Trimmable Wide Binary Neural Network

Anonymous Author(s)
Affiliation
Address
email

Abstract

The expressive power of a binary neural network is highly dependent on the approximation error incurred when mapping its real-valued counterpart to binary. Expanding a binary network via a constant uniform width multiplier is shown to reduce the approximation error, but come at the expense of increased inference time and memory footprint at runtime. In this work, we present Trimmable Wide Binary Neural Network (TWB-Net) to automate the design of accurate, fast and memory-efficient binary networks, via coarse-grained structure expansion and pruning in an end-to-end manner. First, we provide a theoretical analysis to justify the superiority of structure expansion at coarser level for binary network. Second, we propose a method to automatically learn to prune redundant structures from a width-expanded binary network via a simple differentiable soft gate function with sparsity regularization, permitting structure-wise non-uniform width multipliers. Unlike hard pruning with the Heaviside step function, we demonstrate that the sparsity regularized soft gate function is able to stabilize the training of binary networks with guaranteed structure sparsity. Our evaluations show that TWB-Net outperforms state-of-the-art by significantly reducing inference time and memory footprint, while maintaining similar or better accuracy.

1 Introduction

Convolutional neural network has witnessed great success in a wide range of applications, but comes at significantly increased usage of memory and compute. It is particularly challenging to deploy such network on resource-limited mobile or embedded device. Neural network compression [12] has emerged as the mainstream technique to reduce the memory and compute, by either quantizing real-valued weight filters and activations to low precision numbers [12, 41, 42, 18, 40] or pruning redundant weights or neurons from the network architecture [12, 21, 26, 38]. Binary Neural Network (BNN) [5, 6, 30], as the most extreme form of quantization, represents both weight filters and activations as binary values (e.g., +1 and -1). This does not only significantly reduce memory footprint but more importantly replace real-valued matrix multiplication with ultra-fast bit-wise operations XNOR and POPCOUNT.

Despite the promising advantages over full-precision network in terms of memory and compute, the biggest limitation of a vanilla BNN [5, 6, 30] is the significant accuracy degradation, especially for complex tasks such as ImageNet. The degradation is mainly due to the approximation error incurred by real-to-binary projections. Few ideas [22] are attempted to reduce the error propagation by directly approximating real-valued weight filters and activations with multiple binary bases, which in turn narrow down the accuracy gap. Very recently, structure approximation [43, 44] is proposed to alleviate the error propagation by parallel expanding specific structure (e.g., residual block) in a network with a constant uniform width multiplier, which achieves superior accuracy. Nevertheless, all of these approaches trade off the reduction rate of memory and compute for accuracy improvement in a heuristic manner.

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

In this work, we aim to automate the design of accurate, fast and memory-efficient binary neural networks. Our contributions are three-fold:

- We present a general framework, called Trimmable Wide Binary Neural Network (TWB-Net), to automatically co-optimize accuracy, memory and compute by learning to prune redundant structures from a width-expanded binary network in an end-to-end manner. Compared to previous approaches, TWB-Net enables structure-wise non-uniform width multipliers, which in turn offers significant higher reduction of memory and compute, while achieving comparable or better accuracy.
- We provide a theoretical analysis to justify the superiority of structure expansion at coarser level for binary network, which is validated by empirical observations.
- We introduce a simple differentiable soft gate function to suppress and prune less important structures in a wide binary network. A sparsity constraint is added on the soft gate function, which helps maximize the pruning ratio. Unlike hard pruning with the Heaviside step function, we demonstrate that the sparsity-regularized soft gate function is able to stabilize the training of binary networks with coarse-grained structure sparsity.

We evaluate TWB-Net on image classification (CIFAR-10 and ImageNet) and semantic segmentation (PASCAL VOC 2012), with well-known network architectures as backbone. We also benchmark the inference time and peak memory of variants of binary networks on an embedded platform Raspberry Pi 4. Our evaluations show that TWB-Net outperforms state-of-the-art by largely reducing memory footprint and inference time, while maintaining similar or better accuracy.

2 Related Work

Binary Neural Network, as the most extreme case of neural network quantization, suffers from significant degradation in accuracy especially for complicated tasks such as ImageNet classification [5, 6, 30, 35, 11]. To bridge the accuracy gap, one line of research [25, 24, 29] is to build a strong baseline binary network by carefully tweaking the backbone network or adjusting the optimization procedure, such as variants of ReLU function [13], real-valued downsampling layers, 2-stage training strategy which binarizes the activations first and then the weight filters, or knowledge distillation [15] from a real-valued teacher network to a binary student network. Our method is orthogonal to these approaches in the sense that we start with standard baseline binary network built by XNOR-Net [30] without recent advances. We believe a combination of stronger baseline binary network and our method can lead to better performance.

Another line of research is to improve accuracy by expanding the width of binary network at different levels of structure granularity with a uniform width multiplier, e.g., network-wise [43], block-wise and layer-wise [44]. In contrast to these approaches, we automate the design of binary network with structure-wise non-uniform width multipliers, resulting in improved reduction of memory footprint and inference time while with comparable or better accuracy.

