MST-compression: Compressing and Accelerating Binary Neural Networks with Minimum Spanning Tree

Quang Hieu Vo Linh-Tam Tran Sung-Ho Bae Lok-Won Kim Choong Seon Hong
Department of Computer Science and Engineering, Kyung Hee University, South Korea

ICCV23 PARIS

Contribution

Motivation: Propose a comprehensive method from training to HW implementation to compress Binary Neural Networks (BNNs) inference with high throughput via reordering the output calculation for convolution and fully connected layers.

Key contributions:

- Analyze the role of Minimum Spanning Tree (MST) to reorder output calculation in BNNs.
- An algorithm toward reducing MST distance right at training phase, leading to inference computation reduction.
- A supportive hardware accelerator for MST compression method.

Binary Convolution

BNNs binarizes parameters and activations using Eq. (4).

$$x^b = \text{Sign}(x) = \begin{cases} +1, & \text{if } x \geq 0, \\ -1, & \text{otherwise.} \end{cases} \quad (4)$$

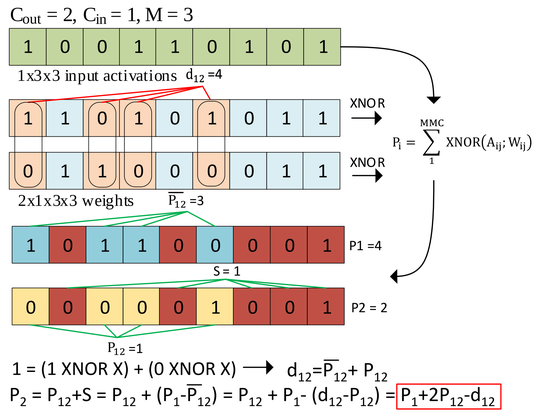
Given a binary convolution, output of the channel ith can be computed using Eq. (5).

$$Y_i = (2 \sum_{j=1}^{C_{in}MM} \text{XNOR}(\mathcal{A}_{ij}^b, \mathcal{W}_{ij}^b) - C_{in} \times M \times M) \odot \alpha, \quad (5)$$

The same input activation is used to compute all output channels, while $(1 \,\text{XNOR}\, x) + (0 \,\text{XNOR}\, x)$ is always 1. We can calculate the output channel j^{th} as in Eq. (6).

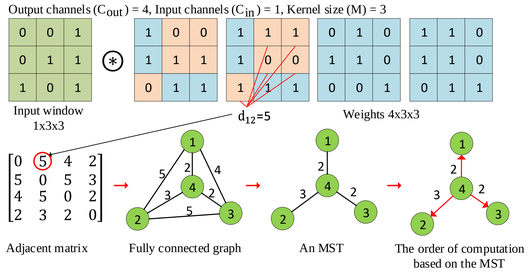
$$Y_j = 2(P_i - d_{ij} + 2P_{ij}) - C_{in} \times M \times M. \quad (6)$$

Relation between two output channels in a binary convolution layer



MST-compression for Inference BNN Acceleration

MST exploration for a conv computation order



- According to parameter set, an adjacent matrix (A) is constructed, where A_{ij} is the Hamming distance between 2 weight sets ($C_{in} \times M \times M$) corresponding to two output channels i^{th} and j^{th}.
- A fully connected graph is constructed based on matrix A.
- A MST is explored and then the MST with the smallest depth is selected for reodering computatio.

Compression ratio

$$\mathcal{R} = \frac{\sum_{j \neq root}^{C_{out}} d_{ij} + C_{in} \times M \times M}{C_{out} \times C_{in} \times M \times M}, \quad (1)$$

