* Some slides are borrowed from AI Robotics KR
https://drive.google.com/file/d/1bz3C-fFVSCrOdnbi-8lf_2NS1yhpGdVO

XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, Ali Farhadi

2021. 06. 04.
Byungyun KONG

KiSTi www.kisti.re.kr 한국과학기술정보연구원 Korea Institute of Science and Technology Information

UST 과학기술연합대학원대학교 UNIVERSITY OF SCIENCE & TECHNOLOGY

Contents

- Introduction
- Binary-Weight-Networks
- XNOR-Networks
- Training Networks
- Evaluation
- Conclusion

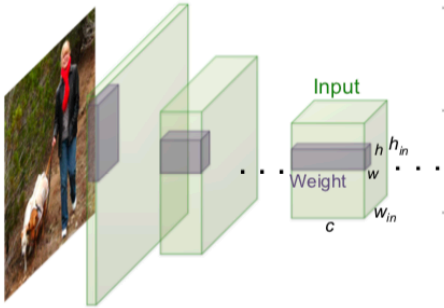
Introduction

- CNN-based recognition systems need large amounts of memory and computational power
  - AlexNet has 61M parameters (249 MB of Memory)
  - Needs 1.5B high precision operations per image
  - These numbers are even higher for deeper CNNs e.g.,VGG

    ➢ **Unsuitable for smaller devices like cell phones and embedded electronics**

- By binarization, one can get simple, efficient, and accurate approximations to CNNs
  - Weights binarization makes the weights significantly smaller (~x32)
  - Convolutions can be estimated by only addition and subtraction (~x2 speed up)
  - Input binarization makes the convolutions can be estimated by only XNOR and bitcount (~x58 speed up)

Introduction



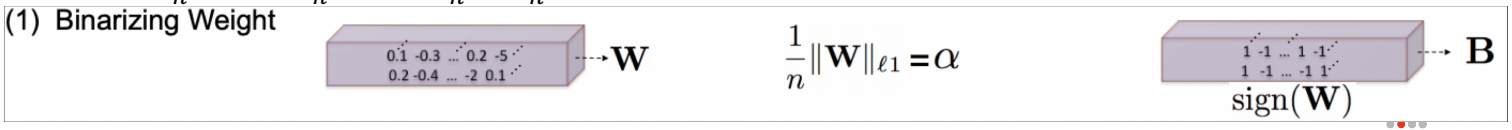
	Network Variations	Operations used in Convolution	Memory Saving (Inference)	Computation Saving (Inference)	Accuracy on ImageNet (AlexNet)
Standard Convolution	Real-Value Inputs 0.11 -0.21 ... -0.34 / -0.25 0.61 ... 0.52; Real-Value Weights 0.12 -1.2 ... 0.41 / -0.2 0.5 ... 0.68	+ , − , ×	1x	1x	%56.7
Binary Weight	Real-Value Inputs 0.11 -0.21 ... -0.34 / -0.25 0.61 ... 0.52; Binary Weights 1 -1 ... 1 / -1 1 ... 1	+ , −	~32x	~2x	%56.8
BinaryWeight Binary Input **(XNOR-Net)**	Binary Inputs 1 -1 ... -1 / -1 1 ... 1; Binary Weights 1 -1 ... 1 / -1 1 ... 1	XNOR , bitcount	~32x	~58x	%44.2

Fig. 1: We propose two efficient variations of convolutional neural networks. **Binary-Weight-Networks**, when the weight filters contains binary values. **XNOR-Networks**, when both weigh and input have binary values. These networks are very efficient in terms of memory and computation, while being very accurate in natural image classification. This offers the possibility of using accurate vision techniques in portable devices with limited resources.

Binary-Weight-Networks

- Constrain a convolutional neural network to have binary weights
  - Estimate the real-value weight filter W ∈ W
  - Using a binary filter B ∈ $\{+1, -1\}^{c \times w \times h}$ and a scaling factor α ∈ $\mathbb{R}^+$ such that W ≈ αB
  - A convolutional operation can be approximated by I ∗ W ≈ $(I \oplus B)\alpha$
    ⊕ indicates a convolutional without any multiplication