Neural Network Pruning reduces the number of operations by removing weights or neurons from network architecture, which can be grouped into saliency-based pruning [12, 21, 38] or sparsity learning based pruning [36, 26, 28, 10, 17]. The former shrinks network size by defining hard threshold for pruning criterion such as magnitude of weights or L1/L2 norm of filters. The latter prunes network by adding sparsity regularization into training loss (e.g., L0/L1 or Group Lasso), while a threshold is still required in order to generate the final pruned model. Our work differs from these works in several aspects. First, our method falls in the sparsity learning based pruning, but we focus on learning coarse-grained structure sparsity in the context of binary networks, rather than full-precision networks. Second, in contrast to most of the pruning approaches, our method is threshold-free thanks to the soft gate function introduced in our framework, which avoids threshold tuning for various pruning criterion.

Neural Architecture Search (NAS). Unlike hand-crafted designed networks like AlexNet [20], VGG [33], ResNet [14] and MobileNets [16], NAS has attracted a lot of attention in recent years. The main stream research on NAS is automatic design of micro and macro structures in a network, using reinforcement learning [45], gradient search [23] or evolutionary search [31]. Recent work [27] suggested that structured network pruning can also be treated as a NAS problem, with the objective of searching width/depth of the pruned network. Very recently, EfficientNet [34] used NAS to design

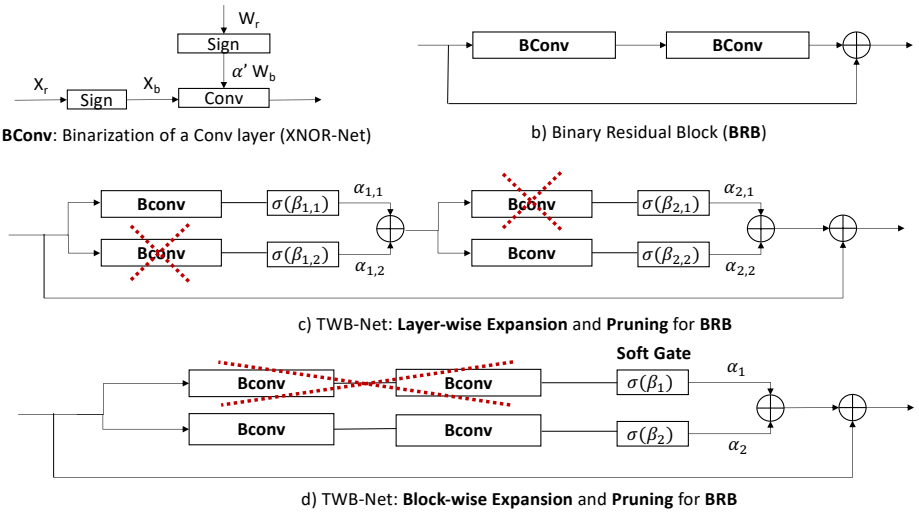

Figure 1: Illustrations of (a) Binarization of a convolution layer in XNOR-Net [30]; (b) Binary Residual Block (BRB); (c) TWB-Net: Layer-wise Expansion and Pruning for a BRB; (d) TWB-Net: Block-wise Expansion and Pruning for a BRB.

a strong baseline network, followed by a heuristic method to scale up/down depth/width/resolution of the baseline network uniformly, yielding much better performance than state-of-the-art. Similarly, our work aims to search structure-wise non-uniform width multipliers for a binary network in an end-to-end manner.

3 Method

In this section, we first revisit basics in binary network in Section 3.1. Then we introduce the coarse-grained structure expansion for wide binary network and the soft gate function for coarse-grained structure pruning in Section 3.2 and Section 3.3, followed by sparsity regularization added to the soft gate function for learning structured sparsity in binary network in Section 3.4.

3.1 Preliminary on Binary Neural Network

Let $\mathbf{W} \in \mathbb{R}^{k\times k\times c\times f}$ and $\mathbf{X} \in \mathbb{R}^{w\times h\times c}$ represent the matrices of weight filters and input activations for a layer, k, c, f, w and h denote kernel size, number of input channels, number of filters, width and height of input activations. As illustrated in Fig. 1(a), binary network attempts to project real-valued $\mathbf{W}_r$ and $\mathbf{X}_r$ to binary matrices $\mathbf{W}_b$ and $\mathbf{X}_b$ with either -1 or 1, by applying a sign function $\mathbf{W}_b = sign(\mathbf{W}_r)$ and $\mathbf{X}_b = sign(\mathbf{X}_r)$. Compared to full-precision network, binary network reduces the memory footprint by 32x, meanwhile, it replaces floating point matrix Multiple-Accumulate $\mathbf{W}_r * \mathbf{X}_r = \sum w_r * x_r$ with bit-wise XNOR-POPCOUNT-Accumulate $\mathbf{W}_b * \mathbf{X}_b = \sum w_b * x_b = \sum popcount(XNOR(w_b, x_b))$, which reduces the number of operations by 43x [9], i.e., from $(2w^{'}h^{'}fkkc)$ to $(3w^{'}h^{'}fkkc/64)$ of which $w^{'}$ and $h^{'}$ denote the width and height of output activations [1].