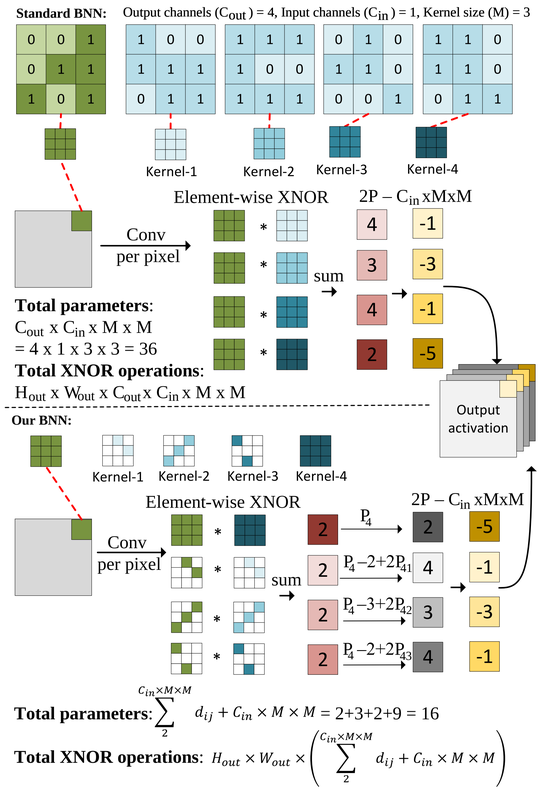
Learning optimization

$$\mathcal{C}(\text{w}_b^{li}) = \underset{x}{\text{argmin}} \left(\sum_{i=1}^{C_{in} \times M \times M} \|x - \text{w}_b^{li}\| \right); x \in \mathbb{C}_l. \quad (2)$$

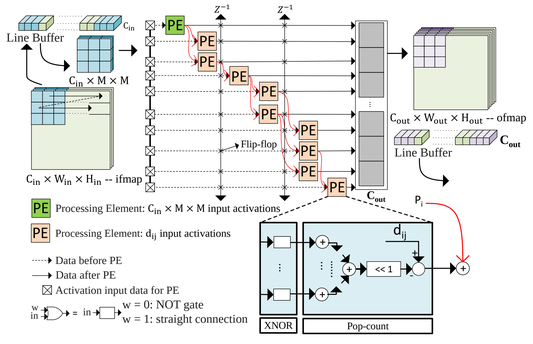
$$\mathcal{L}(\text{input}, \text{w}) = \mathcal{L}_0(\text{input}, \text{w}) + \lambda\gamma \sum_i \|\mathcal{C}(\text{w}_b^i) - \text{w}_b^i\|^2, \quad (3)$$

- where $\text{w}_b^{li}, \mathbb{C}_l \subset \{\pm 1\}^{C_{in} \times M \times M}$ and $|\mathbb{C}_l| = N_l$ and $\mathcal{C}$ includes nerest centers of every w_b^{li}.

Output computaiton in a convolution



HW architecture for a binary convolution



Computation process in HW architecture

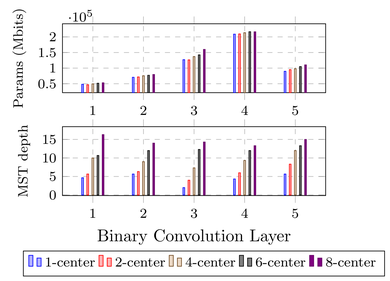
- **Loading input**: Image is loaded from DDR via DMA (Direct Memory Access) to BNN design. Line buffer with the size $C_{in} \times (W_{in} \times (M-1) + M)$ is valid when $C_{in} \times (W_{in} \times (M-2) + M - 1)$ is filled if the padding size = 1, or fully filled if there is no padding.
- **Computation process**: The computation is in order following the MST. The output channel corresponding to the root vertex is calculated first with $C_{in} \times M \times M$ XNOR operations. From the next one, output of a vertex is calculated based on the previous output corresponding to the parent vertex and the output of a specific number of XNOR euqualing to the distance to the parent vertex.
- **Output storage**: The whole design is fully implemented; thus ofmap is delivered to the next layer without intermediate memory.
- **Performance**: Images are loaded to the design every clock cycle; thus throughput is not affected by the inrease of layers.

Experiments & Results

Effect of the number of centers

N_l	MST depth	#Params (Mbit)	#Bit-Ops (GOps)	Top-1 Acc. mean ± std (%)
1	22.3	0.545	0.119	91.49 ± 0.04
2	30.3	0.550	0.118	91.45 ± 0.08
4	47.7	0.574	0.125	91.42 ± 0.06
6	60.3	0.581	0.130	91.53 ± 0.07
8	73.0	0.607	0.136	91.49 ± 0.04

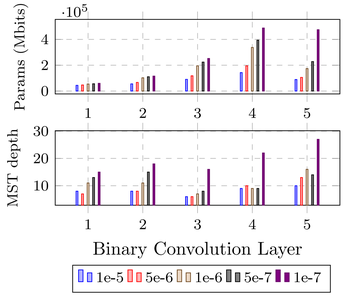
Performance *w.r.t.* different N_l on (Cifar-10 VGG-small).



Effect of hyper-parameter λ

λ	MST depth	#Params (Mbit)	#Bit-Ops (GOps)	Top-1 Acc. mean ± std (%)
1e-7	97.3	1.391	0.217	92.17 ± 0.07
5e-7	58.3	0.998	0.184	92.09 ± 0.06
1e-6	54.0	0.864	0.165	91.99 ± 0.07
5e-6	44.0	0.532	0.115	91.14 ± 0.08
1e-5	42.3	0.422	0.098	90.17 ± 0.14

Performance *w.r.t.* different λ on (Cifar-10 VGG-small).



