Les Cahiers du GERAD ISSN: 0711–2440

Batch normalization in quantized networks

E. Sari,
V. Partovi Nia

G–2020–23–EIW01

April 2020

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2020
– Bibliothèque et Archives Canada, 2020

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2020
– Library and Archives Canada, 2020

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Batch normalization in quantized networks

Eyyüb Sari [a]

Vahid Partovi Nia [a,b]

[a] *Huawei Noah's Ark Lab, Montréal (Québec), Canada, H3N 1X9*

[b] *GERAD, HEC Montréal, Montréal (Québec), Canada, H3T 2A7*

eyyub.sari@huawei.com
vahid.partovinia@huawei.com

April 2020
Les Cahiers du GERAD
G–2020–23–EIW01

Abstract: *Implementation of quantized neural networks on computing hardware leads to considerable speed up and memory saving. However, quantized deep networks are difficult to train and batch normalization (BatchNorm) layer plays an important role in training full-precision and quantized networks. Most studies on BatchNorm are focused on full-precision networks, and there is little research in understanding BatchNorm affect in quantized training which we address here. We show BatchNorm avoids gradient explosion which is counter-intuitive and recently observed in numerical experiments by other researchers.*

1 Introduction

Deep Neural Networks (DNNs) compression through quantization is a recent direction in edge implementation of deep networks. Quantized networks are simple to deploy on hardware devices with constrained resources such as cell phones and IoT equipment. Quantized networks not only consume less memory and simplify computation, it also yields energy saving. Two well-known extreme quantization schemes are binary (one bit) and and ternary (two bit) networks, which allow up to $32\times$ and $16\times$ computation speed up, respectively. Binary quantization only keep track of the sign $\{-1, +1\}$ and ignores the magnitude, and ternary quantization extends the binary case to $\{-1, 0, +1\}$ to allow for sparse representation. BatchNorm facilitates neural networks training as a known fact. A common intuition suggests BatchNorm matches input and output first and second moments. There are two other clues among others: [4] claim that BatchNorm corrects covariate shift, and [6] show BatchNorm bounds the gradient and makes the optimization smoother in full-precision networks. None of these arguments work for quantized networks! The role of BatchNorm is to prevent exploding gradient empirically observed in [1] and [3].

2 Full-precision Network

Suppose a mini batch of size B for a given neuron k. Let $\hat{\mu}_k, \hat{\sigma}_k$ be the mean and the standard deviation of the dot product, between inputs and weights, $s_{bk}, b = 1, \dots B$. For a given layer l, BatchNorm is defined as $\mathrm{BN}(s_{bk}) \equiv z_{bk} = \gamma_k \hat{s}_{bk} + \beta_k$, where $\hat{s}_{bk} = \frac{s_{bk} - \hat{\mu}_k}{\hat{\sigma}_k}$ is the standardized dot product and the pair (γ_k, β_k) is trainable, initialized with $(1, 0)$.

Given the objective function $\mathcal{L}(.)$, BatchNorm parameters are trained in backpropagation

$$\frac{\partial \mathcal{L}}{\partial \beta_k} = \sum_{b=1}^{B} \frac{\partial \mathcal{L}}{\partial z_{bk}}, \quad \frac{\partial \mathcal{L}}{\partial \gamma_k} = \sum_{b=1}^{B} \frac{\partial \mathcal{L}}{\partial z_{bk}} \hat{s}_{bk},$$

For a given layer l, it is easy to prove $\frac{\partial \mathcal{L}}{\partial s_{bk}}$ equals

$$\frac{\gamma_k}{\hat{\sigma}_k} \Big(- \frac{1}{B} \sum_{b'=1}^{B} \frac{\partial \mathcal{L}}{\partial z_{b'k}} - \frac{\hat{s}_{bk}}{B} \sum_{b'=1}^{B} \frac{\partial \mathcal{L}}{\partial z_{b'k}} \hat{s}_{b'k} + \frac{\partial \mathcal{L}}{\partial z_{bk}} \Big). \tag{1}$$

