

Compacting Binary Neural Networks by Sparse Kernel Selection (Appendix)

Yikai Wang¹ Wenbing Huang² Yinpeng Dong^{1,3} Fuchun Sun¹ Anbang Yao⁴

¹BNRist Center, State Key Lab on Intelligent Technology and Systems,
Department of Computer Science and Technology, Tsinghua University

²Gaoling School of Artificial Intelligence, Renmin University of China ³RealAI ⁴Intel Labs China

{yikaiw, fcsun, dongyinpeng}@tsinghua.edu.cn, hwenbing@126.com, anbang.yao@intel.com

A. Proofs of Our Statements

Property 1 We denote $\mathbb{B} = \{-1, +1\}^{K \times K}$ as the dictionary of binary kernels. For each $\mathbf{w} \in \mathbb{R}^{K \times K}$, the binary kernel $\hat{\mathbf{w}}$ can be derived by a grouping process:

$$\hat{\mathbf{w}} = \text{sign}(\mathbf{w}) = \arg \min_{\mathbf{u} \in \mathbb{B}} \|\mathbf{u} - \mathbf{w}\|_2.$$

Proof. We denote $w(k_1, k_2)$ the entry of \mathbf{w} in the k_1 -th row and k_2 -th column, and similar denotation follows for \mathbf{u} . We have,

$$\begin{aligned} \arg \min_{\mathbf{u} \in \mathbb{B}} \|\mathbf{u} - \mathbf{w}\|_2^2 &= \arg \min_{\mathbf{u} \in \mathbb{B}} \sum_{k_1, k_2} |u(k_1, k_2) - w(k_1, k_2)|^2 \\ &= \left\{ \arg \min_{u(k_1, k_2) \in \{-1, +1\}} |u(k_1, k_2) - w(k_1, k_2)|^2 \right\}_{k_1, k_2} \\ &= \left\{ \text{sign}(w(k_1, k_2)) \right\}_{k_1, k_2} \\ &= \text{sign}(\mathbf{w}), \end{aligned}$$

which concludes the proof. □

Before the proofs for the lemma and theorem, in Figure 6, we provide the relationship of notations that are adopted in the main paper to facilitate the understanding of our permutation learning process.

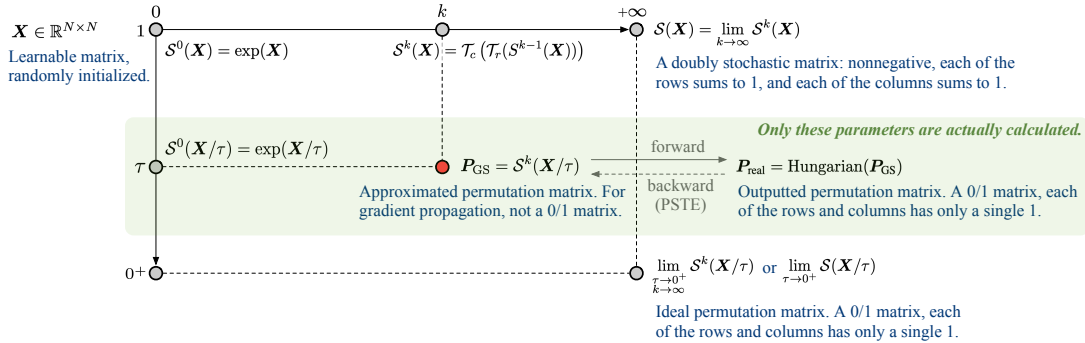


Figure 6. Relationship of notations for a better understanding. The horizontal axis and vertical axis stand for the values of k and τ , respectively. The Gumbel noise ϵ is omitted here. Only parameters in the green region are actually calculated during training, while the others are only for the understanding purpose. All these parameters are NOT needed during inference. Starting from a learnable matrix \mathbf{X} , we adopt a temperature τ and calculate $\mathcal{S}^0(\mathbf{X}/\tau)$. We then perform the Sinkhorn operation for k iterations to further obtain $\mathbf{P}_{\text{GS}} = \mathcal{S}^k(\mathbf{X}/\tau)$. We apply the Hungarian algorithm to further obtain \mathbf{P}_{real} . For sufficiently large k and small τ , \mathbf{P}_{real} equals the ideal permutation matrix.

Lemma 1 For sufficiently large k and small τ , we define the entropy of a doubly-stochastic matrix \mathbf{P} as $h(\mathbf{P}) = -\sum_{i,j} P_{i,j} \log P_{i,j}$, and denote the rate of convergence for the Sinkhorn operator as r ($0 < r < 1$)¹. There exists a convergence series s_τ ($s_\tau \rightarrow 0$ when $\tau \rightarrow 0^+$) that satisfies

$$\|\mathbf{P}_{\text{real}} - \mathbf{P}_{\text{GS}}\|_2^2 = \mathcal{O}(s_\tau^2 + r^{2k}). \quad (1)$$

Proof. Let $\mathbf{X}_\tau = \mathbf{X}/\tau$ and $\mathbf{X}_0 = \lim_{\tau \rightarrow 0^+} \mathbf{X}_\tau$. In addition, follow the definitions in Equation 5 (main paper), we denote $\mathcal{S}^k(\cdot)$ the k -th iteration of the Sinkhorn output and $\mathcal{S}(\cdot) = \lim_{k \rightarrow \infty} \mathcal{S}^k(\cdot)$.

As proved by [7], the Sinkhorn operator has a rate of convergence r ($0 < r < 1$) with respect to k , where r always exists and is bounded by a value lower than 1. There is

$$\|\mathcal{S}^k(\mathbf{X}_\tau) - \mathcal{S}(\mathbf{X}_\tau)\|_2 \leq r \|\mathcal{S}^{k-1}(\mathbf{X}_\tau) - \mathcal{S}(\mathbf{X}_\tau)\|_2 \leq \dots \leq r^{k-1} \|\mathcal{S}^1(\mathbf{X}_\tau) - \mathcal{S}(\mathbf{X}_\tau)\|_2.$$

By the definition of $\mathcal{S}^k(\mathbf{X}_\tau)$ in Equation 5 (main paper), the values of $\mathcal{S}^k(\mathbf{X}_\tau)$ are located in $[0, 1]$ for all $k \geq 1$. In addition, $\mathcal{S}(\mathbf{X}_\tau)$ is a 0/1 matrix with exactly N ones, where N is the dimension. Therefore, $\|\mathcal{S}^1(\mathbf{X}_\tau) - \mathcal{S}(\mathbf{X}_\tau)\|_2^2$ is well bounded and thus we obtain

$$\|\mathcal{S}^k(\mathbf{X}_\tau) - \mathcal{S}(\mathbf{X}_\tau)\|_2 \leq C_1 r^k,$$

where $C_1 > 0$ is a constant.

