Recurrent Bilinear Optimization for Binary Neural Networks

Sheng Xu[1,†], Yanjing Li[1,†], Tiancheng Wang[1], Teli Ma[2], Baochang Zhang[1,3], Peng Gao[2], Yu Qiao[2], Jinhu Lü[1,3], Guodong Guo[4,5]
[1]Beihang University, [2]Shanghai AI Laboratory, [3]Zhongguancun Laboratory, [4]Baidu, [5]NERCDL

北京航空航天大學 BEIHANG UNIVERSITY

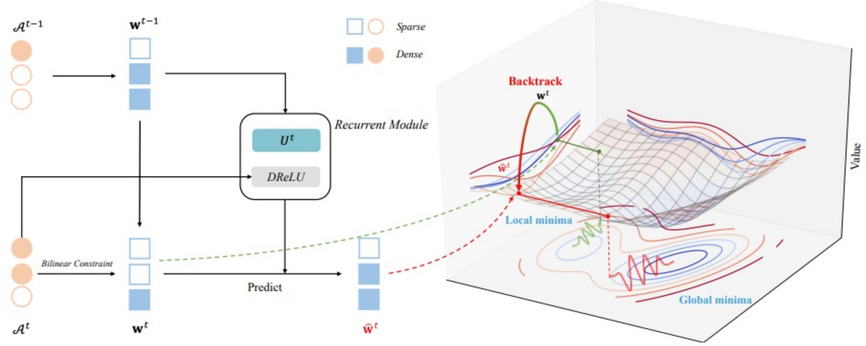
ECCV TEL AVIV 2022

Motivation

$$\arg\min_{\mathbf{w},\mathcal{A}} G(\mathbf{w},\mathcal{A}) = \| \mathbf{b}^{\mathbf{w}} - \mathcal{A}\mathbf{w}\|_2^2 + R(\mathbf{w})$$

- Such bilinear constraint $G(\mathbf{w},\mathcal{A})$ will lead to an asynchronous convergence problem.
- The variable with a slower convergence speed (usually $\mathbf{w}$) is not as sufficiently trained as another faster one.
- $\mathbf{w}$ will tendentiously fall into the local optimum with low magnitude when the magnitude of $\mathcal{A}$ is much larger than 0 (due to $\mathbf{b}^{\mathbf{w}} \in \{-1,+1\}$). On the contrary, $\mathbf{w}$ will have a large magnitude and thus slowly converge when elements of $\mathcal{A}$ are close to 0.

Recurrent Bilinear Optimization



Proposed Method

1. Bilinear Model of BNN

$$\arg\min_{\mathbf{w},\mathcal{A}} L_S(\mathbf{w},\mathcal{A}) + \lambda G(\mathbf{w},\mathcal{A})$$

2. BP considering the Bilinear Coupling Relationship

$$\mathrm{w}_{i,j}^{t+1} = \mathrm{w}_{i,j}^{t} - \eta_2 \frac{\partial L_S}{\partial \mathrm{w}_{i,j}^{t}} - \eta_2\lambda\left(\frac{\partial G}{\partial \mathrm{w}_{i,j}^{t}} + Tr\left(\left(\frac{\partial G}{\partial \mathcal{A}^{t}}\right)^T \frac{\partial \mathcal{A}^{t}}{\partial \mathrm{w}_{i,j}^{t}}\right)\right)$$

$$= \mathrm{w}_{i,j}^{t+1} - \eta_2\lambda Tr\left(\mathbf{w}^t \hat{G}(\mathbf{w}^t,\mathcal{A}^t)\frac{\partial \mathcal{A}^t}{\partial \mathrm{w}_{i,j}^{t}}\right)$$

3. To estimate $Tr\left(\mathbf{w}^t \hat{G}(\mathbf{w}^t,\mathcal{A}^t)\frac{\partial \mathcal{A}^t}{\partial \mathrm{w}_{i,j}^{t}}\right)$

