B⊕LD: Boolean Logic Deep Learning

Van Minh Nguyen, Cristian Ocampo, Aymen Askri, Louis Leconte, Ba-Hien Tran

NEURAL INFORMATION PROCESSING SYSTEMS

HUAWEI

Objective and Contributions

- A novel notion of **Boolean variation** and new mathematical framework of its calculus providing the chain rule similar to continuous gradient.
- A novel **Boolean logic backpropagation and optimization** method allowing for deep neural networks to **operate solely with Boolean logic** and to be **trained directly in Boolean domain.**
- **State-of-the-art results** compared to binarized neural networks, evaluated on challenging tasks with ConvNets, Transformers, ...
- **Significantly energy-efficient**: *both* training & inference.

Boolean Neural Networks

Boolean Neuron. Let L be a logic gate such as AND, OR, XOR, XNOR. The neuron's pre-activation output is given as follows:

$$s = w_0 + \sum_{i=1}^{m} \mathrm{L}(w_i, x_i), \quad \text{where } w_i, x_i \in \mathbb{B} := \{\mathrm{T}, \mathrm{F}\}\,. \tag{1}$$

Mixed Boolean-Real Neuron. For flexibility, we can extend to Boolean weights with real-valued inputs, and real-valued weights with Boolean inputs.

Forward Activation. $y = \mathrm{T}$ if $s \geq \tau$ and $y = \mathrm{F}$ if $s < \tau$ where s is the preactivation, τ is a fixed or learnable scalar threshold.

Boolean Training

Algorithm 1: Illustration with a FC layer.

Input : Learning rate η, number of iterations T;
Initialize: $m_{i,j}^{l,0} = 0$; $\beta^0 = 1$;
for $t = 0, \dots, T-1$ **do**
  Compute $x^{l+1,t}$ following Eq. 2; ; /* 1. Forward pass */
  Receive $\frac{\delta\mathrm{Loss}}{\delta x_{k,j}^{l+1,t}}$ from downstream layer; ; /* 2. Backward pass */
  Compute and backpropagate $g^{l,t}$ of Eq. 6; ; /* 2.1 Backpropagation */
  $N_{\mathrm{tot}} := 0$, $N_{\mathrm{unchanged}} := 0$; ; /* 2.2 Weight update process */
  foreach $w_{i,j}^l$ **do**
    Compute $q_{i,j}^{l,t+1}$ following Eq. 5;
    Update $m_{i,j}^{l,t+1} = \beta^t m_{i,j}^{l,t} + \eta^t q_{i,j}^{l,t+1}$;
    $N_{\mathrm{tot}} \leftarrow N_{\mathrm{tot}} + 1$;
    if $\mathbf{xnor}(m_{i,j}^{l,t+1}, w_{i,j}^{l,t}) = \mathrm{T}$ **then**
      $w_{i,j}^{l,t+1} = \neg w_{i,j}^{l,t}$, $m_{i,j}^{l,t+1} = 0$; /* invert */
    else
      $w_{i,j}^{l,t+1} = w_{i,j}^{l,t}$, $N_{\mathrm{unchanged}} \leftarrow N_{\mathrm{unchanged}} + 1$; /* keep */
    end
  end
  Update η^{t+1}, $\beta^{t+1} = N_{\mathrm{unchanged}}/N_{\mathrm{tot}}$;
end

Boolean Variation

Definition 1. Order relations in $\mathbb{B}$ are defined as: $\mathrm{F} < \mathrm{T}$, and $\mathrm{T} > \mathrm{F}$.

Definition 2. For $a, b \in \mathbb{B}$, the variation from a to b, denoted $\delta(a \to b)$, is defined as: $\delta(a \to b) \overset{\text{def}}{=} \mathrm{T}$ if $b > a$, $\overset{\text{def}}{=} 0$ if $b = a$, and $\overset{\text{def}}{=} \mathrm{F}$ if $b < a$.