As mentioned in Equation 4 (main paper), $\mathcal{S}(\mathbf{X}_\tau)$ must be a doubly-stochastic matrix [17]. According to Lemma 3 in [13], if denoting $f_0(\cdot) = \langle \cdot, \mathbf{X} \rangle_F$, there is $|f_0(\mathcal{S}(\mathbf{X}_0)) - f_0(\mathcal{S}(\mathbf{X}_\tau))| \leq \tau(h(\mathcal{S}(\mathbf{X}_\tau)) - h(\mathcal{S}(\mathbf{X}_0))) = \tau h(\mathcal{S}(\mathbf{X}_\tau)) \leq \tau \max_{\mathbf{P} \in \mathcal{B}_N} (h(\mathbf{P}))$, where \mathcal{B}_N denotes the set of doubly stochastic matrices of dimension N .

As proved by Lemma 3 in [13], $|f_0(\mathcal{S}(\mathbf{X}_0)) - f_0(\mathcal{S}(\mathbf{X}_\tau))| \leq \tau \max_{\mathbf{P} \in \mathcal{B}_N} (h(\mathbf{P}))$ implies the convergence of $\mathcal{S}(\mathbf{X}_\tau)$ to $\mathcal{S}(\mathbf{X}_0)$ and there exists a convergence series s_τ ($s_\tau \rightarrow 0$ when $\tau \rightarrow 0^+$), satisfying $\|\mathcal{S}(\mathbf{X}_0) - \mathcal{S}(\mathbf{X}_\tau)\|_2 \leq C_2 s_\tau$, where $C_2 > 0$ is a constant.

Based on the triangle inequality, there is

$$\|\mathcal{S}(\mathbf{X}_0) - \mathcal{S}^k(\mathbf{X}_\tau)\|_2^2 \leq (C_2 s_\tau + C_1 r^k)^2 \leq 2C_2^2 s_\tau^2 + 2C_1^2 r^{2k}.$$

As mentioned in § 3.3 (main paper), there is $\mathbf{P}_{\text{GS}} = \mathcal{S}^k(\mathbf{X}_\tau)$ if we omit the noise term. Given the convergence property, for sufficiently large k and small τ , the Hungarian algorithm output \mathbf{P}_{real} equals to the real permutation, *i.e.* $\mathbf{P}_{\text{real}} = \mathcal{S}(\mathbf{X}_0)$. In summary, we have

$$\|\mathbf{P}_{\text{real}} - \mathbf{P}_{\text{GS}}\|_2^2 = \mathcal{O}(s_\tau^2 + r^{2k}),$$

which concludes the proof. \square

With the help of Lemma 1 and the inspiration of error-feedback framework [5], we now provide the detailed proof for Theorem 1.

Theorem 1 Assume that the training objective f w.r.t. \mathbf{P}_{GS} is L -smooth, and the stochastic gradient of \mathbf{P}_{real} is bounded by $\mathbb{E}\|\mathbf{g}(\mathbf{P}_{\text{real}})\|_2^2 \leq \sigma^2$. Denote the rate of convergence for the Sinkhorn operator as r ($0 < r < 1$) and the stationary point as \mathbf{P}_{GS}^* . Let the learning rate of PSTE be $\eta = \frac{c}{\sqrt{T}}$ with $c = \sqrt{\frac{f(\mathbf{P}_{\text{GS}}^0) - f(\mathbf{P}_{\text{GS}}^*)}{L\sigma^2}}$. For a uniformly chosen \mathbf{u} from the iterates $\{\mathbf{P}_{\text{real}}^0, \dots, \mathbf{P}_{\text{real}}^T\}$, concretely $\mathbf{u} = \mathbf{P}_{\text{real}}^t$ with the probability $p_t = \frac{1}{T+1}$, it holds in expectation over the stochasticity and the selection of \mathbf{u} :

$$\mathbb{E}\|\nabla f(\mathbf{u})\|_2^2 = \mathcal{O}\left(\sigma \sqrt{\frac{f(\mathbf{P}_{\text{GS}}^0) - f(\mathbf{P}_{\text{GS}}^*)}{T/L}} + L^2 (s_\tau^2 + r^{2k})\right). \quad (2)$$

Proof. Since the objective function f is L -smooth, $\mathbf{g}(\mathbf{P}_{\text{GS}}) = \mathbf{g}(\mathbf{P}_{\text{real}})$ in our PSTE, and $\mathbf{P}_{\text{GS}}^{t+1} = \mathbf{P}_{\text{GS}}^t - \eta \mathbf{g}(\mathbf{P}_{\text{real}}^t)$, we can obtain the following derivations,

$$\begin{aligned} f(\mathbf{P}_{\text{GS}}^{t+1}) &\leq f(\mathbf{P}_{\text{GS}}^t) + \langle \mathbf{P}_{\text{GS}}^{t+1} - \mathbf{P}_{\text{GS}}^t, \nabla f(\mathbf{P}_{\text{GS}}^t) \rangle + \frac{L}{2} \|\mathbf{P}_{\text{GS}}^{t+1} - \mathbf{P}_{\text{GS}}^t\|_2^2 \\ &= f(\mathbf{P}_{\text{GS}}^t) - \eta \langle \mathbf{g}(\mathbf{P}_{\text{real}}^t), \nabla f(\mathbf{P}_{\text{GS}}^t) \rangle + \frac{L\eta^2}{2} \|\mathbf{g}(\mathbf{P}_{\text{real}}^t)\|_2^2. \end{aligned}$$

¹The Sinkhorn operator has a rate of convergence r bounded by a value lower than 1 as proved by [7].