Assume weights and activations are independent, and identically distributed (iid) and centred about zero. Formally, denote the dot product vector $\mathbf{s}_b^l \in \mathbb{R}^{K_l}$ of sample b in layer l, with K_l neurons. Let f be the element-wise activation function, $\mathbf{x}_b$ be the input vector, $\mathbf{W}^l \in \mathbb{R}^{K_{l-1} \times K_l}$ with elements $\mathbf{W}^l = [w_{kk'}^l]$ be the weights matrix; one may use w^l to denote an identically distributed elements of layer l. It is easy to verify

$$\frac{\partial \mathcal{L}}{\partial s_{bk}^l} \quad = \quad f'(s_{bk}^l) \sum_{k'=1}^{K_{l+1}} w_{kk'}^{l+1} \frac{\partial \mathcal{L}}{\partial s_{bk'}^{l+1}},$$

$$\frac{\partial \mathcal{L}}{\partial w^l_{k'k}} = \sum_{b=1}^{B} s^{l-1}_{bk'} \frac{\partial \mathcal{L}}{\partial s^l_{bk}}.$$

Assume that the feature element x and the weight element w are centred and iid. Reserve k to index the current neuron and use k' for the previous or the next layer neuron and where $\mathbb{V}(w^{l'})$ is the variance of the weight in layer l' $\mathbb{V}(s^l_{bk}) = \mathbb{V}(x)\prod_{l'=1}^{l-1} K_{l'}\,\mathbb{V}(w^{l'})$,

$$\mathbb{V}(\frac{\partial \mathcal{L}}{\partial s^l_{bk}}) = \mathbb{V}(\frac{\partial \mathcal{L}}{\partial s^L}) \prod_{l'=l+1}^{L} K_{l'}\,\mathbb{V}(w^{l'}),$$

which explodes or vanishes depending on $\mathbb{V}(w^{l'})$. This is the main reason common full-precision initialization methods suggest $\mathbb{V}(w^l) = \frac{1}{K_l}$. For any full-precision network, BatchNorm affects back-propagation as

$$\begin{aligned}\mathbb{V}\left(\frac{\partial \mathcal{L}}{\partial s^l_{bk}}\right) &= \left(\frac{\gamma^l_k}{B\hat{\sigma}^l_k}\right)^2 \{B^2 + 2B - 1 + \mathbb{V}(\hat{s}^{l^2}_{bk})\} \\ &\quad K_{l+1}\,\mathbb{V}(w^{l+1})\,\mathbb{V}\left(\frac{\partial \mathcal{L}}{\partial s^{l+1}}\right). \end{aligned} \tag{2}$$

3 Binary network

Controlling the variance has no fundamental effect on forward propagation if s_{bk} is symmetric about zero as the sign function filters the magnitude and only keeps the sign of the dot product. The term $b_k = \mu_k - \frac{\hat{\sigma}_k}{\gamma_k}\beta_k$ can be regarded as as a new trainable parameter, thus BatchNorm layer can be replaced by adding biases to the network to compensate. [7] shows that the gradient variance for binary quantized networks without BatchNorm is

$$\mathbb{V}\left(\frac{\partial \mathcal{L}}{\partial s^l_{bk}}\right) = \mathbb{V}\left(\frac{\partial \mathcal{L}}{\partial s^L}\right) \prod_{l'=l+1}^{L} K_{l'},$$

and with BatchNorm is

$$\mathbb{V}\left(\frac{\partial \mathcal{L}}{\partial s^l_{bk}}\right) = \prod_{l'=l}^{L-1} \frac{K_{l'+1}}{K_{l'-1}}\,\mathbb{V}\left(\frac{\partial \mathcal{L}}{\partial s^L}\right) + o\left(\frac{1}{B^{1-\epsilon}}\right),$$

for an arbitrary $0 < \epsilon < 1$.

Gradients are stabilized only if $\left(\frac{\gamma^l_k}{B}\right)^2 \{B^2 + 2B - 1 + \mathbb{V}(\hat{s}^{l^2}_{bk})\} \approx 1$. Moving from full-precision weight w to binary weight $\tilde{w} = \text{sign}(w)$ changes the situation dramatically: i) BatchNorm corrects exploding gradients in BNNs as the layer width ratio $\frac{K_{l+1}}{K_{l-1}} \approx 1$ in common neural models. If this ratio diverges from unity binary training is problematic even with BatchNorm.

