A Survey of Network Binarization in Model Compression and Acceleration

Xianda Xu
University of Electronic Science and Technology of China
xiandaxu@std.uestc.edu.cn

Fumin Shen
University of Electronic Science and Technology of China
fumin.shen@gmail.com

Abstract

Deep Neural Networks have now achieved state-of-the art results in a wide range of tasks including image classification, objection detection and so on. However, they are both computationally intensive and memory intensive, making them difficult to deploy on low-power devices. Emerging solutions to model compression include network pruning, network quantization and so on, among which network binarization is believed to be the most hardware-friendly framework due to its small network size and low computational complexity. In this paper, we provide a comprehensive survey of recent techniques for network binarization. Specifically, we firstly analyze three selected techniques for network binarization and discuss follow-up work based on these techniques. Then, we focus on Binarized Neural Network, providing insightful evaluation of the model and its performance in practice. Finally, we discuss the topic's challenges and possible topics for future research.

1. Introduction

Today, deep neural networks have achieved state-of-the-art results on real-world tasks such as image classification [15], semantic segmentation [22], image captioning [13] and so on. However, energy efficiency becomes the bottleneck for deploying deep neural networks on mobile devices, embedded devices and other devices that are area-and-battery constrained.

Current deep neural network models consist of hundreds of millions of parameters, which brings about two big issues. (1) Energy consumption. One layer can be extracted as follows where X denotes the input matrix, W denotes the parameter matrix, B denotes the offset matrix, σ denotes the activation function and A denotes the activation matrix.

$$A = \sigma(X \cdot W^T + B)$$

We notice that the process of training or inference of the deep neural network model consists large amounts of multiplication in the calculation, which is very energy-consuming. So, small devices with limited battery resources cannot support the calculation of deep neural networks for a long time. (2) Memory cost. [15] provides a deep neural network called AlexNet, which achieved an astonishing performance on ImageNet challenge [8] with 57.1% in top-1 accuracy and 80.2% in top-5 accuracy. AlexNet has around 60 million parameters in total. In the raw model, parameters are stored in the format of float-32, which means that the total memory cost of parameters in AlexNet is around 240MB. Remember that a Xilinx Basys 3 has only 1800 K RAM storage on chip, so on-chip learning seems quite difficult now.

Recent advanced techniques used for model compression and acceleration include network pruning, network quantization, network distillation and network decomposition. (1) Network pruning is a straight forward way to compress the model by removing unimportant parameters and fine-tuning the pruned model. This process can be conducted during the training [2] or by analyzing the significance of each parameter once the raw network has been trained [20]. (2) Network quantization aims at finding efficient representations of each parameter [10]. For instance, in a binarized neural network, weights and activations can be quantified to binary numbers, making the network very efficient but with a limited accuracy. (3) Network distillation [11] tells an idea of training an ensemble of large networks and using their combined output to train a simpler model. (4) Network decomposition is applied in the article [26]. It reconstructs a deep neural network with singular value decomposition. Compression is achieved by removing columns and rows related to the least significant singular values.

All these techniques mentioned above have all shown great performance in model compression and acceleration. However, we think that network binarization, that is, quantifying the model to binary numbers, is one of the most promising techniques in this problem. We give two major reasons.

· One full-precision parameter costs 32 bits while one binary only consumes 1 bit. If all parameters including weights and activations are quantified to -1 or +1, theoretically, the network can be compressed by 32 times.

· If weights are binary, the most multiply-accumulate operations can be replaced by simple accumulations, which is beneficial because as we mentioned above, multipliers are the most space and power-hungry components of the digital implementation of neural networks. If weights and activations are binary, the multiply-accumulation operations can further be replaced by XNOR and bitcount operations.

The rest of the survey is organized as follows: Section 2 takes a review of three selected techniques for network binarization. Section 3 presents our evaluation on one of these three selected techniques called Binarized Neural Network. Section 4 firstly tells related work on Binarized Neural Network and we secondly propose our opinions and possible research prospects.

2. Binarization Techniques

In this section, three selected binarization techniques are analyzed. A discussion follows up, focusing on the comparison of their performance.

2.1. BinaryConnect

BinaryConnect [6] is a model proposed in the very early stage of the study in network binarization.