- One can find optimal B^* and α^* by minimizing $J(B, \alpha) = \|W - \alpha B\|^2$
  - $J(B, \alpha) = \alpha^2 B^T B - 2\alpha W^T B + W^T W$
    $= \alpha^2 n - 2\alpha W^T B + W^T W$ (where $n = c \times w \times h$)
  - $B^* = \text{argmax}_B \{W^T B\}\ s.t.\ \text{B} \in \{+1, -1\}^{c \times w \times h}$
    $= sign(W)$
  - $\alpha^* = \text{argmin}_\alpha \{\alpha^2 n - 2\alpha W^T B\}$ (after derivative $2\alpha n - 2W^T B$)
    $= \frac{W^T B^*}{n} = \frac{W^T sign(W)}{n} = \frac{\sum |W_i|}{n} = \frac{1}{n}\|w\|_{l1}$



XNOR-Networks

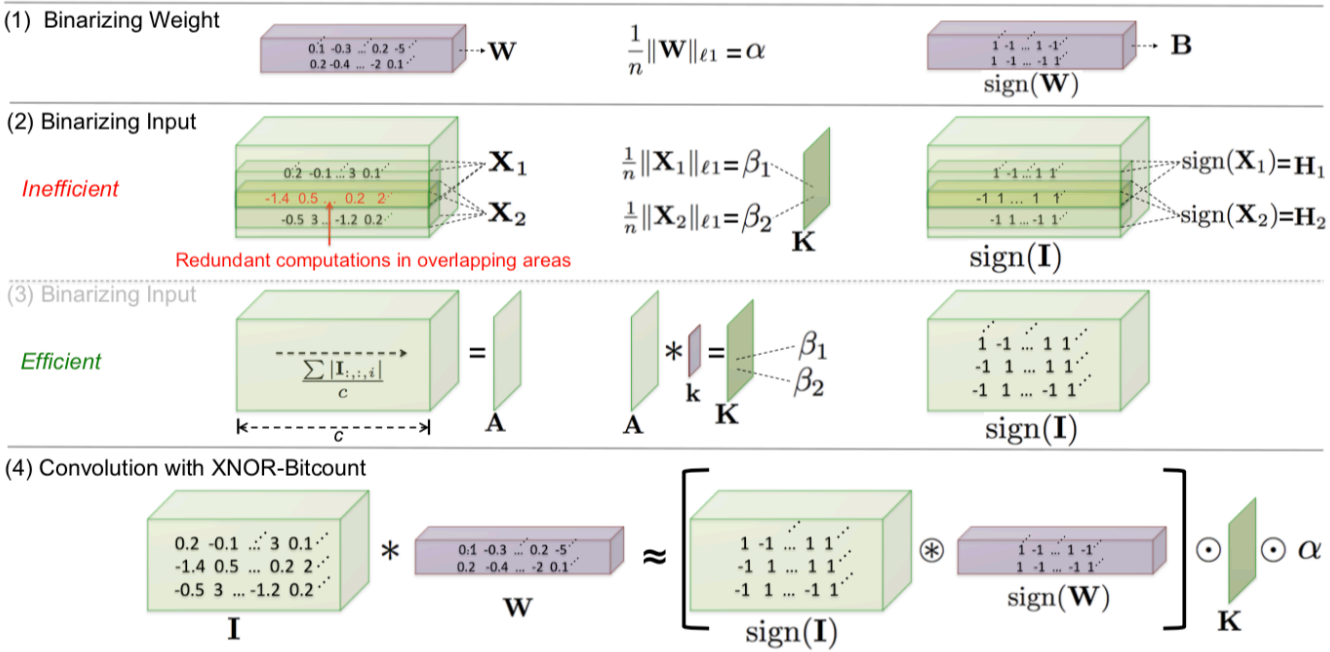
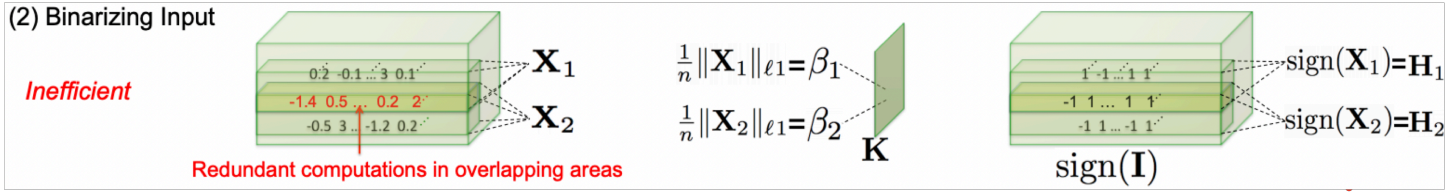


Fig. 2: This figure illustrates the procedure explained in section 3.2 for approximating a convolution using binary operations.

