MABA-NET: MASKED ADDITIVE BINARY ACTIVATION NETWORK

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite significant reduction in memory footprint and computational cost, binary neural networks suffer from noticeable accuracy degradation compared to real-valued counterparts. A few works have attempted to narrow the accuracy gap by increasing the representation bit-width or the network width/depth, but they come at the expense of increased memory and/or compute. In this work, we find that the imbalanced ratio of activations to weights may be the main cause of degraded performance and increased memory overhead. We propose Masked Additive Binary Activation Network (MABA-Net) to reduce approximation errors and the activation bit-width, with minimum increase in the activation size. MABA-Net balances the ratio of the activation size to the weight size, leading to significant memory saving on large CNNs. We demonstrate MABA-Net's superior performance on the ImageNet dataset under various network configurations. Experimental results show that MABA-Net achieves competitive accuracy without increase of computational cost, while reducing memory usage compared to state-of-the-art. We will release the codes upon acceptance.

1 INTRODUCTION

Convolutional neural networks (CNNs) have achieved great success in a wide range of applications but it comes at the expense of large memory and compute consumption. Subsequently, it is particularly challenging to deploy such networks on resource-limited devices. Neural network compression (Han et al. (2016b)) has emerged as a mainstream technique to reduce memory and compute, by either quantizing real-valued weights and activations to low precision numbers (Zhou et al. (2016)) or pruning redundant weights from the network (Li et al. (2016)). An extreme form of quantization is Binary Neural Networks (BNNs) (Courbariaux et al. (2015; 2016)), which replace slow real-valued matrix multiplication with high-speed bit-wise XNOR and POPCOUNT operations. BNNs have demonstrated their potentials in fast and efficient inference.

Despite the promising advantages over real-valued neural network in terms of memory and compute, vanilla BNNs (Courbariaux et al. (2015; 2016)) suffer from significant accuracy degradation, mainly due to the approximation error caused by real-to-binary projections. A common way to mitigate the accuracy drop is designing compact and efficient networks (Bulat et al. (2021); Liu et al. (2020); Bethge et al. (2021)). However, it is difficult to systematically measure the benefits of each component, such as scaling factor and skip connections. Another way is increasing the binary network width (Lin et al. (2017); Zhuang et al. (2019); Zhu et al. (2019)) but it comes at the expense of increased memory and compute.

Matrix-vector multiplication is a basic building block in machine learning models, especially in CNNs. During the matrix multiplication, the memory access is usually the bottleneck, especially when the matrix is larger than the cache capacity (Han et al. (2016a)). On generic CPUs and GPUs, this problem is usually approached by reusing the parameters through batching. However, batching is unsuitable for real-time applications that are latency-sensitive, e.g., pedestrian detection in autonomous driving, because batching substantially increases latency. In this work, we target real time applications. Therefore, we focus on the case of batch size = 1.

An important observation that we made is that, the large memory overhead for such extreme latency-sensitive cases is probably due to the imbalanced ratio of activation size (number of activations) to weight size (number of weights). For example, the number of convolutional activations is $1/6$

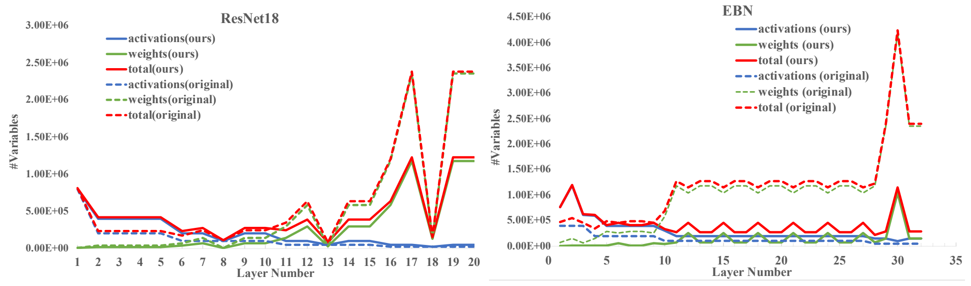


Figure 1: Distribution of activation and weight across layers for ResNet18 (He et al. (2016)) and EBN (Bulat et al. (2021)).

to $1/7$ of the weights for ResNet-like architectures on ImageNet Bulat et al. (2021) (see Fig. 1). In particular, the activation size at most of the layers is an order of magnitude smaller than that of the weight size. We hypothesise that the relatively small size of activations might cause more severe information bottleneck issue after activation binarization, compared with weight binarization. Naturally, by balancing the activations and weights, we can make the network less prone to accuracy loss brought by activation binarization. The experimental results shown in Table 2 suggest that by doing so, our method alleviates the information loss issue caused by activation binarization.

In this paper, we propose a multi-branch bit-aware binary activation network with spatial masking to balance the distribution of binarized weights and activations. Our idea motivates from the fixed-point binary representation. For example, the floating point number 0.7865 can be expressed as $0.7865 = 2^{-1} + 2^{-3} + 2^{-4} = 0.1011_2$. We select a subset of the bits of the binary fixed-point representation to approximate the real-valued activations using matrix multiplication approximation algorithms. In the end, the activations can be approximated by an additive multi-branch network where each branch represents a bit in the fixed-point representation.

Different from ABCNet (Lin et al. (2017)), where the bases and the binary representations are learnt without differentiating the role of different branches, our method assigns different power-of-two values to distinguish the branches, and thus there is no redundant information among different branches. To further reduce the activation memory, we propose to add a spatial masking to the channels of the activations. Finally, experiments on the large-scale dataset demonstrate that our proposed methods maintain the accuracy without an increase in both computational budget and memory usage.

Our contributions are three-fold:

- We propose the Masked Additive Binary Activation Network (MABA-Net) to moderately increase the size of binarized activations, by approximating real-valued activations as the sum of masked binary activations weighted by power-of-2 factors. Compared to previous approaches, our method offers significantly reduced approximation error and fewer bit-widths for binarized activations simultaneously, resulting in smaller accuracy gap to full-precision counterparts.
- By integrating our proposed methods with various network architectures, the distributions of activations to weights becomes balanced, leading to significant savings of total memory.
- Extensive evaluations on ImageNet demonstrate that MABA-Net obtains comparable or better accuracy than state-of-the-art without an increase in computational cost, while with up to $3\times$ reduction in memory footprint.

2 Related Work

2.1 Binary Neural Networks

Binary Neural Networks, as the extreme case of neural network quantization, suffers from significant degradation. Many researchers have explored how to address the gap.

Network architecture One direction is to explore the network architecture to improve the BNN performance. For example, ReActNet (Liu et al. (2020)), Bi-Real-Net (Liu et al. (2018)), Real-to-Bin (Martinez et al. (2020)) and MeliusNet (Bethge et al. (2021)) through modified classical network (e.g., MobileNet, ResNet) to improve the accuracy. EBN (Bulat et al. (2021)), Phan et al. (2020) and Kim et al. (2020a) introduce to search the optimal architecture for BNN to improve the performance. BENN(Zhu et al. (2019)), GroupNet (Zhuang et al. (2019)) and ABCNet (Lin et al. (2017)) propose to expand the block structure to improve the performance. Our methods propose to widen the activation but not the weights through designed additive bit-wise branches to improve the performance. To be fair, we conduct extensive comparison with different network settings.