In the BinaryConnect network, the weights are binary values. Full-precision weights are quantified to -1 or +1 with a method called Stochastic Binarization,

$$w_b = \begin{cases} +1 & \text{with probability p} = \sigma(w) \\ -1 & \text{with probability 1-p} \end{cases}$$

where $\sigma()$ is the "hard sigmoid" function:

$$\sigma(x) = \text{clip}(\frac{x+1}{2}, 0, 1) = \max(0, \min(\frac{x+1}{2}))$$

Remember that multiplying any value with +1 or -1 is equivalent to a NOR operation. Hence, the full-precision multiplication can be decomposed into NOR operations and accumulations.

One thing that needs noticing is that during both forward propagation and backward propagation, all weights are binary, but in parameter updating process, all weights are fully-precise to keep its accuracy at a high level.

One benefit of BinaryConnect is that like dropout[24], it adds noise to the model, which can prevent the network from over-fitting.

2.2. Binarized Neural Network

As far as we know, Binarized Neural Network [7] is the first network to quantify both weights and activations to binary numbers. Instead of using Stochastic Binarization which is more reasonable but more time-consuming, it uses Deterministic Binarization,

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

When both weights and activations are binarized, the convolution operation can be simplified as XNOR and bitcount operations, which boosts speed greatly. However, two problems need to be solved before Binarized Neural Network can be put into practice.

The first problem is caused by using Deterministic Binarization. Remember that the process of backward propagation in each layer can be extracted as follows:

The gradients of the loss function L with respect to I_l, the input of the l layer are,

$$g_l = \frac{\partial L}{\partial I_l}$$

Then the layer activation gradients are,

$$g_{l-1} = g_l W_l^T$$

Then the layer weight gradients are,

$$g_{w_l} = g_l I_{l-1}^T$$

Since the gradient of $Sign(x)$ is always zero, the gradients in the backward propagation are mostly zero.

[3] provides a smart idea of the straight-through estimator designed for discrete network. So the process of backward propagation in Binarized Neural Network is as follows,

The gradients of the loss function C with respect to a_L, the input of the last layer L are calculated as usual,

$$g_{a_l} = \frac{\partial C}{\partial a_L}$$

For the k^{th} hidden layer (mapping $Sign(x)$), firstly go through the straight-through estimator,

$$g_{a_k} = g_{a_k^b} \cdot 1_{|a_k| \leq 1}$$

Then through the back batch normalization,

$$g_{s_k} = BN(g_{a_k})$$

Then the layer activation gradients are,

$$g_{a_{k-1}^b} = g_{s_k} W_k^b$$

Then the layer weight gradients are,

$$g_{W_k^b} = g_{s_k}^T a_{k-1}^b$$

In the backward propagation, the straight-through estimator is used for gradient calculation. Accordingly, $Htanh(x)$, the gradient of which is the straight-through estimator, is used as the activation function in the forward propagation,

$$Htanh(x) = Clip(x, -1, 1)$$

The second problem lies in the accuracy loss because of binarization. To maintain accuracy in the network, as shown in Figure 1, the input of the network is non-binarized and the output is binarized only in the training. During backward propagation, gradients are not binarized until the end when they are clipped to update the weights in each layer.

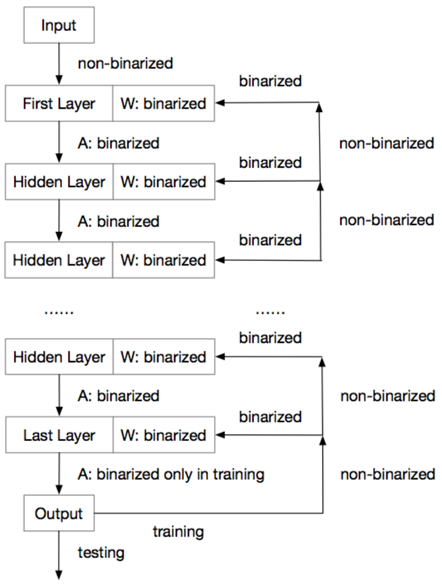

Figure 1. The use of binarization in Binarized Neural Network

Based on the idea of binarization, Binarized Neural Network furthers BinaryConnect by binarizing the activations in addition to the weights in the network, accelerating the model to a higher level.

2.3. XNOR-Net

Like Binarized Neural Network, XNOR-Net [23] also binarizes both weights and activations in the network. However, to ensure a better accuracy rate, it introduces a filter of scaling factors with full precision in each convolutional layer. It approximates a convolution in a way illustrated in Figure 2.