Binarization of real-valued network introduces considerable approximation error that is accumulated across layers, which in turn causes significant accuracy drop. XNOR-Net [30] and ABC-Net [22] directly approximate real-valued $\mathbf{W}_r$ and $\mathbf{X}_r$ via aggregating multiple binary bases $\sum_k^{K_w} \alpha_k^{'} \mathbf{W}_{b,k}$ and $\sum_k^{K_x} \beta_k^{'} \mathbf{X}_{b,k}$, weighted by floating point scaling factor $\alpha_k^{'}$ and $\beta_k^{'}$, $K_w \geq 1, K_x \geq 1$, which is shown to alleviate the error propagation and improve accuracy by a large margin. However, there is still a clear accuracy gap to its full-precision counterpart. Moreover, the operations become $\sum_k^{K_w} \alpha_k \mathbf{W}_{b,k} * \sum_k^{K_x} \beta_k \mathbf{X}_{b,k}$, the savings of memory footprint and operations over full-precision network are reduced to $32/K_w$ and $43/(K_w K_x)$.

[1] Following [9], we count each multiplication, addition, XNOR and POPCOUNT as one operation.

3.2 Coarse-grained Structure Expansion

An alternative strategy to minimize the approximation error is linear combination of parallel expanded structures in a binary network, which achieves superior accuracy compared to multiple binary approximations of real-valued weight filters and activations (see Section 4.1).

$$\mathbf{Y}_s = \sum_{m}^{M} \alpha_{s,m} \mathbb{S}_{s,m}, \tag{1}$$

where the width of binary network is increased by parallel expanding a coarse-grained structure $\mathbb{S}_s$ (e.g., block and layer), M is a constant width multiplier. The output $\mathbf{Y}_s$ of the s^{th} structure $\mathbb{S}_s$ is aggregated from the M parallel paths $\mathbb{S}_{s,m}$, weighted by floating point scaling factors $\alpha_{s,m}$. Fig. 1(c)-(d) show an example of layer-wise and block-wise expansion and aggregation for a typical residual block, e.g., a block-wise expansion gives $\mathbb{S} = \alpha^{'}_{conv2}\mathbf{W}_{b,conv2} * sign(\alpha^{'}_{conv1}\mathbf{W}_{b,conv1} * \mathbf{X}_{b,conv1})$ when $K_w = 1$. Wide BNN with layer-wise and block-wise expansion saves memory footprint and operations by $32/M$ and $43/M$ respectively, while layer-wise reports lower reduction rate $32/M^2$ for memory footprint and $43/M^2$ for operations.

It is perhaps surprising that filter-wise expansion performs much worse than layer-wise and block-wise expansion, even though it has larger model capacity (see Section 4.2). To understand this better, we theoretically analyze the effect of structure-wise expansion on the robustness [37] and stability [4] properties of a binary network classifier, inspired by the classical learning theory [37, 4]. In particular, robustness and stability are defined as the sensitivity of the classifier with respect to perturbed inputs (e.g. Gaussian noise) and perturbed weight parameters respectively, which are closely related to the generalization ability of learning algorithm. In this work, we are interested in structure expansion at three levels of granularity: block-wise, layer-wise and filter-wise. Without loss of generality, we first analyze a typical binary residual block (see Fig. 1(b)), the following proposition can be derived:

Proposition 1 *Given a residual block with real-valued weights $\mathbf{W}_r \sim N(0, var(\mathbf{W}_r))$ and normalized activations $\mathbf{X}_r \sim N(0, 1)$, the residual block is expanded by M times at filter level, layer level and block level with scaling factors α^f, α^l and α^b respectively. Assume input perturbation is $\Delta\mathbf{X} \sim N(0, var(\Delta\mathbf{X}))$, $\mathbf{W}_r$ and $\mathbf{X}_r$ are binarized as $\mathbf{W}_b$ and $\mathbf{X}_b$ with sign function, then*

1. *Layer-wise expanded binary residual block is more robust than filter-wise expanded binary residual block if the variance of layer-wise scaling factors $var\alpha^l < 1$.*
2. *Block-wise expanded binary residual block is more robust than layer-wise expanded binary residual block if $M * F > \frac{var(\alpha^b)}{var(\alpha^l)^2}$, where F is the number of filters in the Convolutional layer in the residual block.*

The principle can be applied to stability property analysis by simply replacing input perturbation Δx with weight perturbation Δw. Moreover, the conclusion can be extended to deeper network with a linear stack of layers or blocks. A complete theoretical analysis can be found in the supplementary material. Finally, we provide empirical analysis to verify the proposition in Section 4.2.