Training strategy and tricks Another direction is investigating how different training scheme and tricks affect the performance of BNNs. For instance, ReActNet, Real-to-Bin, EBN and BCNN (Redfern et al. (2021)) adopt a two-stage training strategy. XNORNet (Rastegari et al. (2016)), XNORNet++ (Bulat & Tzimiropoulos (2019)) and EBN design different scaling factors to avoid inconsistency between the binarized representations and the full-precision data. Liu et al. (2021), Ajanthan et al. (2021) investigate different optimizers to improve the performance. For the activation function, Bi-Real-Net replace ReLU with Htanh while (Tang et al. (2017)) instead use PReLU for BNNs. Our method investigates the training strategy in the experiments.

Gradient approximation STE roughly sets the gradients of the binarized value to the full-precision data as 1, which brings information loss in the weight update of the back-propagation, and thus many research efforts design various surrogate gradients to address the issue such as BinaryDuo (Kim et al. (2020b)), BiRealNet (Liu et al. (2018)), and IR-Net (Qin et al. (2020)). For example, Liu et al. (2022) proposes a generalized straight-through estimator (G-STE) for intractable backward derivative calculation to output uniform quantized values. Instead, our method alleviates the STE issue by replacing the sign function as truncation.

Power-of-Two Quantization Many researchers have explored how to conduct Power-Of-Two quantization as it can convert the multiplication to hardware efficient bit-shift. For example, Gudovskiy & Rigazio (2017) quantizes the weights as a sum of power-of-two values. While APoT Li et al. (2020), based on the distributions of weights and activations, constrains all quantization levels as the sum of Powers-of-Two terms. Yao et al. (2022) dynamically adjusts the Power-of-Two scales of the whole network instead of statically determining them layer by layer in the post-training quantization. DeepShift Elhoushi et al. (2021) proposes to replace multiplications in convolutional layers and fully connected layers with bitwise shift and sign flipping by quantizing the weights during both training and inference. Instead, to binarize the network, our method represents the real value into power-of-two bases with binary representations.

2.2 Network pruning

Pruning Mask Mask has been a handy tool in learning the importance of sub-structure of neural network during pruning. In fact, it enables researchers to explore the effectiveness of pruning at different granularity (He et al. (2017),Liu et al. (2017),Lemaire et al. (2019),Chin et al. (2020),Kusupati et al. (2020)). Masks at filter, channel or stripe level enables the network to learn a structure that is hardware friendly, whereas masks at weight level may produce a sparser network, but such unstructured pruning network might require dedicated hardware for inference.

Pruning in BNN On the contrary, BNN has not received many attentions in the field of network pruning because BNN is already a compact model. Furthermore, it is not the top priority for BNN to further compress the model and reduce compute when there is a large performance gap remained as compared to its real-value counterpart. However, as BNN's performance keeps improving with the help of recently devised techniques, it makes sense to consider the question whether we can maintain BNN's performance with less memory and computation cost.

3 Preliminary

3.1 Binarized Approximation

Let $\mathbf{W}_r$ and $\mathbf{X}_r$ represent the real-valued weights and input activations for a layer. Binary network attempts to approximate $\mathbf{W}_r$ and $\mathbf{X}_r$ with the binarized weights $\mathbf{W}_b$ and activations $\mathbf{X}_b$ (Rastegari et al. (2016); Bulat & Tzimiropoulos (2019); Xu & Cheung (2019)) as below.

$$\mathbf{W}_r \approx \alpha \mathbf{W}_b = \alpha \cdot sign(\mathbf{W}_r), \tag{1}$$

$$\mathbf{X}_r \approx \beta \mathbf{X}_b = \beta \cdot sign(\mathbf{X}_r), \tag{2}$$

where $sign(\cdot)$ denotes the sign function, α and β are learnable scaling factors that match the binarized values to the same scale as the real values. Compared to the full-precision network, the binary network not only reduces the memory footprint by $32\times$, but also replaces the multiply-accumulate (MAC) convolutional operation between $\mathbf{W}_b$ and $\mathbf{X}_b$ with the XNOR-popcount operation as below:

$$\mathbf{W}_b \circledast \mathbf{X}_b = \boldsymbol{w}_b \cdot \boldsymbol{x}_b = \text{popcount}(\text{xnor}(\boldsymbol{w}_b, \boldsymbol{x}_b)) \tag{3}$$

where $\boldsymbol{w}_b$ and $\boldsymbol{x}_b$ are the flattened vector representations of $\mathbf{W}_b$ and $\mathbf{X}_b$, and $\cdot$ here denotes the dot product. The replacement reduces the number of operations by $43\times$ (Fromm et al. (2020)).

3.2 Back-propagation with binarization

In general, the quantization function has the gradients of zeros almost everywhere and thus it is impossible to directly apply the back-propagation to train the network. To overcome this issue, a classical way is using the empirical straight-through estimator (Bengio et al. (2013b)) to re-define the gradients. Namely, the derivative of the binarized value z_b to the real value z_r is an identity mapping, namely $\frac{\partial z_b}{\partial z_r} \approx 1$. Accordingly, the derivative of the loss $\mathcal{L}$ to z_r can be approximated by

$$\frac{\partial \mathcal{L}}{\partial z_r} = \frac{\partial \mathcal{L}}{\partial z_b}\frac{\partial z_b}{\partial z_r} \approx \frac{\partial \mathcal{L}}{\partial z_b}. \tag{4}$$

4 Method

4.1 Binarize Weights

Following Liu et al. (2018), we binarize the weights of each convolutional layer with a magnitude-based scaling factor as below:

$$\mathbf{W}_b = \frac{||\mathbf{W}_r||_1}{|\mathbf{W}_r|} sign(\mathbf{W}_r) \tag{5}$$

where $|\mathbf{W}_r|$ denotes the number of entries in $\mathbf{W}_r$, the updating of $\mathbf{W}_r$ is computed as follows:

$$\mathbf{W}_r^{t+1} = \mathbf{W}_r^t - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}_b^t}\frac{\partial \mathbf{W}_b^t}{\partial \mathbf{W}_r^t} \tag{6}$$

where $\frac{\partial \mathbf{W}_b^t}{\partial \mathbf{W}_r^t} = \frac{||\mathbf{W}_r^t||_1}{|\mathbf{W}_r^t|} \cdot \frac{\partial sign(\mathbf{W}_r^t)}{\partial \mathbf{W}_r^t}$. Because the derivative of $sign(\cdot)$ function is almost zero everywhere and thus preventing the flow of gradients, conventional methods approximate $\frac{\partial sign(\mathbf{W}_r^t)}{\partial \mathbf{W}_r^t}$ with the identity mapping, namely $\frac{\partial sign(\mathbf{W}_r^t)}{\partial \mathbf{W}_r^t} = 1$. As empirically not cancelling the gradients when weights is too large significantly worsen the performance, most methods include a clipping to cancel the gradients when weights is too large. Therefore, we define an indicator function $\mathbb{1}_{|\mathbf{W}_r^t|<1}$ to set the gradients of the weights that are outside of the range to 0. Consequently, the final form of $\frac{\partial \mathbf{W}_b^t}{\partial \mathbf{W}_r^t}$ can be approximated by $\frac{||\mathbf{W}_r^t||_1}{|\mathbf{W}_r^t|} \cdot \mathbb{1}_{|\mathbf{W}_r^t|<1}$.