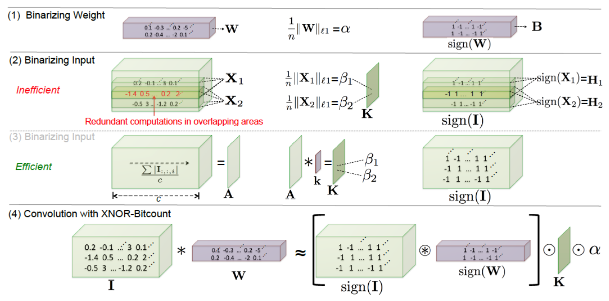

Figure 2. The approximation of convolution in XNOR-Net

Let us consider a convolutional layer where the input $I \in \mathbb{R}^{c \times w_{in} \times h_{in}}$, the weight filter $W \in \mathbb{R}^{c \times w \times h}$ and a matrix $A = \frac{\sum |I_{:,:,i}|}{c} \in \mathbb{R}^{w_{in} \times h_{in}}$.

Suppose α is the scaling factor for the weight and $K \in \mathbb{R}^{w_{out} \times h_{out}}$ is the matrix containing scaling factors for all sub-tensors in the input. Then, $\alpha = \frac{1}{n} \parallel W \parallel$ and $K = A * k$ (where $k \in \mathbb{R}^{w \times h}$ is a filter).

So, the convolution in XNOR-Net can be approximated by mainly using binary operations:

$$I * W \approx (sign(I) \star sign(W)) \odot K \odot \alpha$$

where $\star$ indicates a convolutional operation using XNOR and bitcount operations.

XNOR-Net furthers Binarized Neural Network by adding a filter of scaling factors with full precision in each convolutional layer, in which case the accuracy rate can meet the requirement of practical applications. However, both additional fully-precise scaling factors and non-binary convolutional operations introduced to the network make XNOR-Net not a strictly binarized network and burdened with more memory cost and computing cost compared with Binarized Neural Network.

2.4. Comparison

Having gone through the three techniques discussed above that are commonly used in network binarization, we make a comparison among them based on the classification accuracy and the resource consumption of multiplication, which is illustrated in Table 1.

As for comparison on classification accuracy, all of the three models are built in a structure of AlexNet [15]. The results are aggregated from experiments in [27] and [23].

As for comparison on resource consumption of multiplication, a $W_{10 \times 10} \times A_{10 \times 10}$ matrix multiplication on a Xilinx Virtex-7 FPGA board is the testbench. The results are aggregated from experiments in [18].

	Classification Accuracy		**Resource Consum. of Mul.**	
Methods	Top-1	Top-5	DSP	LUTS
AlexNet	56.6	80.2	\	\
BinaryConnect	35.4	61.0	0	10
BNN	27.9	50.4	0	0.5
XNOR-Net	44.2	69.2	0	5

Table 1. The performance comparison among the three models

It is shown in the table that XNOR-Net achieves the best performance in classification accuracy among the three models. It is understandable since it introduces scaling factors with full precision to the network, which in turn brings about both memory cost and computation cost. In order to run a deep neural network successfully on a small device, reducing resource consumption is the priority. Though Binarized Neural Network does not have an accuracy rate that is good enough to satisfy the requirement of real-life applications, it has the most potential among the three because of its extremely low resource consumption. That is why more and more researchers are looking at the Binarized Neural Network. In the following section, we will talk about our evaluation of Binarized Neural Network.

3. Evaluation

In the second section, we surveyed three different techniques for network binarization and tell the reason why we see a research prospect in Binarized Neural Network. In this section, we talk about the experimental evaluation of Binarized Neural Network.

3.1. Experimental methods

We train two implementations of Binarized Neural Network in our experiments. The first model is built with a LeNet [17] structure fed by the popular benchmark dataset MNIST (28 x 28 x 1) [16]. The second model is built with an adapted AlexNet [15] structure fed by the popular benchmark dataset CIFAR-10 (32 x 32 x 3) [14]. Moreover, our training method uses a batch size of 64 in Binarized LeNet and 128 in Binarized Adapted-AlexNet. An overview of the training and testing information is shown in Table 2.