XNOR-Networks

- Binarizing both weight and input makes convolutions can be implemented using XNOR and bitcount
  - Natural extension of weight binarization
  - With approximation of $X = \beta H$ and $W = \alpha B$ where $H, B \in \{+1, -1\}^n$, scaling factor $\alpha, \beta \in \mathbb{R}^+$
- One can find optimal α^*, B^*, β^* and H^* by minimizing $\|X \odot W - \beta\alpha H \odot B\|$
  - Let $Y_i = X_i W_i$, $C_i = H_i B_i$, and $\gamma = \beta\alpha$
  - This is a same approximation of weight binarization
  - $C^* = sign(Y) = sign(X) \odot sign(W) = H^* B^*$
  - $\gamma^* = \frac{\sum|Y_i|}{n} = \frac{\sum|X_i||W_i|}{n} = \left(\frac{1}{n}\|X\|_{l1}\right)\left(\frac{1}{n}\|W\|_{l1}\right) = \beta^*\alpha^*$
    (Approximation is done with the independency between X_i and W_i)

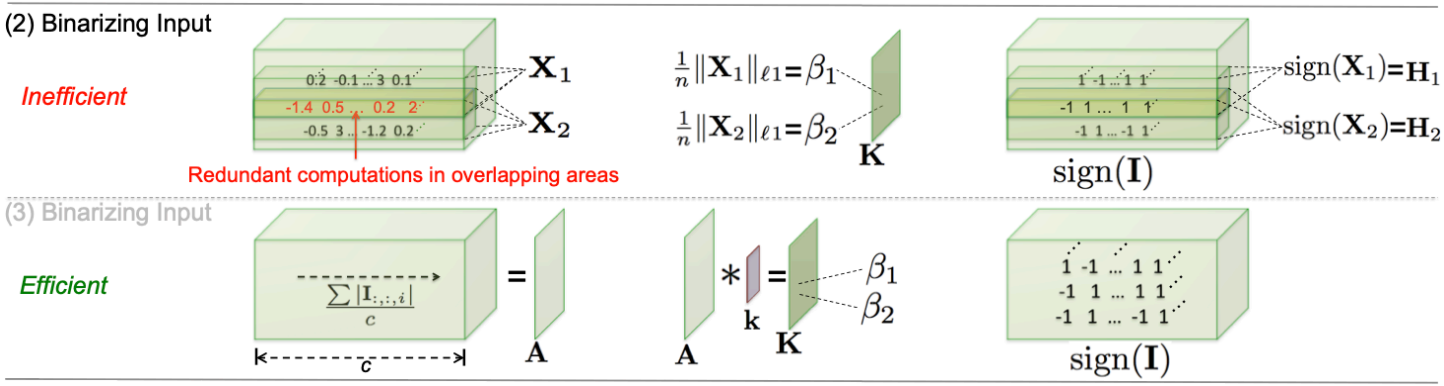
$$J(B, \alpha) = \|W - \alpha B\|^2$$
$$B^* = sign(W)$$
$$\alpha^* = \frac{\sum|W_i|}{n} = \frac{1}{n}\|W\|_{l1}$$



XNOR-Networks

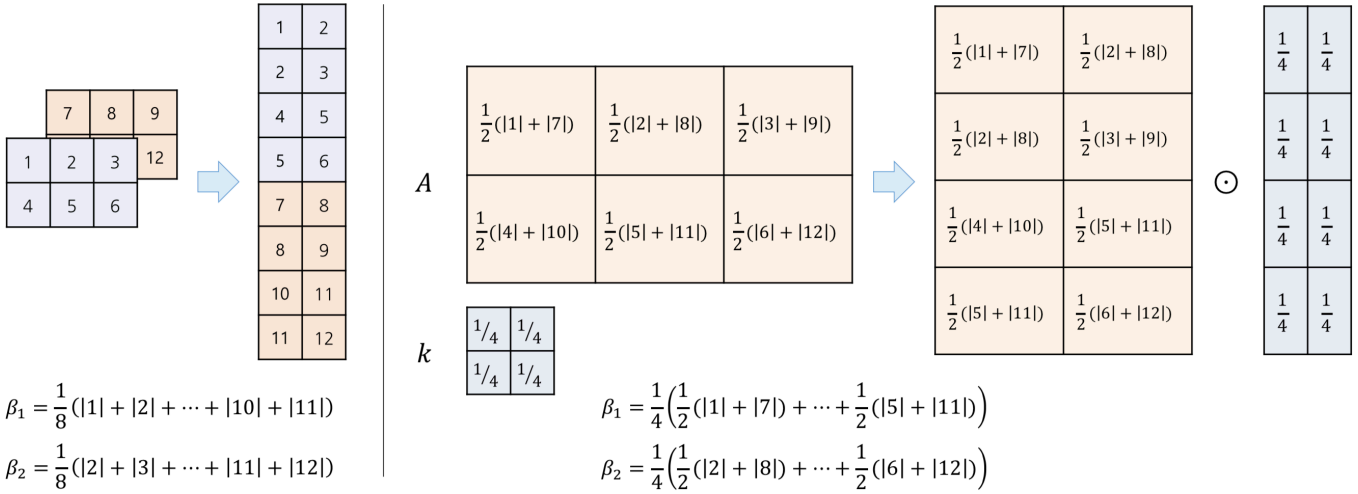
- $\mathrm{I} \in \mathbb{R}^{c \times w_{in} \times h_{in}}$