4.2 Binarize Activations

In many hardware, the floating-point is represented by the IEEE-754 standard (Kahan (1996)) for arithmetic. Our idea motivates from the fixed-point representation (Gupta et al. (2015)), which has a fixed number of bits for the integral and fractional parts. For example, $0.7865 = 2^{-1} + 2^{-3} + 2^{-4}$, is 0.1011_2. Mathematically, the activation $\mathbf{X}_r$ can be represented as the matrix multiplication between a binary matrix $\mathbf{X}_B \in \{0,1\}^{N\times 31}$ and a basis vector $D = [2^{-1}, 2^{-2}, 2^{-3}, ..., 2^{-31}]^T$. We can approximate $\mathbf{X}_r$ by approximating the matrix multiplication of $\mathbf{X}_B D$ as below:

$$\mathbf{X}_r = \mathbf{X}_B D \approx \mathbf{X}_b Q \tag{7}$$

where $\mathbf{X}_b \in \{0,1\}^{N\times S}$ is the subset of $\mathbf{X}_B$, and the corresponding basis vector Q and S denotes the bit-width. We use the importance sampling approach (Mahoney et al. (2011); Drineas et al. (2006)) to compute the optimal $\mathbf{X}_b$ and Q by minimizing the expectation of MSE (mean squre error) as $E(||\mathbf{X_B}D - \mathbf{X_b}Q||_F^2)$ where $||\cdot||_F$ denotes the Frobenius norm. We revise the importance sampling algorithm by selecting the index i_t with top S probabilities as in Algorithm 1. The detailed derivation is shown in Appendix A.1. In the end, the real value $\mathbf{X}_r$ is represented as:

Algorithm 1: Activation Approximation by Importance Sampling

```
1 Input: The binary matrix $\mathbf{X}_B \in (0,1)^{N\times 31}$, the vector $D = [2^{-1}, 2^{-2}, ..., 2^{-31}]^T$, the bit-width $S$.
2 Output: The binary matrix $\mathbf{X}_b \in \{0,1\}^{N\times S}$, the column vector $Q$;
3 for $t = 1$ to $S$ do
4     Pick $i_t \in \{1, ..., 31\}$ with the top $t$ probability $p_{i_t}$, where $p_{i_t} = \frac{\|\mathbf{X}_B^{(i_t)}\| \cdot D_{(i_t)}}{\sum_l \|\mathbf{X}_B^{(l)}\| \cdot D_{(l)}}$, $\mathbf{X}_B^{(i_t)}$ denotes the $i_t$
        row of $\mathbf{X}_B$ and $D_{(i_t)}$ denotes the $i_t$ column of $D$.
5     Set the $t^{th}$ column of $\mathbf{X}_b$ as $\mathbf{X}_b^{(t)} = \mathbf{X}_B^{(i_t)}$ and the $t^{th}$ row of $Q$ as $Q_{(t)} = \frac{D_{(i_t)}}{Sp_{i_t}}$
6 return C and Q
```

$$\mathbf{X}_r \approx \frac{1}{S}\sum_{t=1}^{S} \frac{1}{p_{i_t}} 2^{-i_t} \mathbf{X}_b^{(i_t)} \tag{8}$$

We can make some simplification to the expression above to not only construct a more succinct expression, but also make the approximation more hardware friendly by removing unnecessary floating point operations. Firstly, we can remove $\frac{1}{S}$ from the above expression because it is a constant and can be infused into the weights. Secondly, we ask the question if we can remove $\frac{1}{p_{i_t}}$, which is a floating point number, from the expression. To answer this question, we resolve to empirical evidence; we conducted experiments that are with $\frac{1}{p_{i_t}}$ and without $\frac{1}{p_{i_t}}$. Our experiment results show that $\frac{1}{p_{i_t}}$ does not affect model's performance when S is kept small, which is the case in our implementation with $S = 2$ or 3. Therefore, we approximate the $\mathbf{X}_r$ as following:

$$\mathbf{X}_r \approx \sum_{t=1}^{S} 2^{-i_t} \mathbf{X}_b^{(i_t)} \tag{9}$$

As $\mathbf{X}_r$ is in fixed-point representation, we can obtain $\mathbf{X}_b$ by bit-shifting shown in Appendix A.1. Furthermore, we can obtain the gradients $\frac{\partial \mathbf{X}_b^{(i_t)}}{\partial \mathbf{X}_r} = 2^{i_t}$.

As the network usually has a ReLU as the activation function, we only consider binarizing the non-negative activations. Following Hubara et al. (2016), we set a bounded range for the real-valued activations $\mathbf{X}_r$ as $\mathbf{X}_r = \text{clip}(\mathbf{X}_r, [0, 1])$ to cancel the gradient when $\mathbf{X}_r$ is too large, which significantly worsens the performance.

4.3 Spatial Masking

The increase in bit-width for binarized activations may introduce information redundancy, meanwhile linearly increases the activation size and the number of binary operations (BOPS). To reduce the redundancy, save the memory and the computation cost, a trainable binary mask variable $\mathbf{M} \in R^{W\times H}$ is applied to each spatial position (w, h) of the binarized activations $X_b \in R^{C\times W\times H}$ where C, W and H denote the number of channels, the width and height of input activations. If $\mathbf{M}_{w,h} = 0$, the whole channel of the activation features at location (w, h) are totally removed; otherwise, the features are kept in memory and involved in the computation process. The higher sparsity the binary masks, the smaller the activation size is.

To reduce the activation size, we introduce a BOPS-aware sparsity loss on the binary mask variables $\mathbf{M}$ to explicitly enforce it moving to 0 during training.

$$\mathcal{L}_S = \frac{\sum_l \frac{||\mathbf{M}^l||_1}{W^l \times H^l} \times \text{BOPS}^l}{\sum_l \text{BOPS}^l} \tag{10}$$

where $\mathbf{M}^l$ denotes the binary mask variables of the binazied activations at the l^{th} layer. BOPS^l denotes the number of binary operations at the l^{th} layer. The entry value of $\mathbf{M}$ is either 0 or 1 and thus the L1-norm of $\mathbf{M}$ corresponding to the left activations. For simplicity, we apply STE (Bengio et al. (2013a)) on the training of these mask variables. To generate the binarized entry of $\mathbf{M}$, we assign $\mathbf{M}_{w,h} = 1$ if $\mathbf{M} > 0$ else 0. In the backward pass, the gradients are approximated by the identity mapping. Note that there is memory overhead introduced by the binary mask variables themselves, with $W * H * S$ parameters per layer ($S \leqslant 3$ in our experiments), which is negligible compared to size of weights and activations.