3.3 Coarse-grained Structure Pruning

A wider binary network significantly improves the accuracy, but comes at the expense of increased number of operations as well as memory footprint. Instead of applying a uniform width multiplier M to all structures in a binary network, we propose coarse-grained structure pruning to automatically search a binary network with non-uniform width multipliers, with the objective of maximizing the reduction of operations and memory footprint, while maintaining state-of-the-art accuracy. Formally, we introduce a soft gate function $\sigma(\cdot)$ to suppress and prune less informative parallel paths in the s^{th} coarse-grained structure $\mathbb{S}_s$:

$$\begin{aligned} \mathbf{Y}_s &= \sum_{m}^{M} \sigma(\beta_{s,m}) \alpha_{s,m} \mathbb{S}_{s,m}, \\ \sigma(\beta_{s,m}) &= \min(\max(\gamma\beta_{s,m} + 0.5, 0), 1), \end{aligned} \tag{2}$$

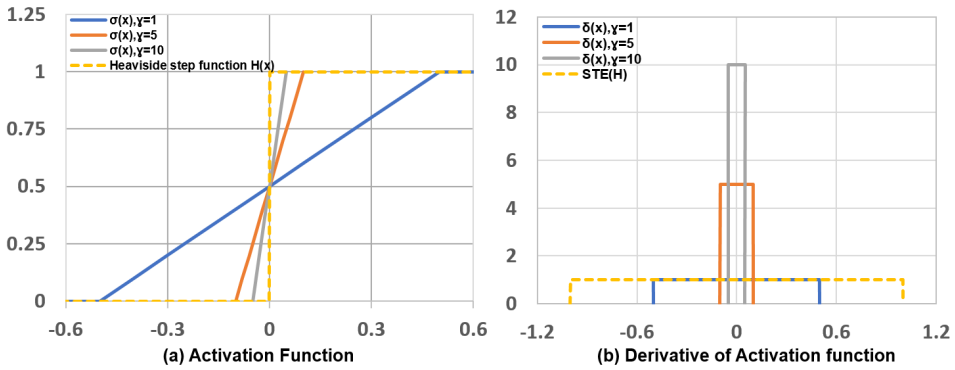

Figure 2: (a) Approximating the Heaviside Step Function with Soft Gate Function with high temperatures; (b) Derivative of the Soft Gate Function with different temperatures. Derivative of the Heaviside Step Function is approximated by Straight-Through Estimator (STE)[1].

where β_s is light-weight trainable gate variables tied to the s^{th} structure $\mathbb{S}_s$, the length of β_s equals to the constant width multiplier M. $\sigma(\beta_{s,m})$ indicates the importance score of the m^{th} parallel path in $\mathbb{S}_s$. If $\sigma(\beta_{s,m}) = 0$, the m^{th} parallel path contributes zero to the output activations $\mathbf{Y}_s$, and thus can be removed from the network without hurting performance. The soft gate function $\sigma(\cdot)$ is zero-centered and clamps input $\beta_{s,m}$ to the range of $[0, 1]$, γ denotes a temperature constant which controls the slope of a specific linear interval, as shown in Fig. 2 (a).

The rationale behind the soft gate function $\sigma(\cdot)$ is that it can be used to approximate the binary Heaviside step function $H(\cdot)$. Since the derivative of $H(\cdot)$ is infinite at zero and zero everywhere else, it cannot be integrated into back-propagation. Though the derivative of $H(\cdot)$ can be approximated via Straight-Through Estimator (STE) [1] as shown in Fig. 2 (b), we observed that updating the gate variables β_s with STE makes the training process unstable, resulting in inferior accuracy (see Section 4.2). In contrast, the soft gate function $\sigma(\cdot)$ approaches to $H(\cdot)$ when γ tends to infinity, enabling a smooth approximation of $H(\cdot)$. Empirical observations also show that $\sigma(\cdot)$ is able to stabilize the training of binary networks with guaranteed structure sparsity.

Unlike previous hard pruning approaches where a hand-crafted threshold is required to decide whether or not to prune, the soft gate function $\sigma(\cdot)$ is threshold-free as it only removes parallel paths with $\sigma(\beta_{s,m}) = 0$. Parallel paths with $\sigma(\beta_{s,m}) \neq 0$ are kept and their importance scores $\sigma(\beta_{s,m})$ can be absorbed into the corresponding scaling factors $\alpha_{s,m}$. One may note that we didn't apply the soft gate function directly on the variables $\alpha_{s,m}$, which would limit the dynamic range of scaling factors as $\sigma(\cdot)$ clamps all values to non-negative within $[0, 1]$.