$$\left(\frac{\partial \mathcal{A}}{\partial \mathbf{w}}\right)_i = \begin{bmatrix} 0 & \dots & . & \dots & 0 \\ . & & . & & . \\ \frac{\partial \mathcal{A}_{i,j}}{\partial \mathrm{w}_{i,1}} & \dots & \frac{\partial \mathcal{A}_{i,i}}{\partial \mathrm{w}_{i,j}} & \dots & \frac{\partial \mathcal{A}_{i,j}}{\partial \mathrm{w}_{i,J}} \\ . & & . & & . \\ 0 & \dots & . & \dots & 0 \end{bmatrix} \quad \mathbf{w}\hat{G}(\mathbf{w},\mathcal{A}) = \begin{bmatrix} \mathbf{w}_1\hat{g}_1 & \dots & \mathbf{w}_1\hat{g}_i & \dots & \mathbf{w}_1\hat{g}_I \\ . & & . & & . \\ . & & . & & . \\ . & & . & & . \\ \mathbf{w}_I\hat{g}_1 & \dots & \mathbf{w}_I\hat{g}_i & \dots & \mathbf{w}_I\hat{g}_I \end{bmatrix}.$$

$$\mathbf{w}\hat{G}(\mathbf{w},\mathcal{A})\left(\frac{\partial \mathcal{A}}{\partial \mathbf{w}}\right)_i = \begin{bmatrix} \mathbf{w}_1\hat{g}_i\frac{\partial \mathcal{A}_{i,i}}{\partial \mathrm{w}_{i,1}} & \dots & \dots & \mathbf{w}_1\hat{g}_i\frac{\partial \mathcal{A}_{i,i}}{\partial \mathrm{w}_{i,j}} \\ . & & & . \\ \mathbf{w}_i\hat{g}_i\frac{\partial \mathcal{A}_{i,i}}{\partial \mathrm{w}_{i,1}} & \dots & \dots & \mathbf{w}_i\hat{g}_i\frac{\partial \mathcal{A}_{i,i}}{\partial \mathrm{w}_{i,j}} \\ . & & & . \\ \mathbf{w}_I\hat{g}_i\frac{\partial \mathcal{A}_{i,i}}{\partial \mathrm{w}_{i,1}} & \dots & \dots & \mathbf{w}_I\hat{g}_i\frac{\partial \mathcal{A}_{i,i}}{\partial \mathrm{w}_{i,J}} \end{bmatrix}$$

$$\hat{\mathbf{w}}^{t+1} = \mathbf{w}^{t+1} - \eta_2\lambda \begin{bmatrix} \hat{g}_1^t \sum_{j=1}^{J} \frac{\partial \mathcal{A}_{1,1}^t}{\partial \mathrm{w}_{1,j}^t} \\ . \\ . \\ . \\ \hat{g}_I^t \sum_{j=1}^{J} \frac{\partial \mathcal{A}_{I,I}^t}{\partial \mathrm{w}_{I,j}^t} \end{bmatrix} \circledast \begin{bmatrix} \mathbf{w}_1^t \\ . \\ . \\ . \\ \mathbf{w}_I^t \end{bmatrix}$$

$$= \mathbf{w}^{t+1} + \eta_2\lambda \boldsymbol{d}^t \circledast \mathbf{w}^t,$$

4. To estimate $d^t = -[\hat{g}_1^t \sum_{j=1}^{J} \frac{\partial \mathcal{A}_{1,1}^t}{\partial \mathrm{w}_{1,j}^t}, \dots, \hat{g}_I^t \sum_{j=1}^{J} \frac{\partial \mathcal{A}_{I,I}^t}{\partial \mathrm{w}_{I,j}^t}]^T$

$$\hat{\mathbf{w}}^{t+1} = \mathbf{w}^{t+1} + U^t \circ DReLU(\mathbf{w}^t, \mathcal{A}^t),$$

$$\mathbf{w}^{t+1} \leftarrow \hat{\mathbf{w}}^{t+1},$$

5. Definition of Density-ReLU

$$DReLU(\mathbf{w}_i, \mathcal{A}_i) = \begin{cases} \mathbf{w}_i & if\ (\neg D(\mathbf{w}_i')) \wedge D(\mathcal{A}_i) = 1, \\ 0 & otherwise, \end{cases}$$

$$D(\boldsymbol{x}_i) = \begin{cases} 1 & if\ ranking(\sigma(\boldsymbol{x})_i) > \mathcal{T}, \\ 0 & otherwise, \end{cases}$$

6. Update learnable step size U

$$U^{t+1} = |U^t - \eta_3 \frac{\partial L}{\partial U^t}|,$$

$$\frac{\partial L}{\partial U^t} = \frac{\partial L_S}{\partial \mathbf{w}^t} \circ DReLU(\mathbf{w}^{t-1}, \mathcal{A}^t),$$

Algorithm

Algorithm 1 RBONN training.