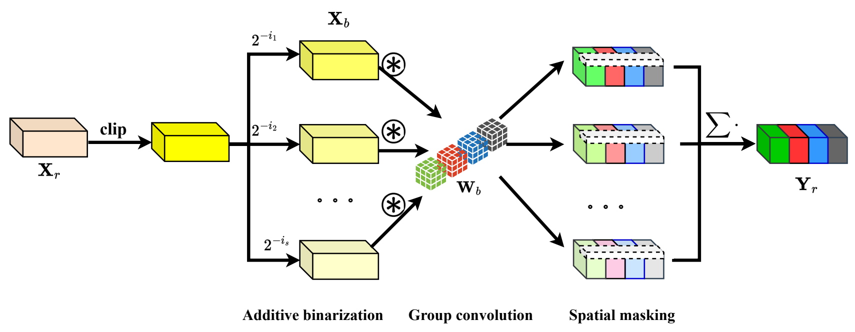


Figure 2: Our framework of the Masked Additive Binary Activations with grouped convolutions.

4.4 Objective Function

The overall framework is shown in Fig. 2. Structurally, it is a multi-branch network where each branch contributes to a specific impact, different from ABCNet without differentiating the role of different branches. To reduce the model parameters, we apply group convolutions to the 3×3 convolutional layer in the residual block.

The final objective function is as below:

$$\min_{\{\mathbf{W},\mathbf{M}\}} \mathcal{L}_E + \lambda \mathcal{L}_S \tag{11}$$

where $\mathcal{L}_E$ is the cross-entropy loss, and $\mathcal{L}_S$ is the sparsity regularization term to enforce the MABA-Net to meet the pre-defined computational budget. λ is to control the impact of the sparsity loss.

4.5 Computation Cost

In this paper, the binarization of activations is unipolar quantization where bits with value 0 represent 0 and bits with value 1 represent 1; while the binarization of weights is bipolar quantization where bits with value 0 represent -1 and bits with value 1 represent 1 (Xu & Cheung (2019); Fromm et al. (2020)). Hence, the convolution of $\mathbf{X}_b$ and $\mathbf{W}_b$ can be computed as follows.

$$\mathbf{W}_b \circledast \mathbf{X}_b = \text{popcount}(\text{AND}(\mathbf{X}_b, \mathbf{W}_b)) - \text{size}(\mathbf{W}_b) \tag{12}$$

where $\text{size}(\mathbf{W}_b)$ denotes the number of elements in $\mathbf{W}_b$ and thus is a constant number. As compared to Eq. (3), the number of binary operations does not increase.

5 Experiments

We perform ablation study of our method on ImageNet ILSVRC 2012 in Section 5.1, then in Section 5.2 we compare our method with state-of-the-art on ImageNet for image classification. It is worth noting that we do not use sophisticated training techniques such as knowledge distillation and gradient approximation for binary weights. For binary network training on ImageNet, we set the initial learning rate as 0.001, with a cosine learning rate scheduler. We train the models for 150 epochs with the Adam optimizer, batch size 256, and weight decay 5e-6. For our proposed methods, the binary mask variables are updated together with weights without warm up.

5.1 Ablation Study

5.1.1 Which training strategy is better

Binarization of real-valued weights/activations both introduce approximation error to the information flow, which in turn results in accuracy degradation. Fig. 3 shows the validation accuracy curves when training binary EBN and RealToBinary network for the ImageNet under three settings: (1) Train from scratch for 150 epochs by binarizing weights and activations simultaneously; (2) BAN-BWN: two-stage training by binarizing the activations for the first 75 epochs then binarizing the weights for the second 75 epochs; (3) BWN-BAN: two-stage training by binarizing weights for the first 75 epochs then binarizing activations for the second 75 epochs. As one can see, BWN-BAN

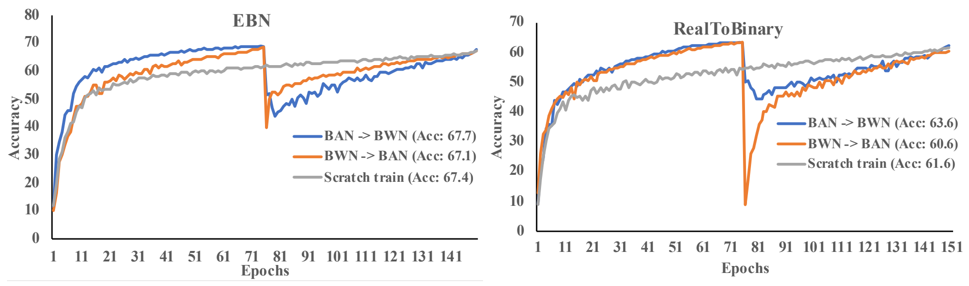


Figure 3: Training curves of EBN and RealToBinary. The final accuracy of a method is in its legend.

	APoT	ABCNet	LQNets	DoRefaNet	Ours
Quantization	non-uniform	learnable	learnable	uniform	uniform
Clip	Yes	Yes	No	Yes	Yes
Binarized	Yes	Yes	Yes	Yes	Yes
Binary representation	Fixed	Learned	Fixed	Fixed	Fixed
Base	$(2^{-1}, 2^{-2}, 2^{-3})$	Learned	Learned	$(\frac{1}{2^2-1}, \frac{1}{2^1-1})$	$(2^{-1}, 2^{-2})$
Accuracy	62.9	36.0	61.5	62.3	62.6

Table 1: Comparison results between various methods for ImageNet classification using RealToBinary by setting the number of groups as 2 and activation bit-width as 2 for the 3*3 convolution.

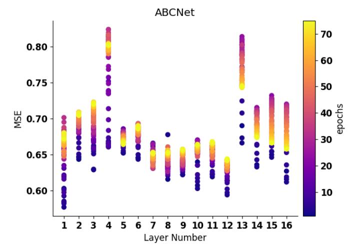
caused a significant accuracy drop when binarizing the activations at the 76^{th} training epoch after the 1^{st} stage of binarizing weights and the drop is much greater than the case of weight binarization in BAN-BWN. Though the accuracy of BWN-BAN restores gradually, we observe that it performs significantly worse than the other two cases, in which BAN-BWN is slightly better than Train from scratch. This may explain why the state-of-the-art methods ReActNet (Liu et al. (2020)) and RealToBinary (Martinez et al. (2020)) adopted the BAN-BWN training strategy. It might be due to that activations are more sensitive to binarization as compared to the weights, and searching the optimal parameters from quantizing activations is easier.