We use \mathbb{E} to represent the expectation with respect to the stochasticity. Based on the bound of the stochastic gradient, *i.e.* $\mathbb{E}\|\mathbf{g}(\mathbf{P}_{\text{real}}^t)\|_2^2 \leq \sigma^2$, and a natural property $\langle \mathbf{x}, \mathbf{y} \rangle \leq \frac{1}{2}\|\mathbf{x}\|_2^2 + \frac{1}{2}\|\mathbf{y}\|_2^2$, it holds that,

$$\begin{aligned} \mathbb{E} [f(\mathbf{P}_{\text{GS}}^{t+1} | \mathbf{P}_{\text{GS}}^t)] &\leq f(\mathbf{P}_{\text{GS}}^t) - \eta \langle \mathbb{E}[\mathbf{g}(\mathbf{P}_{\text{real}}^t)], \nabla f(\mathbf{P}_{\text{GS}}^t) \rangle + \frac{L\eta^2}{2} \mathbb{E}\|\mathbf{g}(\mathbf{P}_{\text{real}}^t)\|_2^2 \\ &\leq f(\mathbf{P}_{\text{GS}}^t) - \eta \langle \nabla f(\mathbf{P}_{\text{real}}^t), \nabla f(\mathbf{P}_{\text{GS}}^t) \rangle + \frac{L\eta^2\sigma^2}{2} \\ &= f(\mathbf{P}_{\text{GS}}^t) - \eta \langle \nabla f(\mathbf{P}_{\text{real}}^t), \nabla f(\mathbf{P}_{\text{real}}^t) \rangle + \frac{L\eta^2\sigma^2}{2} + \eta \langle \nabla f(\mathbf{P}_{\text{real}}^t), \nabla f(\mathbf{P}_{\text{real}}^t) - \nabla f(\mathbf{P}_{\text{GS}}^t) \rangle \\ &\leq f(\mathbf{P}_{\text{GS}}^t) - \eta \|\nabla f(\mathbf{P}_{\text{real}}^t)\|_2^2 + \frac{L\eta^2\sigma^2}{2} + \frac{\eta}{2} \|\nabla f(\mathbf{P}_{\text{real}}^t)\|_2^2 + \frac{\eta}{2} \|\nabla f(\mathbf{P}_{\text{real}}^t) - \nabla f(\mathbf{P}_{\text{GS}}^t)\|_2^2 \\ &\leq f(\mathbf{P}_{\text{GS}}^t) - \frac{\eta}{2} \|\nabla f(\mathbf{P}_{\text{real}}^t)\|_2^2 + \frac{L\eta^2\sigma^2}{2} + \frac{\eta L^2}{2} \|\mathbf{P}_{\text{real}}^t - \mathbf{P}_{\text{GS}}^t\|_2^2. \end{aligned}$$

With Lemma 1,

$$\mathbb{E} [f(\mathbf{P}_{\text{GS}}^{t+1} | \mathbf{P}_{\text{GS}}^t)] \leq f(\mathbf{P}_{\text{GS}}^t) - \frac{\eta}{2} \|\nabla f(\mathbf{P}_{\text{real}}^t)\|_2^2 + \frac{L\eta^2\sigma^2}{2} + \frac{\eta L^2}{2} (C_1 s_\tau^2 + C_2 r^{2k}),$$

where $C_1 > 0$ and $C_2 > 0$ are constants.

By rearranging the orders and further applying the expectation on \mathbf{P}_{GS}^t ,

$$\mathbb{E}\|\nabla f(\mathbf{P}_{\text{real}}^t)\|_2^2 \leq \frac{2}{\eta} (\mathbb{E} [f(\mathbf{P}_{\text{GS}}^t)] - \mathbb{E} [f(\mathbf{P}_{\text{GS}}^{t+1})]) + L\eta\sigma^2 + L^2 (C_1 s_\tau^2 + C_2 r^{2k}).$$

Summing over $t = 0, 1, \dots, T$,

$$\begin{aligned} \sum_{t=0}^T \mathbb{E}\|\nabla f(\mathbf{P}_{\text{real}}^t)\|_2^2 &\leq \frac{2}{\eta} \sum_{t=0}^T (\mathbb{E} [f(\mathbf{P}_{\text{GS}}^t)] - \mathbb{E} [f(\mathbf{P}_{\text{GS}}^{t+1})]) + L\eta\sigma^2 + L^2 \sum_{t=0}^T (C_1 s_\tau^2 + C_2 r^{2k}) \\ &\leq \frac{2}{\eta} (f(\mathbf{P}_{\text{GS}}^0) - f(\mathbf{P}_{\text{GS}}^*)) + (T+1)L\eta\sigma^2 + L^2 \sum_{t=0}^T (C_1 s_\tau^2 + C_2 r^{2k}). \end{aligned}$$

For a uniformly chosen \mathbf{u} from the iterates $\{\mathbf{P}_{\text{real}}^0, \dots, \mathbf{P}_{\text{real}}^T\}$, concretely $\mathbf{u} = \mathbf{P}_{\text{real}}^t$ with the probability $p_t = \frac{1}{T+1}$. Divide the inequation by $T+1$, and extend \mathbb{E} to represent the expectation over the stochasticity and the selection of \mathbf{u} , there is

$$\mathbb{E}\|\nabla f(\mathbf{u})\|_2^2 \leq \frac{2}{\eta(T+1)} (f(\mathbf{P}_{\text{GS}}^0) - f(\mathbf{P}_{\text{GS}}^*)) + L\eta\sigma^2 + L^2 (C_1 s_\tau^2 + C_2 r^{2k}).$$

Substituting the learning rate η , we finally obtain

$$\begin{aligned} \mathbb{E}\|\nabla f(\mathbf{u})\|_2^2 &\leq \frac{2\sigma\sqrt{LT}}{T+1} \sqrt{f(\mathbf{P}_{\text{GS}}^0) - f(\mathbf{P}_{\text{GS}}^*)} + \frac{\sigma\sqrt{L}}{\sqrt{T}} \sqrt{f(\mathbf{P}_{\text{GS}}^0) - f(\mathbf{P}_{\text{GS}}^*)} + L^2 (C_1 s_\tau^2 + C_2 r^{2k}) \\ &\leq 3\sigma \sqrt{\frac{f(\mathbf{P}_{\text{GS}}^0) - f(\mathbf{P}_{\text{GS}}^*)}{T/L}} + L^2 (C_1 s_\tau^2 + C_2 r^{2k}), \end{aligned}$$