Input: a minibatch of inputs and their labels, real-valued weights $\mathbf{w}$, recurrent model weights U, scale factor matrix $\mathcal{A}$, learning rates η_1, η_2 and η_3.
Output: updated real-valued weights $\mathbf{w}^{t+1}$, updated scale factor matrix $\mathcal{A}^{t+1}$, and updated recurrent model weights U^{t+1}.

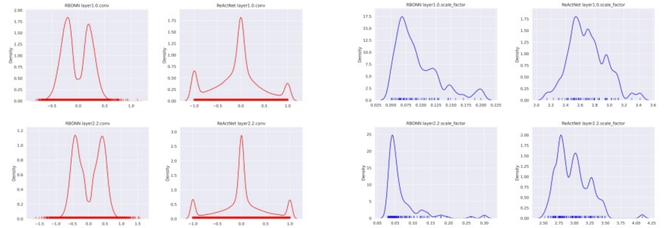
1: **while** Forward propagation **do**
2: $\mathbf{b}^{\mathbf{w}^t} \leftarrow \text{sign}(\mathbf{w}^t)$.
3: $\mathbf{b}^{\mathbf{a}_{in}^t} \leftarrow \text{sign}(\mathbf{a}_{in}^t)$.
4: Features calculation using Eq. 3
5: Loss calculation using Eq. 4
6: **end while**
7: **while** Backward propagation **do**
8: Computing $\frac{\partial L}{\partial \mathcal{A}^t}$, $\frac{\partial L}{\partial \mathbf{w}^t}$, and $\frac{\partial L}{\partial U^t}$ using Eq. 6 , 7 and 18.
9: Update $\mathcal{A}^{t+1}$, $\mathbf{w}^{t+1}$, and U^{t+1} according to Eqs. 5, 13, and 17, respectively.
10: **end while**

Conclusion

- We propose a new learning algorithm, termed recurrent bilinear optimization, to efficiently calculate BNNs, which is the first attempt to optimize BNNs from the bilinear perspective.
- Our method specifically introduces recurrent optimization to sequentially backtrack the sparse real-valued weight filters, which can be sufficiently trained and reach their performance limit based on a controllable learning process.
- RBONNs show strong generalization to gain impressive performance on both image classification and object detection tasks, demonstrating the superiority of the proposed method over state-of-the-art BNNs.

Experiments

✓ **Effectiveness of RBONN**



(a) Weight distribution (b) Scale matrix distribution

✓ **State-of-the-art on image classification using ImageNet**

Network	Method	W/A	OPs ($\times 10^8$)	Top-1	Top-5
ResNet-18	Real-valued	32/32	18.19	69.6	89.2
	DoReFa-Net	1/4	2.44	59.2	81.5
	TBN	1/2	1.81	55.6	79.0
	BNN	1/1	1.63	42.2	67.1
	XNOR-Net			51.2	73.2
	Bi-Real Net			56.4	79.5
	IR-Net			58.1	80.0
	BONN			59.3	81.6
	RBCN			59.5	81.6
	RBNN			59.6	81.6
	RBONN			**61.4**	**83.5**

✓ **State-of-the-art on object detection using COCO**

Framework	Input Resolution	Backbone	Method	mAP @[.5, .95]	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
Faster R-CNN	1000×600	ResNet-18	Real-valued	26.0	44.8	27.2	10.0	28.9	39.7
			DeRoFa-Net	22.9	38.6	23.7	8.0	24.9	36.3
			Xnor-Net	10.4	21.6	8.8	2.7	11.8	15.9
			Bi-Real Net	14.4	29.0	13.4	3.7	15.4	24.1
			BiDet	15.7	31.0	14.4	4.9	16.7	25.4
			RBONN	**20.6**	**37.3**	**19.9**	**7.4**	**21.3**	**32.8**
SSD	300×300	VGG-16	Real-valued	23.2	41.2	23.4	5.3	23.2	39.6
			DoReFa-Net	19.5	35.0	19.6	5.1	20.5	32.8
			XNOR-Net	8.1	19.5	5.6	2.6	8.3	13.3
			Bi-Real Net	11.2	26.0	8.3	3.1	12.0	18.3
			BiDet	13.2	28.3	10.5	5.1	14.3	20.5
			RBONN	**17.4**	**33.2**	**16.4**	**5.3**	**17.1**	**26.7**