5.1.2 Comparison with benchmark quantization methods

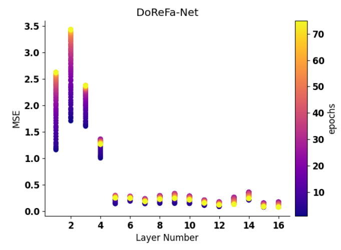
To compare with various benchmark quantization approaches, we conduct experiments on ImageNet with the network RealToBinary by setting activation as 2 bits for 3*3 convolution with the number of group convolutions as 2. Tab. 1 shows the comparison results. APoT, DoRefaNet and ours have fixed base and binary representation. Although APoT requires 2-bit representation, it indeed requires 3 bits when it is expressed in terms of binary representation. The detailed proof is shown in Appendix A.2. ABCNet and LQNets learn the bases by a data-driven way. Our experiments shows that ABCNet learns the same bases for different branches as it does not differentiate the branches, leading to a poor performance. A detailed analysis of a poor performance of ABCNet is shown in Appendix A.3. LQNets has quite different bases for different layers, shown in Fig. 4f. DoRefaNet has pre-defined bases and binary representation. Our method has learnt the base as $(2^{-1}, 2^{-2})$, identified as uniform quantization. Furthermore, our proposed methods show the best result with 2-bit binary representation.

5.1.3 How is the MSE

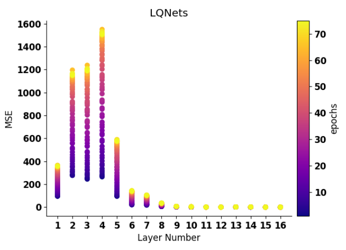
Fig. 4 shows the mean square error (MSE) between original unquantized activation distribution and the corresponding quantized values of various methods under the RealToBinary network setting. The MSE is measured on 1^{st} batch of each epoch for the training dataset at the 1^{st} stage BAN, i.e., quantizing activation with full-precision weights, of the training stage across all the layers. It shows that the MSE of the first several layers is significant among different methods. It might be due to that the real value first passes through a clip function and then do binarization. The real value which is outside of the clipping range without quantization will have larger MSE. Due to the gradient vanishing issue, for the first several layers, it might be difficult to use the data-driven way to enforce the data locating in the clipping range. In particular, LQNets has much larger MSE as compared to other methods for the first several layers, which might indicate the importance of clipping function. Furthermore, the learnt bases of LQNets, like the MSE distribution, are quite



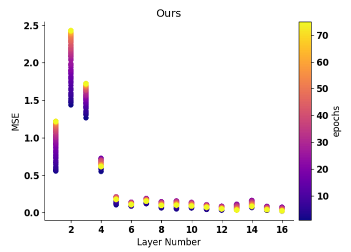
(a)



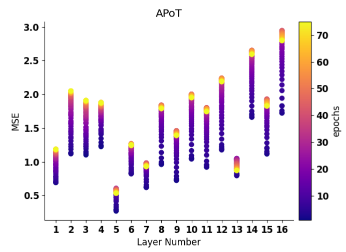
(b)



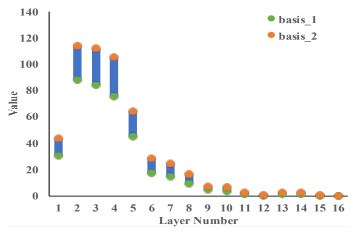
(c)



(d)



(e)



(f)

Figure 4: MSE of various methods on RealToBinary network. (f) shows the learned basis of LQNets.

large for the first several layers, shown in Fig. 4f. Furthermore, we observe that the MSE increases as the epoch increases for some layers. It might indicate that the binarization is diverging with the epoch increases, but due to the clipping, the divergence is not infinite.

5.1.4 What is the effect of the bit-width and grouped convolution

To investigate the effect of the bit-width and grouped convolution to the accuracy and computational cost, we conduct experiments on the network EBN (Bulat et al. (2021). We evaluate the effect of bit-widths for binarized activations N and number of groups for convolutions G using our MABA-Net, with binary mask disabled. We conduct experiments by varying N and G on the backbone network architecture EBN (Bulat et al. (2021)), with configuration $\{1, 2, 6, 2\}\{2\}\{4, 8, 8, 16\}$. As shown in Fig. 5, we set the number of bits for activations of 3×3 convolutions and 1×1 convolutions as $\{1, 2, 3\}$. For the default number of groups for convolution layers in each block (i.e., $\{4, 8, 8, 16\}$), we further increase the numbers by $2\times$, $4\times$ or $8\times$. For instance, the number of groups is increased to $\{16, 32, 32, 64\}$ for the case of $4\times$. As one can see, there is a clear trade-off between network accuracy and the total number of binary operations (BOPS) in a model. Among different group settings, we find that the bit-width configuration $(3, 2)$ is the optimal trade-off between accuracy and BOPS. Though the total number of 1×1 convolutions is much less than the 3×3 convolutions, adding 1 more bit to 1×1 convolutions boosts the accuracy while at the expense of significant increase in BOPS, mainly due to reason that the 1×1 convolutions are not grouped convolutions. E.g., with group size increased by $4\times$, adding 1 bit for the 1×1 convolutions from (3, 2) to (3, 3) introduces more BOPS than adding 1 bit for the 3×3 convolutions from (2, 3) to (3, 3).

5.2 Comparison with the state-of-the-art

To make fair comparison, we compare our proposed methods with state-of-the-art in two network settings: 1) ResNet18: the network architecture is ResNet18 and 2) ResNet-FP: the ResNet variants. We only change the way of binarizing the weights and activations in corresponding network settings. For the 1^{st} network setting, we compare with BNN (Courbariaux et al. (2016)), XNORNet (Rastegari et al. (2016)), CCNN (Xu & Cheung (2019)), Rethink. BNN (Helwegen et al. (2019)), XNOR-NET++ (Bulat & Tzimiropoulos (2019)), IR-Net (Qin et al. (2020)). For the 2^{nd} network setting, we compare with different networks: Bi-Real Net (Liu et al. (2018)), RealToBinary (Martinez et al. (2020)), EBN (Bulat et al. (2021)), ReActNet-A (Liu et al. (2020)). In particular, for Bi-Real Net, RealToBinary and ReActNet-A, we simply set the number of group convolutions for 3*3 convolutions as 2 and the bit-width for the activation as 2. While for EBN, we conduct a fine-grained search, shown in Section 5.1.4.