Definition 3. For $f \in \mathcal{F}(\mathbb{B}, \mathbb{D})$, $\forall x \in \mathbb{B}$, write $\delta f(x \to \neg x) := \delta(f(x) \to f(\neg x))$. The variation of f w.r.t. x, i.e., $f'(x)$, is defined as: $f'(x) \overset{\text{def}}{=} \mathbf{xnor}(\delta(x \to \neg x), \delta f(x \to \neg x))$. Here, $\mathbb{D}$ is either a logic set $\mathbb{B}$ or a numeric set, e.g., $\mathbb{R}$ or $\mathbb{Z}$.

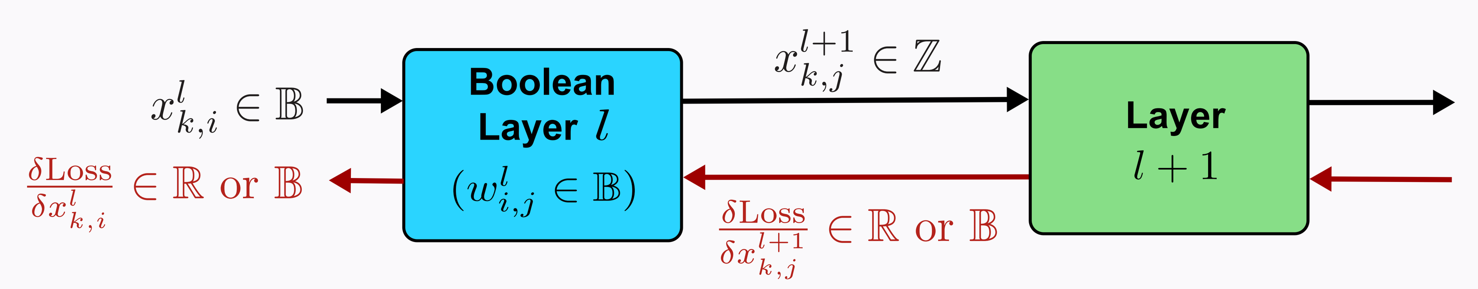
Remark. The variation of f w.r.t. x is T if f varies in same direction with x.

Definition 4. For $f \in \mathcal{F}(\mathbb{Z}, \mathbb{D})$, the variation of f w.r.t. $x \in \mathbb{Z}$ is defined as $f'(x) \overset{\text{def}}{=} \delta f(x \to x+1)$, where δf is the variation defined in $\mathbb{D}$.

Theorem 5. The following properties hold:

1. For $\mathbb{B} \xrightarrow{f} \mathbb{B} \xrightarrow{g} \mathbb{D}$: $(g \circ f)'(x) = \mathbf{xnor}(g'(f(x)), f'(x))$, $\forall x \in \mathbb{B}$.
2. For $\mathbb{B} \xrightarrow{f} \mathbb{Z} \xrightarrow{g} \mathbb{D}$, $x \in \mathbb{B}$, if $|f'(x)| \leq 1$ and $g'(f(x)) = g'(f(x) - 1)$, then: $(g \circ f)'(x) = \mathbf{xnor}(g'(f(x)), f'(x))$.

Boolean Backpropagation



Consider the Boolean l-th layer, which is assumed a fully-connected layer

$$x_{k,j}^{l+1} = w_{0,j}^l + \sum_{i=1}^{m} \mathrm{L}(x_{k,i}^l, w_{i,j}^l), \quad 1 \leq j \leq n, \tag{2}$$

where k is sample index, m and n are input and output sizes.

Remark. Layer l is connected to layer $l+1$ that can be an activation layer, a batch normalization, an arithmetic layer, or any others.

Atomic Variation. Consider $\mathrm{L} = \mathbf{xnor}$

$$q_{i,j,k}^l := \frac{\delta\mathrm{Loss}}{\delta w_{i,j}^l}\Big|_k = \mathbf{xnor}\Big(\frac{\delta\mathrm{Loss}}{\delta x_{k,j}^{l+1}}, \frac{\delta x_{k,j}^{l+1}}{\delta w_{i,j}^l}\Big) \overset{\mathbf{xnor}}{=} \mathbf{xnor}\Big(\frac{\delta\mathrm{Loss}}{\delta x_{k,j}^{l+1}}, x_{k,i}^l\Big), \tag{3}$$

$$g_{k,i,j}^l := \frac{\delta\mathrm{Loss}}{\delta x_{k,i}^l}\Big|_j = \mathbf{xnor}\Big(\frac{\delta\mathrm{Loss}}{\delta x_{k,j}^{l+1}}, \frac{\delta x_{k,j}^{l+1}}{\delta x_{k,i}^l}\Big) \overset{\mathbf{xnor}}{=} \mathbf{xnor}\Big(\frac{\delta\mathrm{Loss}}{\delta x_{k,j}^{l+1}}, w_{i,j}^l\Big). \tag{4}$$