3.4 Sparsity Regularization

To maximize the reduction of operations and memory footprint, the output activations derived by soft gate function $\sigma(\cdot)$ should be sparse, i.e., majority of $\sigma(\beta_{s,m})$ are 0. Thus, we introduce sparsity constraint on $\sigma(\cdot)$ to explicitly enforce $\sigma(\beta_{s,m})$ moving to 0 during training. The overall objective function of our method is formulated as

$$\begin{aligned} \min_{\{\mathbf{W},\alpha',\alpha,\beta\}} \sum_{n}^{N} \mathcal{L}_E(Y_n, \{\mathbf{Y}_1, ..., \mathbf{Y}_S\}) + \lambda \mathcal{L}_S, \\ \mathcal{L}_S = \sum_{s}^{S} \sum_{m}^{M} \sigma(\beta_{s,m}) \end{aligned} \tag{3}$$

where $\mathcal{L}_E$ is the cross-entropy loss for the n^{th} data sample, and $\mathcal{L}_S$ is the regularization loss to enforce sparsity of the output activations derived by the soft gate function. The regularization multiplier λ is introduced to control the regularization strength. One may note that $\mathcal{L}_S$ is computed as the sum of $\sigma(\beta_{s,m})$, which is equivalent to L1 regularization since $\sigma(\beta_{s,m})$ is always non-negative.

Table 1: Comparison with state-of-the-art on CIFAR-10. M: base width multiplier. $\downarrow$: target pruning ratio. Top-1 accuracy for Full Precision VGG-11: 89.79%; ResNet-20: 91.25%; ResNet-32: Top-1 92.49%. For brevity, we only count the number of XNOR oprations.

Network	Method	Top-1 (%)	# Weights ($\times 10^6$)	# XNOR Ops ($\times 10^6$)
VGG-11	XNOR-Net [30]	85.06	9.30	74
	GroupNet (M=2) [44]	87.78	18.60	148
	Ours (M=2, $\downarrow$30%)	87.85	13.13	125.49
ResNet-20	XNOR-Net [30]	84.87	0.27	41.17
	GroupNet (M=3) [44]	86.67	0.81	123.51
	Ours(M=4, $\downarrow$25%)	88.59	0.82	116.05
ResNet-32	XNOR-Net [30]	86.74	0.46	69.12
	GroupNet (M=2) [44]	87.20	0.92	138.24
	Ours (M=3, $\downarrow$35%)	88.91	0.84	126.56

4 Experiment

In Section 4.1, we compare our method with state-of-the-art on CIFAR-10 [19] and ImageNet ILSVRC 2012 [7] for image classification and PASCAL VOC 2012 [8] for semantic segmentation, with different network architectures as backbone, including VGG-16 [33], ResNet-18/20/32/34 [14] and FCN [32]. Then we perform in-depth analysis of our method in Section 4.2. For fair comparison, the binarization of weight filters and activations simply follows standard XNOR-Net [30], without any modifications on baseline binary network or advanced training strategies used in [25, 24, 29]. We expand the baseline binary network and train a full precision width-expanded network with SGD optimizer, then retrain the binarized counterpart jointly with gate variables with Adam optimizer. We use standard protocols for data augmentation and learning rate schedule. If not stated otherwise, we use block-wise expansion for our TWB-Net, with temperature γ=1 and regularization multiplier λ=5. More results and implementation details are in the supplementary material.

4.1 Comparison with the state-of-the-art

CIFAR-10. We compare our method with state-of-the-art binary networks with VGG-11, ResNet-20 and ResNet-32 as backbone architectures, as shown in Table 1. We report mean Top-1 accuracy, and observe the standard deviation of 10 runs by varying the random seeds is considerably low ($\sim \pm 0.7\%$). One can see that our method performs consistently better than GroupNet [44] in terms of accuracy, while comes at comparable (and in most of the cases fewer) weights and XNOR operations with different configurations for width multiplier M.

ImageNet. Table 2 compares our method with state-of-the-art binary networks with ResNet-18 and ResNet-34 as backbone architectures. For ResNet-18 with base width multiplier $M = 3$, our method reduces weights and XNOR operations by 40%, while achieves significantly better accuracy than BENN [43] and ABC-Net [22]. Our method performs comparable with GroupNet [44] with 15% to 20% fewer weights and XNOR operations.

PASCAL VOC 2012. We further evaluate the transferability of our method by adapting the image classification model trained on ImageNet to semantic segmentation task on PASCAL VOC 2012, measured in terms of mean Intersection over Union (mIoU). In Table 3, we train FCN-16s and FCN-32s [32] with block-wise expanded ResNet-18 (M=5) as backbone. For each experiment, we compute the mean and standard deviation over 5 runs by varying the random seeds. Compared to GroupNet [44], our method reduces weights by 17.2% and XNOR operations by 18.7%, while achieves similar mIoU. Note that GroupNet reported results with much higher standard deviation.