Therefore,

$$\mathbb{E}\|\nabla f(\mathbf{u})\|_2^2 = \mathcal{O} \left(\sigma \sqrt{\frac{f(\mathbf{P}_{\text{GS}}^0) - f(\mathbf{P}_{\text{GS}}^*)}{T/L}} + L^2 (s_\tau^2 + r^{2k}) \right),$$

which concludes the proof. \square

B. Experiment Details and Analysis

B.1. Implementation Details

Implementation of ImageNet training. We follow the two-step scheme (as detailed in § 4 of the main paper) and the training settings in [12]. Specifically, for each step, the model is trained for 640k training iterations with batch size 512. We adopt the Adam optimizer [6] and set the initial learning rate to 10^{-3} . Weight decay rates in the first and second steps are 10^{-5} and 0, respectively. For experiments on ImageNet, models are trained with 8 V100 GPUs. We follow the training settings and data augmentation strategies in [12].

Implementation of CIFAR10 training. For experiments on CIFAR10, each experiment is performed on a single V100 GPU. We train the network with 256 epochs in each step. We set the batch size to 256 and use an Adam optimizer [6]. The learning rate is initialized to 5×10^{-4} and is updated by a linear learning rate decay scheduler. Results of our method on CIFAR10 are averaged over three runs.

Implementation of our selection and ablation studies. We further clarify our implementation of the codeword selection process based on Figure 4 (Middle) (main paper), where we provide four experiment settings including using kernel-wise/channel-wise codewords, and selection-based/product-quantization-based learning. Implementation details for these four experiments (labeled as (a)-(d), respectively) are described as follows.

- (a) *Selection, kernel-wise (our proposed method):* each codeword is a 3×3 convolutional kernel with values being ± 1 . We constantly keep the 1st (all -1 s) and 512th (all $+1$ s) codewords in the sub-codebook, as these two codewords take a large proportion. We divide the remaining 510 codewords into two halves (1st half: from index 2 to 256; 2nd half: from index 257 to 511). Obviously, each codeword in one half has a corresponding codeword with opposite signs in another half. This technique speeds up the selection process without noticeably affecting the performance.
- (b) *Selection, channel-wise:* each codeword is a 1×9 sub-vector with values being ± 1 , and codewords are obtained from flattened convolutional weights. We follow the speed up strategy in (a) by dividing codewords into two parts. Unlike (a), we do not constantly send the 1st (all -1 s) and 512th (all $+1$ s) codewords to the sub-codebook, since this process does not bring improvements for the channel-wise setting.
- (c) *Product quantization, kernel-wise:* we follow the common method of product quantization [8, 18] to learn real-valued codewords. Before training, we randomly initialize 2^n different 3×3 real-valued codewords, with their values being ± 1.0 . During training, in the forward pass, we obtain the binary codewords by applying the sign function on the real-valued codewords. In the backward pass, the Straight-Through Estimator technique is adopted which copies the gradients of binary codebooks to real-valued codebooks.
- (d) *Product quantization, channel-wise:* the learning process follows (c), yet each codeword is a 1×9 sub-vector across multiple channels, obtained from flattened convolutional weights.

Storage and BOPs calculation. In Table 4, we provide details of how storages and BOPs in Table 2 (main paper) are calculated. The calculation follows the analysis described in § 3.4 (main paper).

B.2. Additional Experiment Analysis

Power-law property. Figure 7 illustrates the distributions when ranking codewords in Figure 1 (main paper) according to the codeword frequency. It shows that in Figure 7(b), codewords nearly obey the power-law distribution.

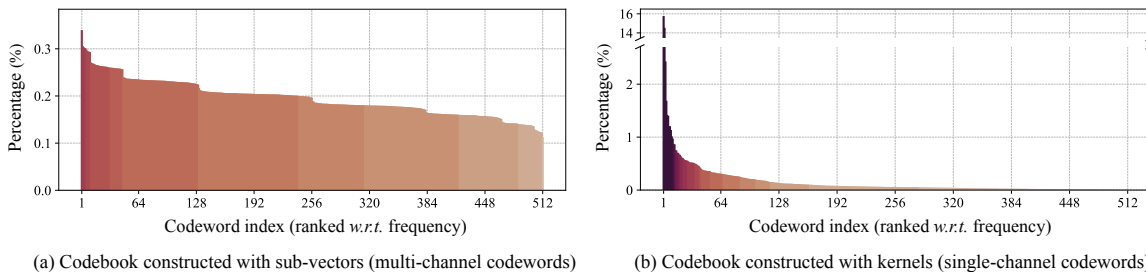


Figure 7. A supplement to Figure 1 (main paper) by ranking codewords *w.r.t.* the frequency. It shows that the ranked codewords in (b) nearly follow the power-law distribution.

Table 4. Calculation details for the storage and BOPs as reported in Table 2 (main paper). Networks are evaluated on the ImageNet dataset with ResNet-18.