Aggregation. Using the chain rules given by Theorem 5, we have

$$q_{i,j}^l := \frac{\delta\mathrm{Loss}}{\delta w_{i,j}^l} = \sum_k \mathbf{1}(q_{i,j,k}^l = \mathrm{T})|q_{i,j,k}^l| - \sum_k \mathbf{1}(q_{i,j,k}^l = \mathrm{F})|q_{i,j,k}^l|, \tag{5}$$

$$g_{k,i}^l := \frac{\delta\mathrm{Loss}}{\delta x_{k,i}^l} = \sum_j \mathbf{1}(g_{k,i,j}^l = \mathrm{T})|g_{k,i,j}^l| - \sum_j \mathbf{1}(g_{k,i,j}^l = \mathrm{F})|g_{k,i,j}^l|. \tag{6}$$

Boolean Optimizer. With $q_{i,j}^l$ obtained in Eq. 5, the rule for optimizing $w_{i,j}^l$ subjected to making the loss decreased is simply given according to its definition as:

$$\boxed{w_{i,j}^l = \neg w_{i,j}^l \text{ if } \mathbf{xnor}(q_{i,j}^l, w_{i,j}^l) = \mathrm{T}.} \tag{7}$$

Binarized Neural Networks — An Approximated Binary

They learn binary weights, $\mathbf{w}_{\mathrm{bin}}$, through *full-precision latent weights*, $\mathbf{w}_{\mathrm{fp}}$ using common gradient-descent methods, i.e.

$$\mathbf{w}_{\mathrm{bin}} = \mathrm{sign}(\mathbf{w}_{\mathrm{fp}} - \eta \cdot \mathbf{g}_{\mathbf{w}_{\mathrm{fp}}})$$

✗ No gains during the training!
✗ Sub-optimal performance due the approximations of latent weight gradient!

Experimental Results

Our method, B⊕LD, significantly outperforms the SOTA binarized neural networks (BNNs) both in terms of predictive performance and energy efficiency.

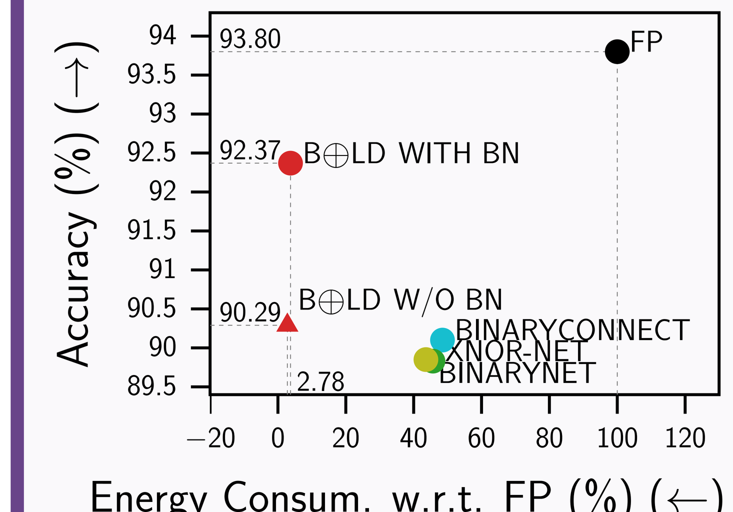


Figure: Comparisons of our method against notable BNNs on CIFAR10 with VGG-SMALL.

Table: Natural language understanding results with BERT models.

