lines, that correspond to the use of GC, have a lower training loss than without GC, and a better test accuracy as well (or at least as good).

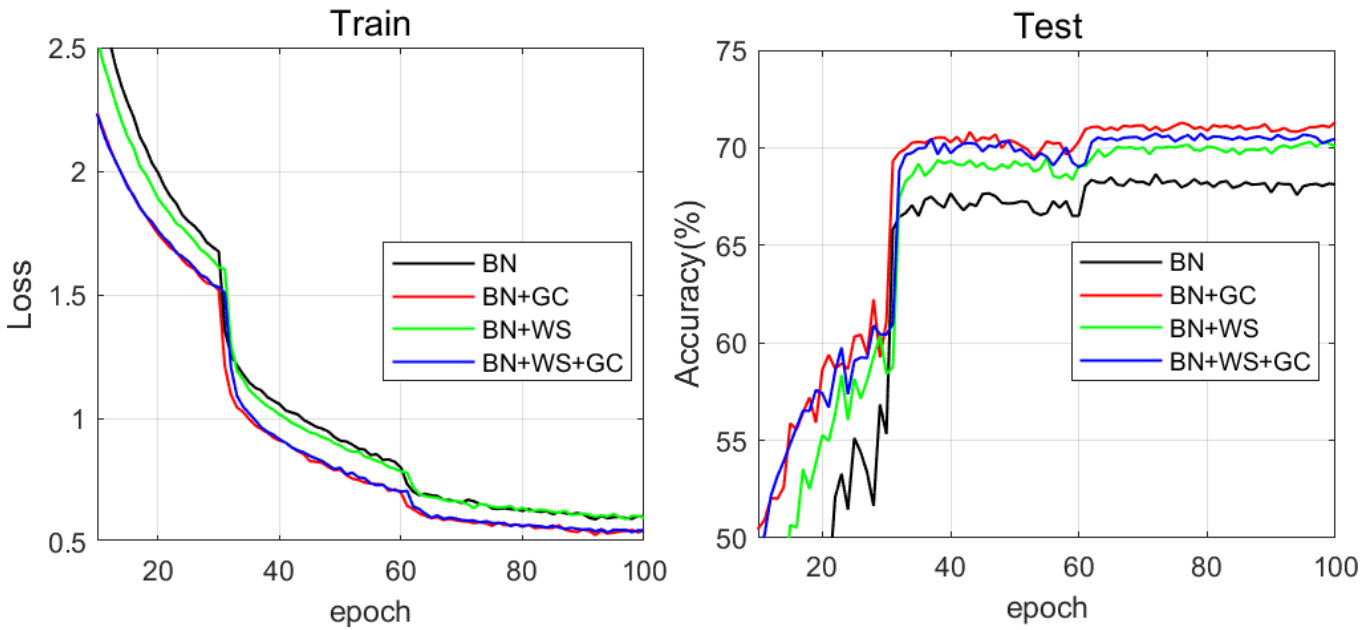

Fig. 3. Training loss and test accuracy curves vs. training epoch on Mini-ImageNet. The ResNet50 is used as the DNN model.

CIFAR100

In **Table 1.** are displayed the test accuracies [2] on the dataset CIFAR100 with different DNN architectures and optimizers.

Architecture	Optimizer	w/o GC	w/ GC	Increase
ResNet18	SGDM	76.87	78.82	1.95
ResNet50	SGDM	78.23	79.14	0.89
	Adam	71.64	72.80	1.16
	Adagrad	70.34	71.58	1.24
VGG11	SGDM	70.94	71.69	0.75
DenseNet121	SGDM	79.31	79.68	0.37

Table 1. Testing accuracies (%) on CIFAR100

[2]More test accuracies with different weight decays and learning rates are present in [1].

Novosibirsk State University
*THE REAL SCIENCE

ImageNet

Fig. 4. shows that GC speed up the training with Group Normalization. More results in [1] with ResNet50, ResNet101, BN, and GN show an improvement of performance of 0.5% ~ 1.2%.

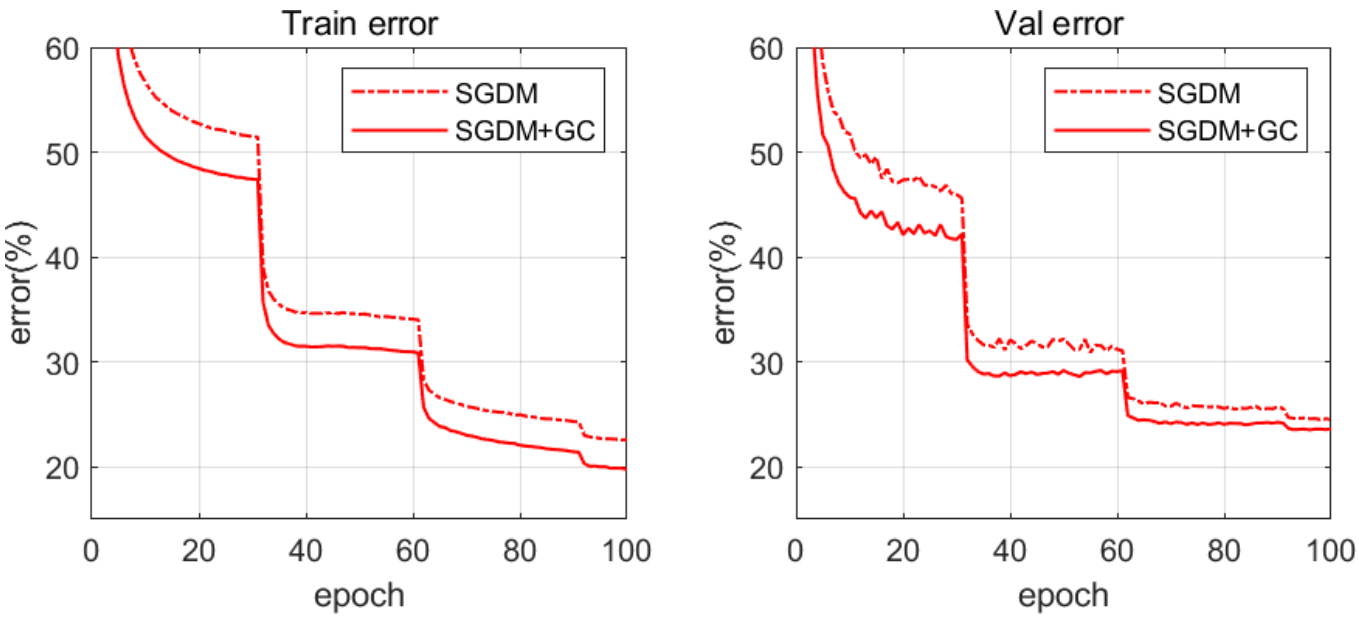

Fig. 4. Training error and validation error curves vs. training epoch on ImageNet. The ResNet50 architecture is used as the DNN model with GN.

Other datasets

Results on the datasets for fine-grained image classification show improvements of up to 2.1% in classification accuracy with GC.

Results on the datasets for object detection and segmentation also show performance gain in the average precision of the order of $0.3\% \sim 0.9\%$ with GC.

Conclusion

GC has demonstrated improvement in accuracy for DNNs, it showed that it speeds up the training process and has a smoother optimization landscape. It can be used in addition to other techniques such as BN and WS, and works with different optimizers as well as different DNN architectures. Finally, it can also be used to fine-tune pre-trained models. All of this with an easy implementation.

Paper: arxiv.org/abs/2004.01461

Code: github.com/Yonghongwei/Gradient-Centralization