- $A = \frac{\sum |\mathrm{I}_{:,:,i}|}{n}$
- $\mathrm{k} \in \mathbb{R}^{w \times h}$, $k = \frac{1}{w \times h}$



XNOR-Networks

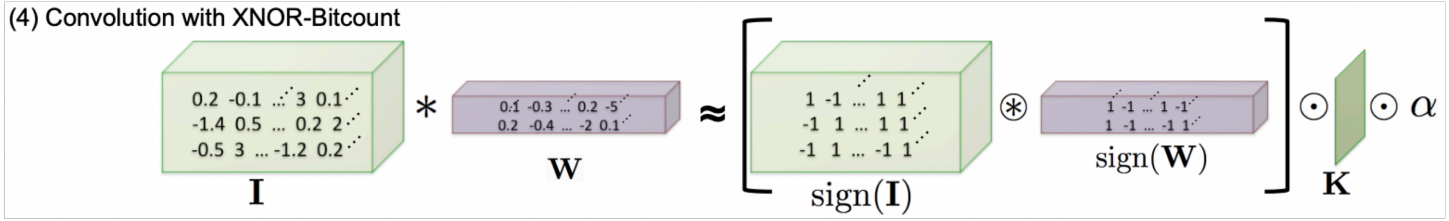
- Efficient input binarization can be done with channel direction absolute input value averaging



- Convolving A with k gives same result

XNOR-Networks

- Now, one can approximate the convolution between input I and weight filter W mainly using binary operations
  - $I * W \approx \left(sign(I) \circledast sign(W)\right) \odot K\alpha$
  - $\circledast$ indicates a convolutional operation using XNOR and bitcount



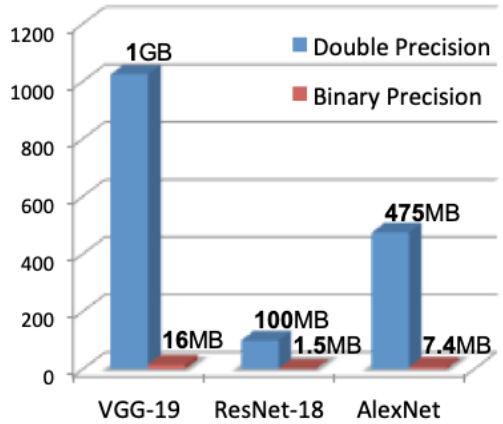
Training Networks

- To prevent loss of information, changed CNN block structure
  - Applying pooling to binary input results in tensors with +1 almost everywhere
  - Normalizing input before binarization reduces information loss caused by thresholding at zero

- k-bit quantization is available
  - $q_k(x) = 2\left(\frac{\left[(2^k-1)\left(\frac{x+1}{2}\right)\right]}{2^k-1} - \frac{1}{2}\right)$, $where$ $[.]$indicates rounding