Method	GLUE Benchmark (Accuracy, ↑)								
	MNLI	QQP	QNLI	SST-2	COLA	SST-B	MRPC	RTE	**Avg.**
FP BERT	84.9	91.4	92.1	93.2	59.7	90.1	86.3	72.2	83.9
BINARYBERT	35.6	66.2	51.5	53.2	0.0	6.1	68.3	52.7	41.0
BIBERT	66.1	84.8	72.6	88.7	25.4	33.6	72.5	57.4	63.2
BIT	77.1	82.9	85.7	87.7	25.1	71.1	79.7	58.8	71.0
B⊕LD	75.6	85.9	84.1	88.7	27.1	68.7	78.4	58.8	70.9

Table: Image classification results with RESNET18 on IMAGENET. 'Base' is the mapping dimension of 1st layer.

Training Modality	Method	Acc. (%)	Cons. (%) Tesla V100
FP BASELINE	RESNET18	69.7	100.00
	BINARYNET	42.2	−−
	XNOR-NET	51.2	−−
	B⊕LD + BN	51.8	3.87
FP SHORTCUT	BI-REALNET:18	56.4	32.99
LARGE MODELS	BI-REALNET:34	62.2	58.24
	BI-REALNET:152	64.5	−−
	MELIUS-NET:29	65.8	−−
	B⊕LD (Base 256)	**66.9**	24.45
KD: RESNET34	REAL2BINARY	65.4	−−
	REACTNET-RESNET18	65.5	77.89
	BNEXT:18	68.4	37.51
	B⊕LD + BN (Base 192))	65.9	16.91
	B⊕LD (Base 256)	**70.0**	24.45
KD: RESNET50	POKEBNN-RESNET18	65.2	−−

Table: Super-resolution results measured in PSNR (dB) (↑), using the EDSR baseline.

Task	Method	SET5	SET14	BSD100	URBAN100	DIV2K
×2	FULL EDSR (FP)	38.11	33.92	32.32	32.93	35.03
	SMALL EDSR (FP)	38.01	33.63	32.19	31.60	34.67
	B⊕LD	37.42	33.00	31.75	30.26	33.82
×3	FULL EDSR (FP)	34.65	30.52	29.25	−−	31.26
	SMALL EDSR (FP)	34.37	30.24	29.10	−−	30.93
	B⊕LD	33.56	29.70	28.72	−−	30.22
×4	FULL EDSR (FP)	32.46	28.80	27.71	26.64	29.25
	SMALL EDSR (FP)	32.17	28.53	27.62	26.14	29.04
	B⊕LD	31.23	27.97	27.24	25.12	28.36

Table: Image segmentation results.

Dataset	Model	mIoU (%) (↑)
CITYSCAPES	FP BASELINE	70.7
	BINARY DAD-NET	58.1
	B⊕LD	**67.4**
PASCAL VOC 2012	FP BASELINE	72.1
	B⊕LD	67.3

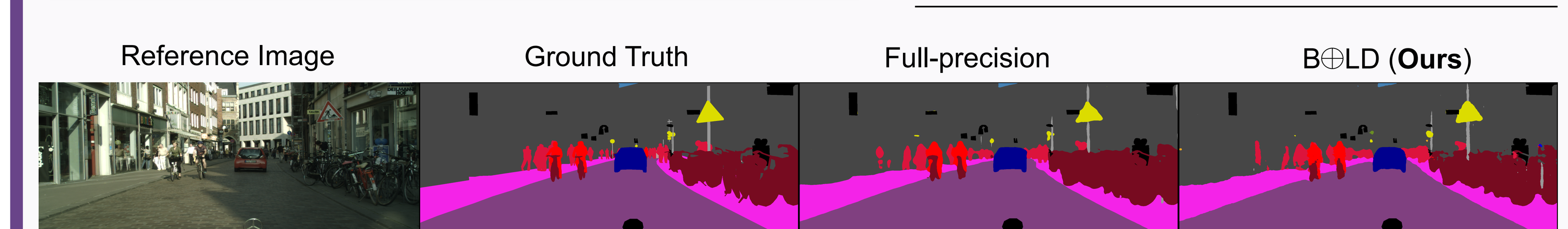


Figure: An example of CITYSCAPES.

References

[1] Nguyen. "Boolean Variation and Boolean Logic BackPropagation". *Arxiv 2023*
[2] Hubara et al. "Binarized Neural Networks". *NIPS 2016*