layer-name	input-w	input-h	input-c	output-w	output-h	output-c	kernel-w	kernel-h	Storage (bit)				BOPs			
									1-bit (Base)	0.78-bit	0.67-bit	0.56-bit	1-bit (Base)	0.78-bit	0.67-bit	0.56-bit
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
conv1	224	224	3	112	112	64	7	7	-	-	-	-	-	-	-	-
conv2-1a	56	56	64	56	56	64	3	3	36864	28672	24576	20480	115605504	115605504	115605504	64225248
conv2-1b	56	56	64	56	56	64	3	3	36864	28672	24576	20480	115605504	115605504	115605504	64225248
conv2-2a	56	56	64	56	56	64	3	3	36864	28672	24576	20480	115605504	115605504	115605504	64225248
conv2-2b	56	56	64	56	56	64	3	3	36864	28672	24576	20480	115605504	115605504	115605504	64225248
conv3-1a	56	56	64	28	28	128	3	3	73728	57344	49152	40960	57802752	57802752	32112576	17661888
conv3-1b	28	28	128	28	28	128	3	3	147456	114688	98304	81920	115605504	115605504	64225216	35323840
conv3-2a	28	28	128	28	28	128	3	3	147456	114688	98304	81920	115605504	115605504	64225216	35323840
conv3-2b	28	28	128	28	28	128	3	3	147456	114688	98304	81920	115605504	115605504	64225216	35323840
conv4-1a	28	28	128	14	14	256	3	3	294912	229376	196608	163840	57802752	32112512	17661824	10436480
conv4-1b	14	14	256	14	14	256	3	3	589824	458752	393216	327680	115605504	64225152	35323776	20873088
conv4-2a	14	14	256	14	14	256	3	3	589824	458752	393216	327680	115605504	64225152	35323776	20873088
conv4-2b	14	14	256	14	14	256	3	3	589824	458752	393216	327680	115605504	64225152	35323776	20873088
conv5-1a	14	14	256	7	7	512	3	3	1179648	917504	786432	655360	57802752	17661696	10436352	6823680
conv5-1b	7	7	512	7	7	512	3	3	2359296	1835008	1572864	1310720	115605504	35323648	20872960	13647616
conv5-2a	7	7	512	7	7	512	3	3	2359296	1835008	1572864	1310720	115605504	35323648	20872960	13647616
conv5-2b	7	7	512	7	7	512	3	3	2359296	1835008	1572864	1310720	115605504	35323648	20872960	13647616
fc1000	1	1	512	1	1	1000	-	-	-	-	-	-	-	-	-	-
Total									10985472	8544256	7323648	6103040	1676279808	1215461888	883898624	501356672
									=11.0Mbit	=8.57Mbit	=7.32Mbit	=6.10Mbit	=1.68×10 ⁹	=1.22×10 ⁹	=0.88×10 ⁹	=0.50×10 ⁹


Storage: $J=G \times D \times H \times I$; $K=J \times \log_2(128)/9$; $L=J \times \log_2(64)/9$; $M=J \times \log_2(32)/9$.

BOPs: $N=D \times E \times F \times H \times I \times G$; $O= \min\{N, N/G \times 128 + G \times (D \times E \times F - 1)/2\}$; $P= \min\{N, N/G \times 64 + G \times (D \times E \times F - 1)/2\}$; $Q= \min\{N, N/G \times 32 + G \times (D \times E \times F - 1)/2\}$.

Codewords selection and overlaps. In Figure 8, we further compare the codewords learning processes for the four settings in Figure 4 (Middle) (main paper). As a supplement to Figure 4 (Right) (main paper), Figure 9 provides the change of sub-codebooks during training of the four experiment settings (a)-(d). As codewords in (c) and (d) tend to overlap during training, the diversity is severely affected. In addition, we also conduct experiments that at each step or every several steps, we replenish the sub-codebook with random different codewords so that the sub-codebook size recovers to 2^n , but the performance is very close to directly selecting codewords at random (as already shown in Figure 4 (Left) (main paper), randomly selection achieves low performance).

Acceleration of training. We consider two approaches that can accelerate the training process on large datasets (e.g. ImageNet), without causing much detriment to the performance. (1) We conduct permutation learning only in the first 30×10^3 training steps, and fix the selected codewords for the later training. (2) We obtain the sub-codebook by pretraining on a small dataset like CIFAR10, and directly adopt the sub-codebook for ImageNet without further permutation learning. Compared with the results of our Sparks reported in Table 2 (main paper), the performance does not decrease when using the acceleration approach (1), and decreases slightly (-0.4% , -0.6% , and -1.1% for 0.78-bit, 0.67-bit, and 0.56-bit, respectively) when using the approach (2).

Sensitivity analysis of hyper-parameters. In Figure 10, we compare the accuracy with different hyper-parameter settings for k and τ in Equation 6 (main paper). In the PSTE optimization, k is the iteration number and τ is the temperature. Experiments are performed with ResNet-18 and VGG-small on CIFAR10. We observe that both hyper-parameters are insensitive around their default settings $k = 10$ and $\tau = 10^{-2}$. The parameter k is quite easy to choose since results are stable when $k = 5 \sim 20$. In addition, regarding two extreme cases, setting τ too small (e.g. 10^{-4}) will hinder the smoothness of gradient back-propagation, and assigning a too large value (e.g. 1) will enlarge the permutation approximation error, both of which may cause detriment to the final performance. Luckily, according to Figure 10, the performance is stable when changing τ by 10 or 1/10 times around the default value, implying the high stability of our method.

About top-n most frequent codewords: In Figure 4 (main paper), we compare sampling top-n most frequent codewords and our method. We display an example of 0.44-bit top-n codewords here: . It shows the top-n codewords tend to choose adjacent codewords more frequently, which could hinder the diversity of the codebook. By contrast, as shown in Figure 5 (main paper), our learned 0.44-bit method outputs diverse codewords, yielding better performance particularly at 0.56-bit from 61.7 to 64.3.