4.2 In-depth Analysis

Structure-wise Expansion: Robustness and Stability. We analyze the robustness [37] and stability [4] of structure-expanded binary ResNet-20 at three levels of granularity: block-wise, layer-wise and filter-wise. (1) Robustness of binary network is defined as the variation caused by perturbed inputs with Gaussian noise. The smaller the variation, the more robust the model. We measure

Table 2: Comparison with state-of-the-art on ImageNet. K_w, K_x: the number of binary bases for weight filters and activations [22]. M: base width multiplier. $\downarrow$: target pruning ratio. $\star$ represents GroupNet without the learnt soft connections which changed the connections between blocks [44]. Top-1 and Top-5 accuracy for Full Precision ResNet-18: Top-1 69.3%, Top-5 89.2%; ResNet-34: Top-1 73.2%, Top-5 91.4%.

Network	Method	Top-1 (%)	Top-5 (%)	# Weights ($\times 10^6$)	# XNOR Ops ($\times 10^9$)
ResNet-18	BNN [6]	42.2	67.1	11.6	1.814
	XNOR-Net [30]	51.2	73.2	11.6	1.814
	BENN (M=3) [43]	53.6	-	34.8	5.442
	ABC-Net (K_w=3,K_x=1) [22]	49.1	67.6	34.8	5.442
	Ours (M=3, $\downarrow$40%)	56.0	78.7	20.9	3.221
	BENN (M=6) [43]	61.0	-	69.6	10.884
	GroupNet (M=5)* [44]	64.1	84.9	58.0	9.070
	Ours (M=5, $\downarrow$15%)	63.5	84.2	48.3	7.363
	Ours (M=5, $\downarrow$40%)	62.5	83.3	32.2	6.149
ResNet-34	BNN [6]	54.1	78.1	21.3	3.663
	GroupNet (M=5)* [44]	65.6	85.9	106.4	18.319
	Ours (M=5, $\downarrow$15%)	65.2	84.9	91.3	15.081

Table 3: Comparison with GroupNet [44] on PASCAL VOC 2012.

Network	Method	mIOU	Δ
ResNet-18, FCN-32s	GroupNet (M=5)* [44]	60.5%	4.4
	Ours (M=5, $\downarrow$15%)	58.7%	0.47
ResNet-18, FCN-16s	GroupNet (M=5)* [44]	62.7%	4.6
	Ours (M=5, $\downarrow$15%)	62.7%	0.67

the robustness in terms of output fluctuation for randomly initialized binary models (Fig. 3(a)) and accuracy fluctuation for trained binary models on CIFAR-10 (Fig. 3(b)). Results show that filter-wise expansion is more prone to input perturbation while block-wise and layer-wise expansion have better robustness. Correspondingly, accuracy of the trained block-wise, layer-wise and filter-wise expanded binary networks are 91.4%, 89.8% and 86.7% respectively. (2) Stability of binary network is defined as the oscillation of prediction during training, given the same inputs. We measure the stability as the variation of accuracy fluctuation for the last 50 iterations after 200 epochs of training. The smaller the variation, the more stable the model. As shown in Fig. 3(c), filter-wise expansion has the worst stability among the three structure-wise expansions. More details on the measurement of robustness and stability can be found in the supplementary material.

Structure-wise Pruning:Soft Gate Function vs. Heaviside Step Function. Fig. 4 demonstrates the effect of the Soft Gate Function (SGF) and the Heaviside Step Function (HSF) on evaluation accuracy and pruning ratio, during training of a ResNet-20 on CIFAR-10. We activate the sparsity regularization loss for SGF (Eq. 3) till Epoch 180. We make the following observations. First, training with the differentiable SGF is much more stable than the HSF with its derivative approximated by

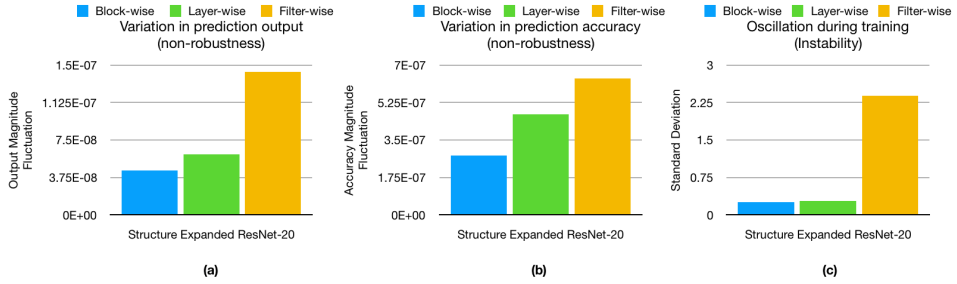

Figure 3: Robustness (a)-(b) and Stability (c) of structure-wise expansion from filter level, layer level to block level, measured with ResNet-20 on CIFAR-10.