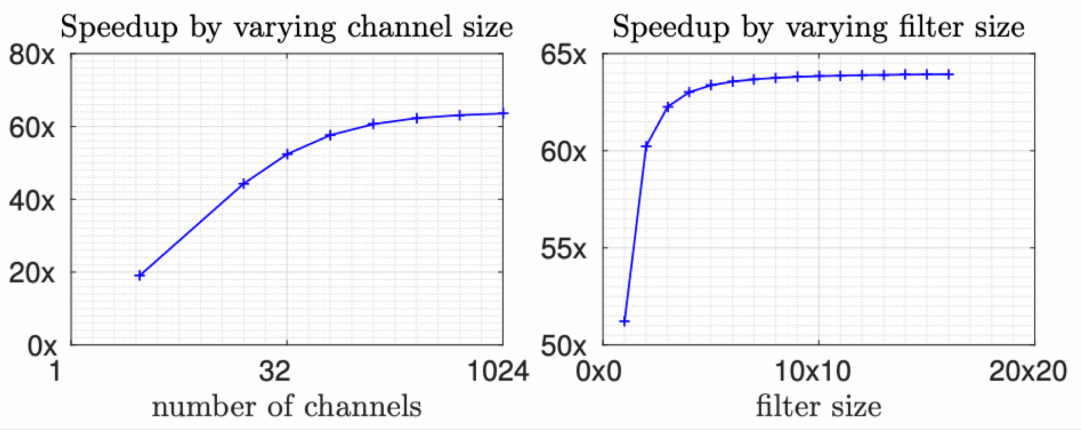
Evaluation - Efficiency

- the required memory for three different CNN architectures (AlexNet, VGG-19, ResNet-18)



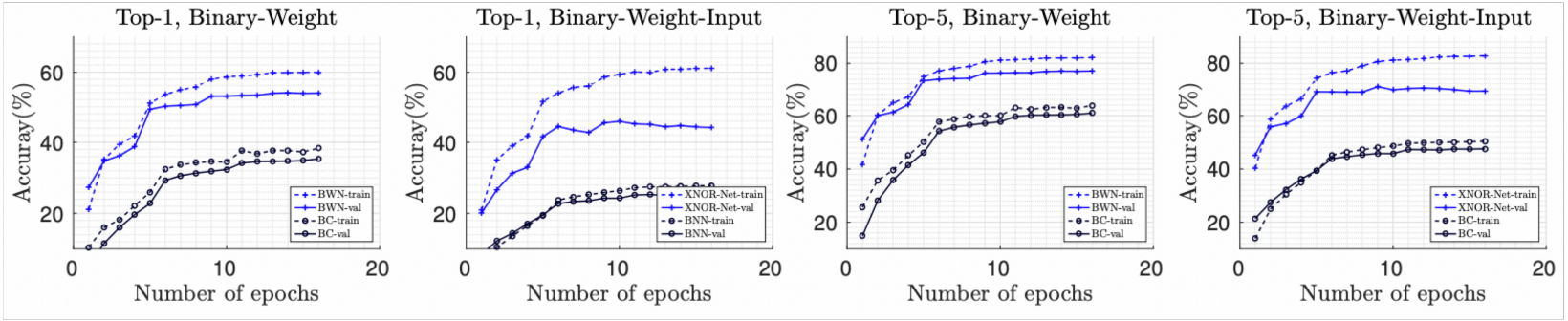
Evaluation - Efficiency

- Speed up
  - Standard convolution needs $c \times N_W \times N_I$ operations where $N_W = wh$ and $N_I = w_{in} h_{in}$
  - Binary convolution needs $c \times N_W \times N_I$ binary operations and N_I non-binary operations
  - One can perform 64 binary operations in one clock of CPU
  - $Speed\ up = \frac{c \times N_W \times N_I}{\frac{1}{64} \times c \times N_W \times N_I + N_I} = \frac{64 \times c \times N_W}{c \times N_W + 64}$
  - Compare to ResNet, 58× speed up is achieved (62.27× theoretical speed up)



Evaluation – Image Classification

- Vs. BinaryConnect & BinaryNeuralNetwork

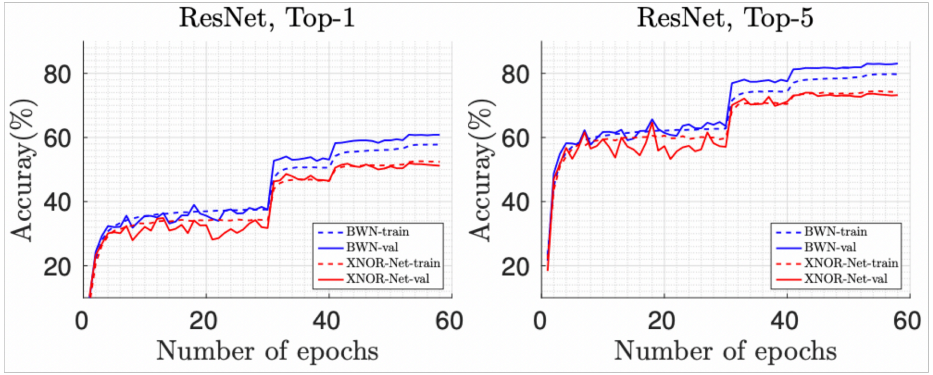


- Vs. AlexNet

Classification Accuracy(%)									
Binary-Weight				Binary-Input-Binary-Weight				Full-Precision	
BWN		BC[11]		XNOR-Net		BNN[11]		AlexNet[1]	
Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
56.8	**79.4**	35.4	61.0	**44.2**	**69.2**	27.9	50.42	56.6	80.2