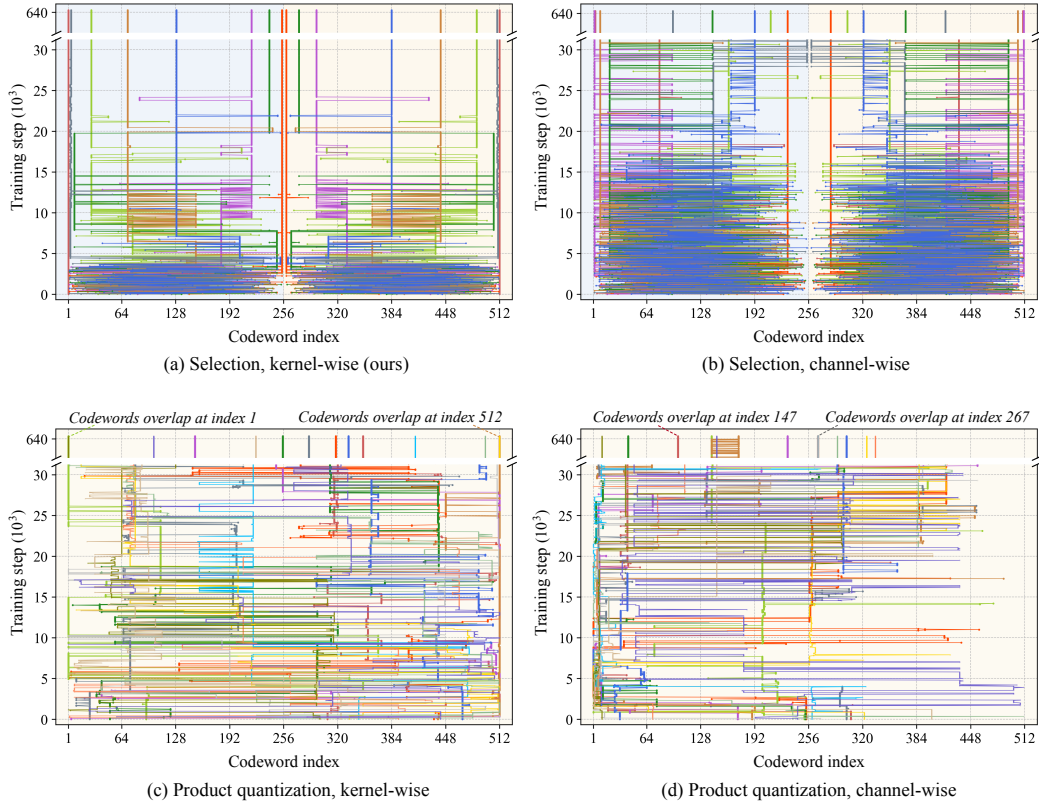


Figure 8. Comparison of codewords learning processes when using kernel-wise/channel-wise codewords and selection-based/product-quantization-based learning. The corresponding experimental results are already provided in Figure 4 (Middle) (main paper). All experiments are based on 0.44-bit settings with $n = 16$, and are performed on ImageNet upon ResNet-18. We observe that, when both using selection-based learning, kernel-wise codewords in (a) converge much faster (within 25×10^3 steps) than channel-wise codewords in (b); codewords with product-quantization-based learning in (c) and (d) also converge slower than (a), and are likely to overlap during training which degenerates the codebook diversity.

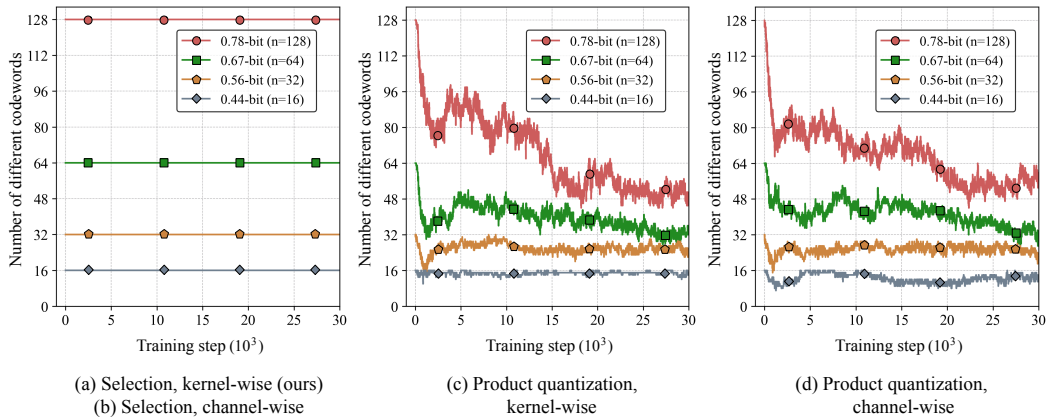


Figure 9. Numbers of different codewords when using selection-based/product-quantization-based learning, as a supplement to Figure 4 (Right) (main paper). (a)-(d) correspond to the experiments in Figure 4 (Right) (main paper) and Figure 8. Experiments are performed on ImageNet upon ResNet-18. We provide four settings, including 0.78-bit, 0.67-bit, 0.56-bit and 0.44-bit, with $n = 128, 64, 32$ and 16 , respectively. We observe that the sub-codebook highly degenerates during training in (c) and (d), since codewords tend to be repetitive when being updated independently. While in (a) and (b), the diversity of codewords preserves, which implies the superiority of our selection-based learning.

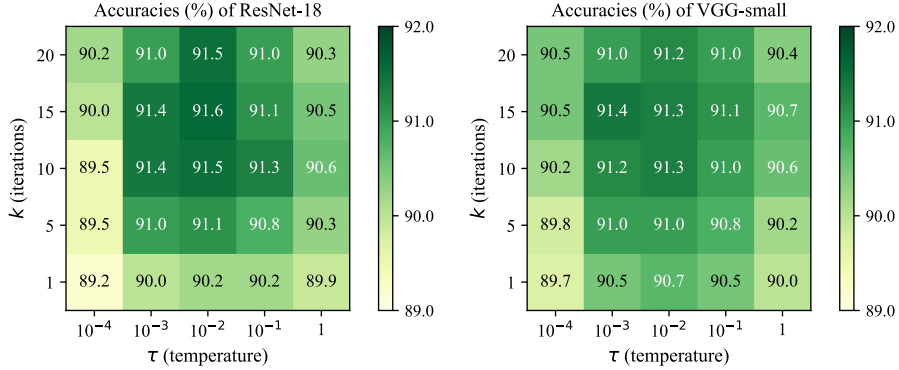


Figure 10. Sensitivity analysis for hyper-parameters including the iteration number k and the temperature τ , as adopted in Equation 6 (main paper) and also clearly illustrated in Figure 6. Experiments are conducted on the CIFAR10 dataset with 0.56-bit ResNet-18 and VGG-small, respectively. Results are averaged over three runs with different random seeds. Both hyper-parameters are insensitive around the default values $k = 10$ and $\tau = 10^{-2}$.