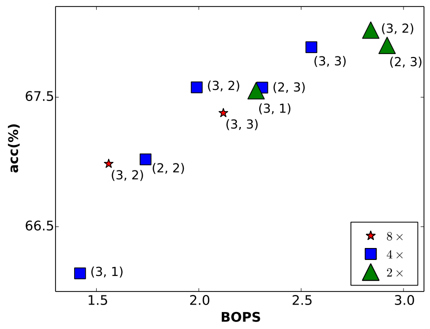


Figure 5: Effect of bit-widths for binarized activations and number of groups for convolutions in our MABA-Net trained on ImageNet. $(\cdot, \cdot)$ denotes the number of bits for activations of 3×3 convolutions and 1×1 convolutions, respectively. On top of the network EBN, we further increase the number of groups by $2\times$, $4\times$ or $8\times$.

Table 2: Comparison with state-of-the-art binary networks on ImageNet.

Architecture		Accuracy (%)		Operations		Memory		
		Top-1	**Top-5**	**BOPS** $\times 10^9$	**FLOPS** $\times 10^8$	**# W** $\times 10^7$	**# A** $\times 10^7$	**# Total** $\times 10^7$
ResNet18	BNN	42.2	69.2	1.7	1.3	1.2	0.2	1.4
	XNOR-Net	51.2	73.2	1.7	1.3	1.2	0.2	1.4
	CCNN	54.2	77.9	1.7	1.3	1.2	0.2	1.4
	Rethink. BNN	56.6	79.4	1.7	1.3	1.2	0.2	1.4
	XNOR-Net++	57.1	79.9	1.7	1.4	1.2	0.2	1.4
	IR-Net	58.1	80.0	1.7	1.3	1.2	0.2	1.4
	Ours	**59.1**	**83.9**	1.7	1.3	**0.6**	0.4	**1.0**
ResNet-FP	Bi-Real Net	56.4	79.5	1.7	1.5	1.2	**0.2**	1.4
	Bi-Real Net (Ours)	**58.9**	**81.3**	1.7	1.5	**0.6**	0.4	1.0
	Bi-Real Net (Ours)	56.7	79.8	**1.6**	1.5	**0.6**	0.4	**1.0**
	RealToBinary	60.9	-	1.7	1.5	1.2	**0.2**	1.4
	RealToBinary (Ours)	60.9	82.6	**1.6**	1.5	**0.6**	0.4	1.0
	RealToBinary (Ours)	**62.4**	**82.7**	1.7	1.5	**0.6**	0.4	**1.0**
	EBN	67.5	**87.5**	1.7	1.1	3.4	**0.5**	3.9
	EBN (Ours)	**67.7**	87.4	1.7	1.1	**0.4**	1.0	**1.4**
	ReActNet-A	65.4	-	4.8	0.2	2.9	**0.5**	3.4
	ReActNet-A (Ours)	**65.7**	86.1	4.8	0.2	**1.5**	1.0	**2.5**

As shown in Tab. 2, for different network settings, our method improves on top of the currently best performing methods with a large margin with less memory cost. The sizes of weights and activations are balanced, shown in Fig. 1. Although our method does increase the activation size, by $2\times$, the run-time memory of both weights and activations largely reduces.

6 Conclusion and Future Work

In this paper, we propose MABA-Net to approximate real-valued activations by accumulating a limited set of binary activations weighted by power-of-2 factors. A trainable binary mask variable is attached to each spatial position in each set of the binary activations, as an indicator to keep or drop activation features at the spatial position, with negligible overhead for storage of the mask variables. By integrating MABA-Net with a deeper and wider grouped convolutional network, we address the problem of unbalanced distribution of binarized activations/weights, which in turn improves accuracy without incurring an increase in both computational budget and memory usage. In the future, we will explore how to binarize the first several layers to improve the performance and also explore other computational and memory saving techniques such as filter pruning, and incorporate advanced training techniques such as knowledge distillation, to further improve the performance.

REFERENCES

Thalaiyasingam Ajanthan, Kartik Gupta, Philip Torr, Richad Hartley, and Puneet Dokania. Mirror descent view for neural network quantization. In *International Conference on Artificial Intelligence and Statistics*, pp. 2809–2817. PMLR, 2021.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv:1308.3432*, 2013a.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013b.

Joseph Bethge, Christian Bartz, Haojin Yang, Ying Chen, and Christoph Meinel. Meliusnet: An improved network architecture for binary neural networks. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1439–1448, 2021.

Adrian Bulat and Georgios Tzimiropoulos. XNOR-Net++: Improved binary neural networks. In *BMVC*, 2019.

Adrian Bulat, Brais Martinez, and Georgios Tzimiropoulos. High-capacity expert binary networks. In *ICLR*, 2021.

Ting-Wu Chin, Ruizhou Ding, Cha Zhang, and Diana Marculescu. Towards efficient model compression via learned global ranking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1518–1528, 2020.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *NeurIPS*, 2015.

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. *arXiv preprint arXiv:1602.02830*, 2016.

Petros Drineas, Ravi Kannan, and Michael W Mahoney. Fast monte carlo algorithms for matrices i: Approximating matrix multiplication. *SIAM Journal on Computing*, 36(1):132–157, 2006.

Mostafa Elhoushi, Zihao Chen, Farhan Shafiq, Ye Henry Tian, and Joey Yiwei Li. Deepshift: Towards multiplication-less neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2359–2368, 2021.

Joshua Fromm, Meghan Cowan, Matthai Philipose, Luis Ceze, and Shwetak Patel. Riptide: Fast end-to-end binarized neural networks. In *Proceedings of Machine Learning and Systems 2020*, 2020.

Denis Gudovskiy and Luca Rigazio. ShiftCNN: Generalized low-precision architecture for inference of convolutional neural networks. *arXiv preprint arXiv:1706.02393*, 2017.

Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International conference on machine learning*, pp. 1737–1746. PMLR, 2015.

Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. Eie: Efficient inference engine on compressed deep neural network. *ACM SIGARCH Computer Architecture News*, 44(3):243–254, 2016a.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*, 2016b.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 1389–1397, 2017.

Koen Helwegen, James Widdicombe, Lukas Geiger, Zechun Liu, Kwang-Ting Cheng, and Roeland Nusselder. Latent weights do not exist: Rethinking binarized neural network optimization. In *NeurIPS*, 2019.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. *Advances in neural information processing systems*, 29, 2016.

William Kahan. Ieee standard 754 for binary floating-point arithmetic. *Lecture Notes on the Status of IEEE*, 754(94720-1776):11, 1996.

Dahyun Kim, Kunal Pratap Singh, and Jonghyun Choi. Learning architectures for binary networks. In *European Conference on Computer Vision*, pp. 575–591. Springer, 2020a.

Hyungjun Kim, Kyungsu Kim, Jinseok Kim, and Jae-Joon Kim. Binaryduo: Reducing gradient mismatch in binary activation network by coupling binary activations. *arXiv preprint arXiv:2002.06517*, 2020b.

Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham Kakade, and Ali Farhadi. Soft threshold weight reparameterization for learnable sparsity. In *International Conference on Machine Learning*, pp. 5544–5555. PMLR, 2020.