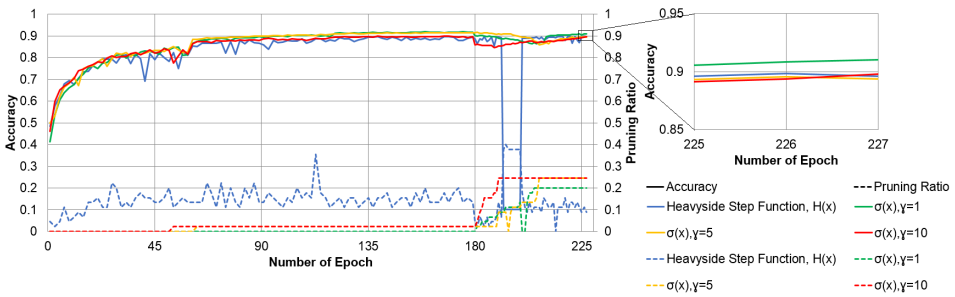

Figure 4: Evaluation accuracy and pruning ratio of ResNet-20 as function of training epochs on CIFAR-10, training with Heaviside Step Function or Soft Gate Function with different temperatures. The sparsity regularization loss in Eq. 3 is activated till Epoch 180.

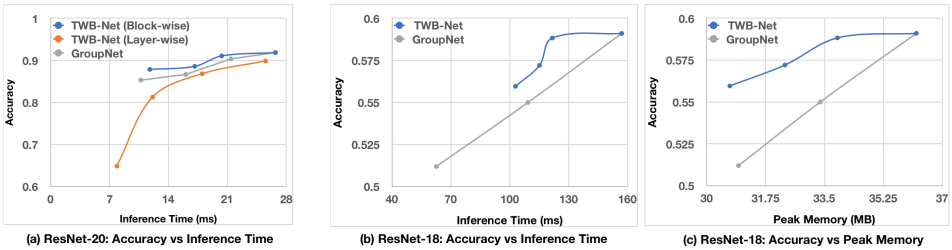

Figure 5: (a): Accuracy versus Inference Time for ResNet-20 (Batch size = 64) on CIFAR-10. (b)-(c): Accuracy versus Inference Time and Peak Memory at Runtime for ResNet-18 (Batch size = 1) on ImageNet.

STE [1], which in turn achieves better accuracy. Moreover, HSF prunes faster and more than SGF at early training epochs without the sparsity regularization loss activated. These phenomena are due to the fact that HSF clamps gate variables to either 0 and 1 while SGF enables a smooth transition from 1 to 0. Second, for SGF, higher temperature leads to faster convergence to the target pruning ratio, especially when the sparsity regularization loss is activated. This is reasonable given that the higher the temperature, the larger the gradients during backpropagation, as shown in Fig. 2. Again, faster convergence on network sparsity comes at the expense of lower accuracy, suggesting that gradual pruning is more favored than aggressive pruning.

Reductions of Inference Time and Peak Memory at Runtime. Besides comparison in terms of number of weights and XNOR operations, we also benchmark our method against GroupNet [44] in terms of inference time and peak memory on practical embedded platform Raspberry Pi 4, leveraging the Larq Compute Engine (LCE) [2] which is a highly optimized inference engine for deploying binary network. As shown in Fig. 5, TWB-Net with block-wise expansion performs much better than TWB-Net with layer-wise expansion at different pruning ratios, suggesting that expansion and pruning at coarser-grained structure benefits the design of accurate and fast binary networks. Compared to GroupNet, TWB-Net with block-wise expansion achieves consistently better performance at all dimensions for both ResNet20 on CIFAR-10 and ResNet-18 on ImageNet.

5 Conclusion

In this work, we propose to automate the design of binary network with non-uniform structure-wise widths, which bridges the accuracy gap to the full-precision counterpart and maximizes the reduction of model size and bit-wise operations simultaneously. This is achieved by coarse-grained structure expansion with theoretical guarantees and stable redundant structure pruning with sparsity constrained soft gate function. We benchmark the inference time and peak memory of binary networks on embedded platform Raspberry Pi 4. Our method performs consistently better than state-of-the-art in consideration of accuracy, inference time and memory footprint at runtime.

[2] https://github.com/larq/compute-engine

Broader Impact

Binary neural network, as the most efficient form of quantization, has the potential to enable 43x inference speedup and 32x lower memory footprint compared to its real-valued counterpart. In spite of these advantages, binary network receives much less attention than the fixed-point quantization techniques (e.g., 8-bit integers for weights and activations), due to several barriers. First and foremost, there is significant accuracy degradation because of the noise introduced by the binarziation of floating point weights and activations. Second, there is a need to design new hardware architectures tailored for binary representations and bit-wise operations, while most of the state-of-the-art AI hardware platforms are customized for floating-point or fixed-point arithmetic. Accordingly, the community devotes tremendous effort to develop complete software toolchains (from compiler to highly optimized libraries like BLAS, CUDNN, TensorRT, TensorFlow Lite, etc) to maximize the inference speedup and energy efficiency of floating-point or fixed-point arithmetic, which unfortunately is not the case for binary network.