A two-step recipe with product quantization. Given the purpose of attaining a compact BNN, we also conduct an intuitive two-step baseline method: at first, load parameters of a standard BNN (ReActNet-18, from the open-sourced model) as the pre-trained model; then perform product quantization on the binary weights to compress the BNN. By this method, we achieve 59.1% accuracy on ImageNet under 0.56-bit, and the sub-codebook degeneration still exists. Such result is much inferior to our Sparks under 0.56-bit (64.3%).

Symbol table. We list the definition and usage of important symbols in Table 5.

Whether indexing process hinders valid acceleration: No, the indexing process of binary kernels does not hinder valid acceleration. (1) Indexing n codewords is very cheap, *e.g.*, only 32 codewords for our 0.56-bit model (< 0.5 nanosecond on our FPGA). (2) Indexing 3×3 pre-calculated results (3×3 codeword $\otimes 3 \times 3$ feature region, see § 3.4 of the main paper) is also negligible based on our implementation with a Lookup Table (LUT) that stores 3×3 pre-calculated results: The practical LUT size, instead of $n \times C_{in} \times H \times W$ by dividing input feature maps into $1 \times 3 \times 3$ slices, and sending only one slice to a Processing Engine (PE) at each clock cycle. This leads to very low latency for the lookup process, *e.g.*, LUT size is 32 for our 0.56-bit model (< 0.5 nanosecond for indexing), which is easily implemented in the current clock cycle.

C. Object Detection

C.1. Implementation

We evaluate our method for object detection on two benchmark datasets: PASCAL VOC [2] and COCO [9]. We follow the standard data split settings [19]. Regarding the PASCAL VOC dataset, we train our model on both the VOC 2007 and VOC 2012 trainval sets, which together contain about 16k natural images of 20 different categories in total. We evaluate the performance on VOC 2007 test set that is composed of about 5k images. COCO dataset (2014 object detection track) is a

Table 5. The definition and usage of important symbols.

Symbol	Definition and Usage
$\mathbf{w} \in \mathbb{R}^{K \times K}$	Convolutional kernel with the kernel size K .
$\hat{\mathbf{w}} \in \{-1, +1\}^{K \times K}$	Selected binary kernel for \mathbf{w} . There is $\hat{\mathbf{w}} \in$ sub-codebook $\mathbb{U} \subseteq$ codebook $\mathbb{B} = \{-1, +1\}^{K \times K}$.
$N \in \mathbb{Z}^+$	$N = \mathbb{B} $, the codebook size, $N = 512$ when $K = 3$.
$n \in \mathbb{Z}^+$	$n = \mathbb{U} $, the sub-codebook size, <i>e.g.</i> , for 0.56-bit Sparks, $n = 32$.
$\mathbf{B} \in \{\pm 1\}^{K^2 \times N}$	Column by column indexing from $ \mathbb{B} $.
$\mathbf{U} \in \{\pm 1\}^{K^2 \times n}$	Column by column indexing from $ \mathbb{U} $.
$\mathbf{V} \in \{0, 1\}^{N \times n}$	A pre-defined selection matrix.
$k \in \mathbb{Z}^+$	The number of iteration to approximate the permutation matrix in Equation 8 (main paper).
$\tau \in \mathbb{R}^+$	A small temperature to approximate the permutation matrix in Equation 8 (main paper).
$\mathbf{X} \in \mathbb{R}^{N \times N}$	A randomly initialized, learnable matrix.
$\mathbf{P}_{GS} \in \mathbb{R}^{N \times N}$	$\mathbf{P}_{GS} = \mathcal{S}^k((\mathbf{X} + \epsilon)/\tau)$, the approximated permutation matrix for propagation, not a 0/1 matrix.
$\mathbf{P}_{real} \in \{0, 1\}^{N \times N}$	$\mathbf{P}_{real} = \text{Hungarian}(\mathbf{P}_{GS})$, the outputted permutation matrix, a doubly stochastic 0/1 matrix.

Table 6. Performance comparisons with object detection methods on the PASCAL VOC dataset. Sparks* indicates using the two-step training method and the generalized Sign/PReLU functions (as adopted in [12] for image classification).

Method (SSD300)	Bit-width (W/A)	mAP (%)	Storage Saving	BOPs Saving	Method (Faster R-CNN)	Bit-width (W/A)	mAP (%)	Storage Saving	BOPs Saving
Full-precision	32/32	72.4	1×	1×	Full-precision	32/32	74.5	1×	1×
BNN [4]	1/1	42.0	32×	64×	BNN [4]	1/1	35.6	32×	64×
XNOR-Net [14]	1/1	50.2	32×	64×	XNOR-Net [14]	1/1	48.4	32×	64×
Bi-RealNet [11]	1/1	63.8	32×	64×	Bi-RealNet [11]	1/1	58.2	32×	64×
BiDet [19]	1/1	66.0	32×	64×	BiDet [19]	1/1	59.5	32×	64×
Sparks (ours)	0.78/1	65.2	41.0×	108×	Sparks (ours)	0.78/1	58.9	41.0×	88×
Sparks (ours)	0.56/1	64.3	57.1×	285×	Sparks (ours)	0.56/1	57.7	57.1×	214×
Sparks* (ours)	0.78/1	68.9	41.0×	108×	Sparks* (ours)	0.78/1	66.2	41.0×	88×
Sparks* (ours)	0.56/1	68.0	57.1×	285×	Sparks* (ours)	0.56/1	65.5	57.1×	214×

Table 7. Performance comparisons with object detection methods on the COCO dataset. Sparks* indicates using the two-step training method and the generalized Sign/PReLU functions (as adopted in [12] for image classification).