Evaluation – Image Classification

- ImageNet accuracy



- Vs. ResNet & GoogLeNet

	ResNet-18		GoogLenet	
Network Variations	top-1	top-5	top-1	top-5
Binary-Weight-Network	60.8	83.0	65.5	86.1
XNOR-Network	51.2	73.2	N/A	N/A
Full-Precision-Network	69.3	89.2	71.3	90.0

Evaluation - Ablation Studies

- Effect of learned α
  - α as a training parameter performed worse

Binary-Weight-Network		
Strategy for computing α	top-1	top-5
Using equation 6	56.8	79.4
Using a separate layer	46.2	69.5

$$\alpha^* = \frac{1}{n}\|w\|_{l1}$$

- Effect of block structure
  - Typical block structure is not suitable for binarization

XNOR-Network		
Block Structure	top-1	top-5
C-B-A-P	30.3	57.5
B-A-C-P	44.2	69.2

Convolutional
BatchNormalization
Active function (here binary activation)
Pooling respectively

Conclusion

- Binarization can reduce the size of network by ~32×
- XNOR-net achieved ~58× speed up
- Provided the possibility of loading very deep neural networks into portable devices
- Provided the possibility of running the inference on CPU in real time

Thank You

Training Networks

Algorithm 1 Training an L-layers CNN with binary weights:

Input: A minibatch of inputs and targets $(\mathbf{I}, \mathbf{Y})$, cost function $C(\mathbf{Y}, \hat{\mathbf{Y}})$, current weight $\mathcal{W}^t$ and current learning rate η^t.

Output: updated weight $\mathcal{W}^{t+1}$ and updated learning rate η^{t+1}.

1: Binarizing weight filters:
2: **for** $l = 1$ to L **do**
3: **for** k^{th} filter in l^{th} layer **do**
4: $\mathcal{A}_{lk} = \frac{1}{n}\|\mathcal{W}^t_{lk}\|_{\ell 1}$
5: $\mathcal{B}_{lk} = \text{sign}(\mathcal{W}^t_{lk})$
6: $\widetilde{\mathcal{W}}_{lk} = \mathcal{A}_{lk}\mathcal{B}_{lk}$
7: $\hat{\mathbf{Y}} =$ **BinaryForward**$(\mathbf{I}, \mathcal{B}, \mathcal{A})$ // standard forward propagation except that convolutions are computed using equation 1 or 11
8: $\frac{\partial C}{\partial \widetilde{\mathcal{W}}} =$ **BinaryBackward**$(\frac{\partial C}{\partial \hat{\mathbf{Y}}}, \widetilde{\mathcal{W}})$ // standard backward propagation except that gradients are computed using $\widetilde{\mathcal{W}}$ instead of $\mathcal{W}^t$
9: $\mathcal{W}^{t+1} =$ **UpdateParameters**$(\mathcal{W}^t, \frac{\partial C}{\partial \widetilde{\mathcal{W}}}, \eta_t)$ // Any update rules (*e.g.*,SGD or ADAM)
10: $\eta^{t+1} =$ **UpdateLearningrate**(η^t, t) // Any learning rate scheduling function

- $\frac{\partial C}{\partial W_i} = \frac{\partial C}{\partial \widetilde{W}_i}\frac{\partial \widetilde{W}_i}{\partial W_i}$

$$= \frac{\partial C}{\partial \widetilde{W}_i}\frac{\partial}{\partial W_i}\left(\frac{1}{n} \| W_i \|_1 * sign(W_i)\right)$$

$$= \frac{\partial C}{\partial \widetilde{W}_i}\left(\frac{1}{n} + \frac{\partial sign(W_i)}{\partial W_i}\alpha\right)$$

$$where \quad \frac{\partial sign(r)}{\partial r} = r\mathbb{I}_{|r|\leq 1}$$