Carl Lemaire, Andrew Achkar, and Pierre-Marc Jodoin. Structured pruning of neural networks with budget-aware regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9108–9116, 2019.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *ICLR*, 2016.

Yuhang Li, Xin Dong, and Wei Wang. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. In *ICLR*, 2020. URL `https://openreview.net/forum?id=BkgXT24tDS`.

Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. In *NeurIPS*, pp. 345–353, 2017.

Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-Real Net: Enhancing the performance of 1-bit CNNs with improved representational capability and advanced training algorithm. In *ECCV*, pp. 747–763, 2018.

Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. Reactnet: Towards precise binary neural network with generalized activation functions. In *ECCV*, 2020.

Zechun Liu, Zhiqiang Shen, Shichao Li, Koen Helwegen, Dong Huang, and Kwang-Ting Cheng. How do adam and training strategies help bnns optimization. In *International Conference on Machine Learning*, pp. 6936–6946. PMLR, 2021.

Zechun Liu, Kwang-Ting Cheng, Dong Huang, Eric P Xing, and Zhiqiang Shen. Nonuniform-to-uniform quantization: Towards accurate quantization via generalized straight-through estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4942–4952, 2022.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pp. 2736–2744, 2017.

Michael W Mahoney et al. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning*, 3(2):123–224, 2011.

Brais Martinez, Jing Yang, Adrian Bulat, and Georgios Tzimiropoulos. Training binary neural networks with real-to-binary convolutions. *ICLR*, 2020.

Hai Phan, Zechun Liu, Dang Huynh, Marios Savvides, Kwang-Ting Cheng, and Zhiqiang Shen. Binarizing mobilenet via evolution-based searching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13420–13429, 2020.

Haotong Qin, Ruihao Gong, Xianglong Liu, Mingzhu Shen, Ziran Wei, Fengwei Yu, and Jingkuan Song. Forward and backward information retention for accurate binary neural networks. In *CVPR*, pp. 2250–2259, 2020.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: ImageNet classification using binary convolutional neural networks. In *ECCV*, 2016.

Arthur J Redfern, Lijun Zhu, and Molly K Newquist. Bcnn: A binary cnn with all matrix ops quantized to 1 bit precision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4604–4612, 2021.

Wei Tang, Gang Hua, and Liang Wang. How to train a compact binary neural network with high accuracy? In *Thirty-First AAAI conference on artificial intelligence*, 2017.

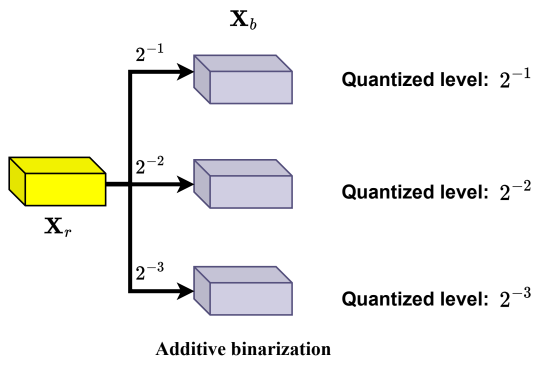
Zhe Xu and Ray CC Cheung. Accurate and compact convolutional neural networks with trained binarization. *BMVC*, 2019.

Hongyi Yao, Pu Li, Jian Cao, Xiangcheng Liu, Chenying Xie, and Bingzhang Wang. Rapq: Rescuing accuracy for power-of-two low-bit post-training quantization. In Lud De Raedt (ed.), *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pp. 1573–1579. International Joint Conferences on Artificial Intelligence Organization, 7 2022. doi: 10.24963/ijcai.2022/219. URL `https://doi.org/10.24963/ijcai.2022/219`. Main Track.

Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *CoRR*, abs/1606.06160, 2016.

Shilin Zhu, Xin Dong, and Hao Su. Binary ensemble neural network: More bits per network or more networks per bit? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4923–4932, 2019.

Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian Reid. Structured binary neural networks for accurate image classification and semantic segmentation. In *CVPR*, 2019.



$\mathbf{X}_b$
2^{-1}
2^{-2}
2^{-3}
$\mathbf{X}_r$
Quantized level: 2^{-1}
Quantized level: 2^{-2}
Quantized level: 2^{-3}
Additive binarization

Figure 6: APoT decomposition in binary encodings.

A Appendix

A.1 Binarize Activations

Importance sampling Different from the standard importance sampling approaches of approximating the matrix multiplication, which randomly select the index i_t based on the probability distribution of p_{i_t}, we select the index i_t with the top probabilities to make it deterministic.

Besides, the approximation algorithm sets the t^{th} column of $\mathbf{X}_b$ as $\mathbf{X}_b^{(t)} = \mathbf{x}_B^{(i_t)} / \sqrt{Sp_{i_t}}$ and the t^{th} row of Q as $Q_{(t)} = D_{(i_t)} / \sqrt{Sp_{it}}$. As in the matrix multiplication of $\mathbf{X}_b$ and Q, only the elemnts of t^{th} column of $\mathbf{X}_b$ multiplies with the elements of the t^{th} row of Q, we merge the numerator of $\mathbf{X}_b$ to Q and get $\mathbf{X}_b^{(t)} = \mathbf{X}_B^{(i_t)}$, $Q_{(t)} = D_{(i_t)} / Sp_{i_t}$.

Binary matrix computation When representing the real-valued $\mathbf{X}_r$ in fixed-point binary presentation, the binary matrix $\mathbf{X}_b$ can be computed as below:

$$\mathbf{X}_b^{(i_t)} = (\mathbf{X}_r << i_t)) \ \ \& \ \ 1 \tag{13}$$

Through shifting $\mathbf{X}_r$ by i_t bits and bit-wise AND operation with 1, the ${i_t}^{th}$ most significant bit is obtained. For example, 0.7865 represented in binary format is 0.1011_2. If we left shifting $0.10\underline{\mathbf{1}}1_2$ by 3 bits, and bit-wise AND with 1, we will get the 3^{rd} most significant bit.

A.2 APoT vs Ours

APoT defines a quantized value set U and then selects the quantized value that is closest to the real value x. When only 2 bits are used, APoT has quantization levels: $U = \{0, 2^{-1}, 2^{-2}, 2^{-3}\}$. APoT then selects the base that results in smallest quantization error. The procedure can be succinctly expressed with following formula:

$$\hat{x} = \underbrace{[0, 2^{-1}, 2^{-2}, 2^{-3}]}_{\text{Basis}} \cdot \underbrace{\begin{bmatrix} \mathbb{1}_{\{\arg\min_{0 \leqslant i \leqslant 3} |x - U_i| = 0\}} \\ \mathbb{1}_{\{\arg\min_{0 \leqslant i \leqslant 3} |x - U_i| = 1\}} \\ \mathbb{1}_{\{\arg\min_{0 \leqslant i \leqslant 3} |x - U_i| = 2\}} \\ \mathbb{1}_{\{\arg\min_{0 \leqslant i \leqslant 3} |x - U_i| = 3\}} \end{bmatrix}}_{\text{binary representation}} \tag{14}$$

Where x is the real-value input of APoT quantization, $\hat{x}$ is the quantized output, and $\mathbb{1}(\cdot)$ is an indicator function to select the base that has the smallest quantization error.