Method (SSD300)	Bit-width (W/A)	mAP (%)	AP ₅₀ (%)	AP ₇₅ (%)	AP _s (%)	AP _m (%)	AP _l (%)	Method (Faster R-CNN)	Bit-width (W/A)	mAP (%)	AP ₅₀ (%)	AP ₇₅ (%)	AP _s (%)	AP _m (%)	AP _l (%)
Full-precision	32/32	23.2	41.2	23.4	8.6	23.2	39.6	Full-precision	32/32	26.0	44.8	27.2	10.0	28.9	39.7
BNN [4]	1/1	6.2	15.9	3.8	2.4	10.0	9.9	BNN [4]	1/1	5.6	14.3	2.6	2.0	8.5	9.3
XNOR-Net [14]	1/1	8.1	19.5	5.6	2.6	8.3	13.3	XNOR-Net [14]	1/1	10.4	21.6	8.8	2.7	11.8	15.9
Bi-RealNet [11]	1/1	11.2	26.0	8.3	3.1	12.0	18.3	Bi-RealNet [11]	1/1	14.4	29.0	13.4	3.7	15.4	24.1
BiDet [19]	1/1	13.2	28.3	10.5	5.1	14.3	20.5	BiDet [19]	1/1	15.7	31.0	14.4	4.9	16.7	25.4
Sparks (ours)	0.78/1	13.4	28.6	10.6	5.3	14.5	20.8	Sparks (ours)	0.78/1	15.6	30.7	14.0	4.7	16.5	25.1
Sparks (ours)	0.56/1	12.5	27.7	10.0	4.9	14.1	19.6	Sparks (ours)	0.56/1	14.9	29.9	13.6	4.1	15.7	24.5
Sparks* (ours)	0.78/1	18.6	35.7	17.4	7.1	19.3	31.0	Sparks* (ours)	0.78/1	21.2	37.5	18.2	7.8	22.6	31.7
Sparks* (ours)	0.56/1	17.6	33.9	17.0	6.6	18.1	29.4	Sparks* (ours)	0.56/1	20.0	36.8	17.4	7.0	20.2	30.5

large-scale dataset that collects images from 80 different categories. We train our model with 80k training images as well as 35k images sampled from the validation set (denoted as trainval35k [1]), and carry out evaluations on the remaining 5k images in the validation set (minival [1]).

We follow BiDet [19] for the basic training settings including parameters and data augmentation methods. Specifically, we train 50 epochs in total with batch size 32 and the Adam optimizer. We initialize the learning rate to 10^{-3} which decays by multiplying 0.1 at the 6th and 10th epoch. We consider two typical architectures including SSD300 [10] (with VGG-16 [16]) and Faster R-CNN [15] (with ResNet-18 [3]) to verify the effectiveness and generalization of our method.

C.2. Results

Evaluation on PASCAL VOC. We contrast Sparks against SOTA detection binarization methods including the standard BNN [4], Bi-RealNet [11], XNOR-Net [14] and BiDet [19] in Table 6. We implement two different versions of Sparks including 0.56-bit and 0.78-bit. Compared with BiDet, our 0.56-bit method obtains about model compression by twice (0.56 vs 1) and computation acceleration by more than 3 times (e.g. 285 vs 64 on VGG16+SSD300). Besides, by adopting the two-step training scheme and the generalized Sign/PReLU functions, our methods achieve new records on 1-bit object detection.

Evaluation on COCO. To further assess the proposed method on a larger and more challenging dataset, we conduct experiments on COCO. Comparisons with SOTA methods are provided in Table 7. Following the standard COCO evaluation metrics, we report the average mAP over different IoU thresholds from 0.5 to 0.95, the APs at particular thresholds: AP₅₀

and AP_{75} , and the scale-aware metrics: AP_s , AP_m and AP_l . The benefits of Sparks are still observed, namely, clearly saving in complexity. Results with SSD300 indicate that our 0.78-bit Sparks even defeats BiDet in terms of all evaluation metrics. We speculate that Sparks reduces information redundancy by selecting essential codewords, and thus eliminates some of the false positives. In addition, our method performs stably for both the one-stage SSD300 and the two-stage Faster R-CNN, implying its robustness on different backbones. In addition, results of Sparks* indicate that our method also has compatibility with the two-step training scheme and the generalized functions.

D. Discussion and Limitation

In this research, we propose Sparks that largely enhances both the storage and computation efficiencies of BNNs. Our work is motivated by that kernel-wise codewords are highly clustered. For this reason, we propose a novel selection-based approach for kernel-wise sub-codebook learning instead of previously used channel-wise product quantization. By extending our Sparks with more layers or other blocks, the performance could surpass the standard BNN model with still fewer parameters and BOPs. This provides us with a new research line of training lighter and better BNN models. As an open-sourced research on well-used benchmarks, our method does not raise ethical concerns. However, one should notice that compressing a model needs to access the model parameters which might need further protection methods for the model privacy.

References

- [1] Sean Bell, C. Lawrence Zitnick, Kavita Bala, and Ross B. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *CVPR*, 2016. 8
- [2] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. In *IJCV*, 2010. 7
- [3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 8
- [4] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *NeurIPS*, 2016. 8
- [5] Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian U. Stich, and Martin Jaggi. Error feedback fixes signsgd and other gradient compression schemes. In *ICML*, 2019. 2
- [6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 4
- [7] Philip A. Knight. The sinkhorn-knopp algorithm: Convergence and applications. In *SIAM J. Matrix Anal. Appl.*, 2008. 2
- [8] Weichao Lan and Liang Lan. Compressing deep convolutional neural networks by stacking low-dimensional binary convolution filters. In *arXiv preprint arXiv:2010.02778*, 2020. 4
- [9] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *ECCV*, 2014. 7
- [10] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. In *ECCV*, 2016. 8
- [11] Zechun Liu, Wenhan Luo, Baoyuan Wu, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-real net: Binarizing deep network towards real-network performance. In *IJCV*, 2020. 8
- [12] Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. Reactnet: Towards precise binary neural network with generalized activation functions. In *ECCV*, 2020. 4, 8
- [13] Gonzalo E. Mena, David Belanger, Scott W. Linderman, and Jasper Snoek. Learning latent permutations with gumbel-sinkhorn networks. In *ICLR*, 2018. 2
- [14] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, 2016. 8
- [15] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 8
- [16] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 8
- [17] Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. In *The Annals of Mathematical Statistics*, 1964. 2
- [18] Pierre Stock, Armand Joulin, Rémi Gribonval, Benjamin Graham, and Hervé Jégou. And the bit goes down: Revisiting the quantization of neural networks. In *ICLR*, 2020. 4
- [19] Ziwei Wang, Ziyi Wu, Jiwen Lu, and Jie Zhou. Bidet: An efficient binarized object detector. In *CVPR*, 2020. 7, 8