4 Ternary network

Ternary neural networks (TNNs) are studied in [8] and the BatchNorm effect is detailed there. Full-precision weights during training are ternarized during forward propagation. Given a threshold Δ ternary quantization function is

$$\text{tern}(x) = \begin{cases} -1 & \text{if } x < -\Delta \\ +1 & \text{if } x > \Delta \\ 0 & \text{if } -\Delta \leq x \leq \Delta \end{cases} \tag{3}$$

Let's suppose the threshold is given so that the learning is feasible, for instance Δ is tuned so that $< 50\%$ of ternary weights are set to zero

$$\mathbb{V}(\tilde{w}_t^l) = 2p_1 = 1 - \frac{\Delta}{\sqrt{\frac{6}{K_l}}}. \tag{4}$$

In the literature [5] suggests to set $\Delta_l = 0.7\mathbb{E}(|w^l|)$. Under simplified assumptions of iid weight and activation

$$\Delta_l = \frac{0.7}{2}\sqrt{\frac{6}{K_l}} \tag{5}$$

and (4) reduces to $\mathbb{V}(\tilde{w}_t^l) = 1 - \frac{0.7}{2} = 0.65$. In this setting, variance is bigger than $\frac{2}{K_l}$ which produces exploding gradients similar to the binary case. Suppose weights and activation are iid and weights are centred about zero, for a layer l,

$$\hat{\sigma}_k^2 = K_{l-1}\frac{1}{2}\,\mathbb{V}(\hat{s}_b^{l-1})\,\mathbb{V}(\tilde{w}_t^l) = K_{l-1}\frac{1}{2}\,\mathbb{V}(\tilde{w}_t^l). \tag{6}$$

Therefore (2) reduces to

$$\mathbb{V}\left(\frac{\partial \mathcal{L}}{\partial s_{bk}^l}\right) = \left\{1 + o\left(\frac{1}{B^{1-\epsilon}}\right)\right\} \tag{7}$$

$$\frac{K_{l+1}}{K_{l-1}}\,\mathbb{V}\left(\frac{\partial \mathcal{L}}{\partial s^{l+1}}\right), \tag{8}$$

see [8] for details. Similar to the binary case, in most deep architectures $K_{l+1} \approx K_{l-1}$ or equivalently $\frac{K_{l+1}}{K_{l-1}} \approx 1$, so the variance would not explode for networks with BatchNorm layer.

5 Conclusion

We derived the analytical expression for full-precision network under assumptions of [2] and extended it for binary and ternary case. Our study shows that the real effect of BatchNorm is played in scaling. The main role of BatchNorm in quantized training is to adjust gradient explosion.

References

[1] Arash Ardakani, Zhengyun Ji, Sean C. Smithson, Brett H. Meyer, and Warren J. Gross. Learning recurrent binary/ternary weights. In International Conference on Learning Representations, 2019.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. CoRR, abs/1502.01852, 2015.

[3] Lu Hou, Jinhua Zhu, James Kwok, Fei Gao, Tao Qin, and Tie-yan Liu. Normalization helps training of quantized lstm. In Advances in Neural Information Processing Systems, pages 7344–7354, 2019.

[4] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. CoRR, abs/1502.03167, 2015.

[5] Fengfu Li and Bin Liu. Ternary weight networks. arXiv, abs/1605.04711, 2016.

[6] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, Advances in Neural Information Processing Systems 31, pages 2483–2493. Curran Associates, Inc., 2018.

[7] Eyyüb Sari, Mouloud Belbahri, and Vahid Partovi Nia. How does batch normalization help binary training? arXiv preprint arXiv:1909.09139v2, 2019.

[8] Eyyüb Sari and Vahid Partovi Nia. Understanding batchnorm in ternary training. Journal of Computational Vision and Imaging Systems, 5(1):2, Jan. 2020.