There is growing interest in development of new algorithms for training more accurate binary network and design of new AI hardware tailored for binary network from both academic and industry (e.g., xnor.ai and xcore.ai). Our work tries to narrow down the accuracy gap from the perspective of binary network architecture search via defining a constrained search space (i.e., structure-wise non-uniform width multipliers), but we expect to see new paths to further optimize and improve the accuracy of binary network without significantly increased overheads. On the other side, recent work attempted to implement dedicated hardware architecture for vanilla binary network [2]. Finally, it would be exciting to see a full-stack hardware-software co-optimization for binary network, which in turn advances the deployment of extremely energy-efficient and high-performance AI-oriented decision making for a wide range of edge IoT applications, such as autonomous driving, the last mile delivery with drones, remote health monitoring, IoT for smart agriculture, etc.

Binary network benefits in-memory computing [39, 3], a technology being explored as one of the promising directions for the next-generation AI hardware. Most of today's AI hardware follow the Von Neumann architecture, in which data needs to move back and forth between memory and processor. When scaling up the AI workloads, data movements are likely to dominate the execution time and power consumption (a.k.a., the "Memory Wall" problem). In contrast, in-memory computing integrates the compute in memory, by leveraging the emerging non-volatile memory technologies such as Resistive RAM (RRAM) and Magnetoresistive RAM (RRAM). As such, in-memory computing has potential to break down the memory wall, as data movement is not required at all. In contrast to fixed-point quantized networks, binary network is particularly well-suited for the new non-volatile memory technologies, mainly attributed to its simplicity (i.e., 0/1 weights and activations, and multiplier-free operations).

References

[1] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv:1308.3432*, 2013. 5, 8

[2] Michaela Blott, Thomas B. Preußer, Nicholas J. Fraser, Giulio Gambardella, Kenneth O'brien, Yaman Umuroglu, Miriam Leeser, and Kees Vissers. Finn-r: An end-to-end deep-learning framework for fast exploration of quantized neural networks. *ACM Trans. Reconfigurable Technol. Syst.*, 11(3), 2018. 9

[3] Marc Bocquet, Tifenn Hirtzlin, Jacques-Olivier Klein, Etienne Nowak, Elisa Vianello, Jean Michel Portal, and Damien Querlioz. In-memory and error-immune differential RRAM implementation of binarized deep neural networks. *CoRR*, abs/1902.02528, 2019. 9

[4] Olivier Bousquet and André Elisseeff. Stability and generalization. *J. Mach. Learn. Res.*, 2:499–526, March 2002. 4, 6

[5] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems*, 2015. 1, 2

[6] 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:1602.02830*, 2016. 1, 2, 7

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 6

[8] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303–338, 2010. 6

[9] 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. 3

[10] Ariel Gordon, Elad Eban, Ofir Nachum, Bo Chen, Hao Wu, Tien-Ju Yang, and Edward Choi. Morphnet: Fast & simple resource-constrained structure learning of deep networks. In *CVPR*, 2018. 2

[11] Yiwen Guo, Anbang Yao, Hao Zhao, and Yurong Chen. Network sketching: Exploiting binary structure in deep cnns. In *CVPR*, 2017. 2

[12] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*, 2016. 1, 2

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, 2015. 2

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 6

[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. 2

[16] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. 2

[17] Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *ECCV*, 2018. 2

[18] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *CVPR*, 2018. 1

[19] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. 6

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012. 2

[21] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *ICLR*, 2016. 1, 2

[22] Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. In *Advances in Neural Information Processing Systems*, 2017. 1, 3, 6, 7

[23] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *ICLR*, 2019. 2

[24] Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. Reactnet: Towards precise binary neural network with generalized activation functions, 2020. 2, 6

[25] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 722–737, 2018. 2, 6

[26] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *ICCV*, 2017. 1, 2

[27] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *ICLR*, 2019. 2

[28] Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. In *ICLR*, 2017. 2

[29] Brais Martinez, Jing Yang, Adrian Bulat, and Georgios Tzimiropoulos. Training binary neural networks with real-to-binary convolutions. In *ICLR*, 2020. 2, 6

[30] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, 2016. 1, 2, 3, 6, 7

[31] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc Le. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. 2

[32] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):640–651, 2017. 6

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 2, 6

[34] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6105–6114, 2019. 2

[35] Wei Tang, Gang Hua, and Liang Wang. How to train a compact binary neural network with high accuracy? In *AAAI*, 2017. 2

[36] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, 2016. 2

[37] Huan Xu and Shie Mannor. Robustness and generalization. *Mach. Learn.*, 86(3):391–423, 2012. 4, 6

[38] Jianbo Ye, Xin Lu, Zhe Lin, and James Z Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. In *ICLR*, 2018. 1, 2

[39] Shimeng Yu. Neuro-inspired computing with emerging nonvolatile memorys. *Proceedings of the IEEE*, 106:260–285, 2018. 9

[40] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *ECCV*, 2018. 1

[41] 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. 1

[42] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. In *ICLR*, 2017. 1

[43] Shilin Zhu, Xin Dong, and Hao Su. Binary ensemble neural network: More bits per network or more networks per bit? In *CVPR*, 2019. 1, 2, 6, 7

[44] 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. 1, 2, 6, 7, 8

[45] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018. 2