As illustrated in Fig. 6, APoT requires 3 binary activations bases, which correspond to the bases $\{2^{-1}, 2^{-2}, 2^{-3}\}$. Although APoT only uses 2-bit representation in the case, it indeed requires 3 bits when it is expressed in terms of binary representation. This is due to the intrinsic nature of APoT quantization which is not tailored for binary quantization. In contrast, with tailored

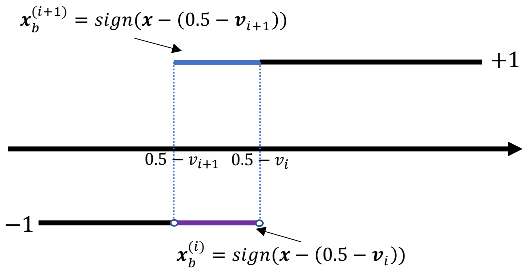

$x_b^{(i+1)} = sign(x - (0.5 - v_{i+1}))$
+1
$0.5 - v_{i+1}$
$0.5 - v_i$
−1
$x_b^{(i)} = sign(x - (0.5 - v_i))$

Figure 7: Visualization of x_b^i and x_b^{i+1} given the same x.

binarization technique for activation, our method achieves comparable performance to APoT with only 2 binary branches, resulting in 1 bit reduction, as shown in Table 1.

A.3 ABCNet vs Ours

Given a real-valued input x, ABCNet compute the binarized value x_b^i of the i branch as follows:

$$x_b^i = sign(h_{v_i}(x) - (0.5 - v_i)) \tag{15}$$

where v_i is the shifting scalar for the i^{th} branch and $h_{v_i}(x) = \text{clip}(x + v_i, (0, 1))$. Suppose the scalar parameters v_i is sorted, namely, $v_i < v_{i+1}$. As shown in Fig. 7, given the same input x, x_b^i and x_b^{i+1} are different only when $0.5 - v_{i+1} \leqslant x < 0.5 - v_i$ in Eq. 16. If v_i and v_{i+1} are very close, branches i and $i + 1$ will have a lot of same values and thus are highly correlated.

$$\begin{cases} x_b^i = x_b^{i+1} = 1 & \text{when } x \geqslant 0.5 - v_i \\ x_b^i = x_b^{i+1} = -1 & \text{when } x < 0.5 - v_{i+1} \\ x_b^i = -1, x_b^{i+1} = 1 & \text{when } 0.5 - v_{i+1} \leqslant x < 0.5 - v_i \end{cases} \tag{16}$$

ABCNet does not specify the gradient for the shifting parameter v_i; however, it can be deduced as follow: $\frac{\partial \mathcal{L}}{\partial v_i} = \frac{\partial \mathcal{L}}{\partial \mathbf{X}_b^i} \times \mathbb{1}_{0 \leqslant \mathbf{X}_r - v_i \leqslant 1}$ where $\mathcal{L}$ denotes the loss function, $\mathbf{X}_b^i$ denotes the binarized activations of the i^{th} branch and $\mathbf{X}_r$ denotes the real-valued activations. For simplification, we assume branch i and branch j has the same scaling factor and then $\frac{\partial \mathcal{L}}{\partial \mathbf{X}_b^i} = \frac{\partial \mathcal{L}}{\partial \mathbf{X}_b^j}$. Therefore, if v_i and v_j are initialised close to each other, their gradients, $\frac{\partial \mathcal{L}}{\partial v_i}$ and $\frac{\partial \mathcal{L}}{\partial v_j}$, will be close. Thus if v_i and v_j are kept close, based on Eq. (15), $\mathbf{X}_b^i$ and $\mathbf{X}_b^j$ are also close. This effect extends to the learning of the scaling factor β_i, too. The gradient of β_i is calculated as follows: $\frac{\partial \mathcal{L}}{\partial \beta_i} = \frac{\partial \mathcal{L}}{\partial \overline{A}} \times \frac{\partial \overline{A}}{\partial \beta_i} = \frac{\partial \mathcal{L}}{\partial \overline{A}} \times \mathbf{X}_b^i$, where $\overline{A} = \sum_i \beta_i * \mathbf{X}_b^i$. If $\mathbf{X}_b^i$ and $\mathbf{X}_b^j$ are close to each other, then the gradients of β_i and β_j are also close. So the learnt β_i and β_j would be close. In the end, we can see that the performance of ABCNet is really affected by the initialization of the shifting parameter and the scaling factor.

A.4 Spatial mask visualization

Our proposed spatial masking belongs to the category of structured pruning as it prunes the whole channel of the activations when the mask equals to 0. Fig. 8 shows the pruning ratio of the learnt spatial masking and corresponding BOPS of our proposed methods across different layers on EBN network with the optimal setting of 3 branches for the 3×3 convolutions, 2 branches for 2×2 convolutions and $4\times$ grouped convolutions for 3×3 convolutions. We make few observations from the plot. Firstly, the activation of the first and last several network layers are pruned more than other layers. Secondly, it shows that the activation of the layer with larger computational cost (BOPS) are pruned more than other layers, which is consistent with our motivation of computation cost based pruning. Thirdly, we observe that the branch with smaller base has larger pruning ratio. It might be due to that the branch of the larger base plays a more important role in approximating the real values.

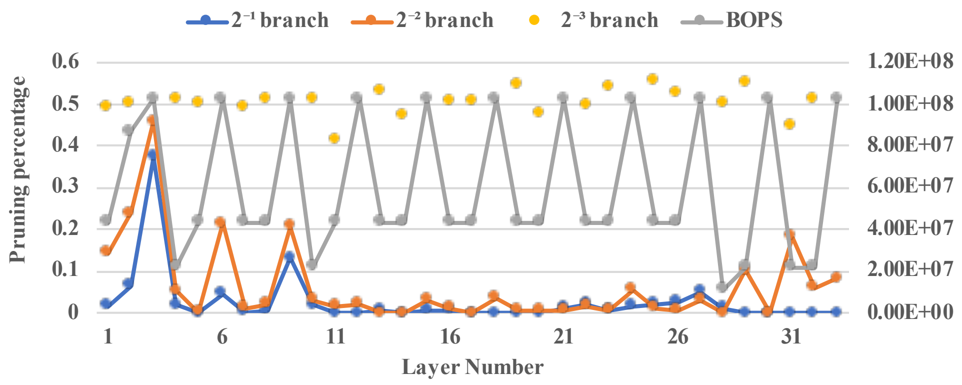


Figure 8: Visualization of the pruning ratio of the learnt spatial masking and corresponding BOPS of our proposed methods across different layers on EBN network.