Training comparison to SOTA

- **Ours 1**: After apply fine-tuning using the proposed learning optimization method.
- **Ours 2**: Apply only MST exploration and computation arrangement based on the MST for the inference without accuracy drop.

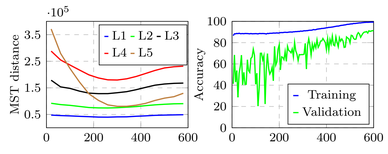
N-work	Method	#Params (Mbit)	#Bit-Ops (GOps)	Top-1 Acc. (%)
VGG small	RAD	4.571	0.603	90.0
	IR-Ne	4.571	0.603	90.4
	RBNN	4.571	0.603	91.3
	Adabin	4.571	0.603	92.3
	ReCU	4.571	0.603	92.4
	SNN	3.047	0.194	91.0
	Ours 1|2	0.556|1.611	0.122|0.232	91.5|93.3[1]
ResNet 18	RAD	10.99	0.547	90.5
	IR-Net	10.99	0.547	91.5
	RBNN	10.99	0.547	92.2
	ReCU	10.99	0.547	92.8
	Adabin	10.99	0.547	93.1
	SNN	7.324	0.289	91.0
	Ours 1|2	0.814|4.293	0.104|0.216	91.6|93.2[1]
ResNet 20	DSQ	0.267	0.040	84.1
	IR-Net	0.267	0.040	86.5
	RBNN	0.267	0.040	86.5
	ReCU	0.267	0.040	87.4
	Adabin	0.267	0.040	88.2
	SNN	0.178	0.040	85.1
	Ours 1|2	0.096|0.116	0.015|0.017	86.5|88.0[1]

(a) Comparison with the state-of-the-art methods on CIFAR-10.

N-work	Method	#Params (Mbit)	#Bit-Ops (GOps)	Top-1 Acc. (%)
ResNet 18	BNN+	10.99	1.677	53.0
	Bi-Real	10.99	1.677	56.4
	XNOR++	10.99	1.677	57.1
	IR-Net	10.99	1.677	58.1
	Adabin	10.99	1.677	63.1
	ReCU	10.99	1.677	61.0
	SNN	7.32	0.883	56.3
	Ours 1|2	3.43|4.84	0.636|0.716	57.0|61.2[1]
ResNet 34	Bi-Real	21.09	3.526	62.2
	IR-Net	21.09	3.526	62.9
	Adabin	21.09	3.526	66.4
	ReCU	21.09	3.526	65.1
	SNN	14.06	1.696	61.4
	Ours 1|2	9.44|9.51	1.550|1.558	62.9|65.4[1]

(b) Comparison with the state-of-the-art methods on ImageNet.
[1] Accuracy after fine-tuning is at https://github.com/z-hXu/ReCU.

Effect of hyper-parameter γ



a) MST distances of each layer (L). b) Accuracy curves.

HW comparison to SOTA designs

Design	Freq (MHz)	LUTs	Acc. (%)	FPS (K)	FPS/ LUTs
FINN	200	46,253	80.1	22	0.47
FINN	125	365,963	80.1	128	0.35
FINN-R	237	332,637	80.1	105	0.31
FINN-R	300	41,733	80.1	20	0.48
FINN	300	25,431	80.1	1.9	0.07
ReBNet	200	53,200	80.6	6	0.11
Streaming-Arc	210	290,012	80.2	205	0.70
Ours (K-mean)	210	201,434	80.5	205	1.01
Ours (MST)	**210**	**161,294**	**80.5**	**205**	**1.27**

HW performance comparison with SOTA designs-Cifar-10.

References

[1] ReCU: Reviving the dead weights in binary neural networks, ICCV 2021
[2] RAD: Regularizing activation distribution for training binarized deep networks, CVPR 2019
[3] IR-Net: Forward and backward information retention for accurate binary neural networks, CVPR 2020
[4] SNN: Sub-bit neural networks: Learning to compress and accelerate binary neural networks, ICCV 2021
[5] Streaming-Arc: A deep learning accelerator based on a streaming architecture for binary neural networks, IEEE Access 2022
[6] Finn: A framework for fast, scalable binarized neuralnetwork inference, ACM/SIGDA international symposium on FPGAs, 2017

Future Research

We have proposed a comprehensive compression method for BNNs from learning to accelerating with high throughput and resource efficiency purposes. In future work, to diversify the method's applications, we focus on improve the learning optimization that help the acceleration better compress. In terms of hardware implementation, architectures with limited number of processing elements would be considered to reduce the resources with acceptable throughput.