Dataset	MNIST	CIFAR-10
Dimension	28 x 28 x 1	32 x 32 x 3
Labels	10	10
Training set	5K	40K
Validation set	1K	10K
Testing set	1K	10K
Training epochs	100	200

Table 2. The overview of the training and testing information

The commonly used convolutional block in Adapted-AlexNet, as shown in Figure 3, consists of two convolutional layers where both weights and activations are binarized, two batch normalization layers, two activation layers which use $Htanh(x)$ as their activation function and one maxpooling layer. Netscope is a tool for visualizing neural network architectures. The architecture of Binarized LeNet can be seen here. The architecture of Binarized Adapted-AlexNet can be seen here.

3.2. Experimental results

The experiments are conducted using Tensorflow [1], a widely used machine learning system. The code is uploaded to GitHub [1].

As illustrated in Table 3, because of network binarization, the model size is greatly reduced. However, the consequence is, the accuracy rate of image classification decreases by 6% in LeNet on MNIST and by 8% in Adapted-AlexNet on CIFAR-10.

[1]https://github.com/brycexu/BinarizedNeuralNetwork

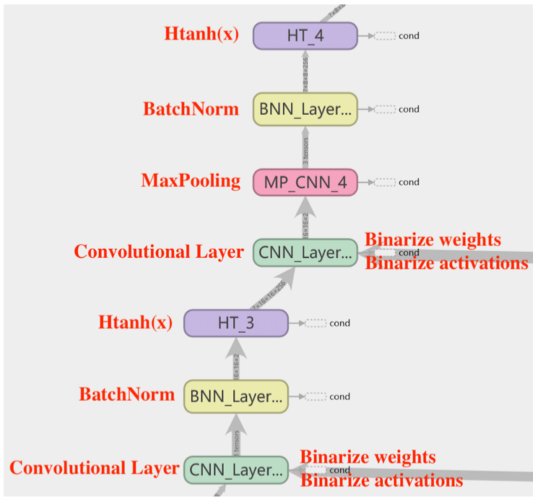

Figure 3. The convolutional block in Adapted-AlexNet

Experiment 1		
Network	LeNet	
Binarized	No	Yes
Dataset	MNIST	
Accuracy	98%	92%
Parameters	3.22M	3.22M
Model Size	12.90MB	0.40MB
Experiment 2		
Network	Adapted-AlexNet	
Binarized	No	Yes
Dataset	CIFAR-10	
Accuracy	92%	84%
Parameters	14.02M	14.02M
Model Size	56.10MB	1.75MB

Table 3. The results of two experiments

It is obvious that the accuracy rate in both binarized models cannot satisfy the requirements of real-life applications including image classification. As a matter of fact, how to increase the accuracy rate of a binarized model and improve its stability is now one of the biggest issues that researchers are looking at.

In the following section, we will have a discussion on Binarized Neural Network based on the low-accuracy problem reflected in our experiments.

4. Discussion

In this section, we analyze two issues lying in Binarized Neural Network that lead to the low-accuracy problem. Then, we present related work that has put forward possible solutions to these flaws in the model. Finally, we provide our thoughts and possible topics for future research on model compression and acceleration using network binarization.

4.1. Two issues

Researchers have been systematically surveying the binarized deep neural network model. As discussed in [28], two major issues account for the low-accuracy problem in the binarized model.

The first issue is the robustness issue. Because of network binarization, one binarized model has larger output changes which makes it more susceptible to input perturbation. Experimental results in [28] show that binarized models suffer more from overfitting effects and that having more bits at activations can improve their robustness significantly.

The second issue is the stability issue. It is known that the quantified model is more difficult to optimize because of the gradient mismatch [19]. The gradient mismatch tells that because of the non-smoothness of the network architecture, the effective activation function in a fixed point network (a) is actually a non-differentiable function in a discrete point network (b), as shown in Figure 4. In [5], researchers find that the gradient mismatch starts to impact the stability of SGD [4] when the bit-widths become too small. The gradient mismatch also exacerbates as the error signal propagates deeper down the network.

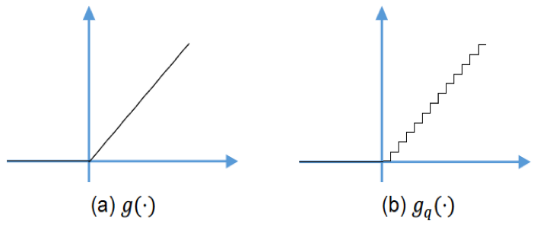

Figure 4. The illustration of the gradient mismatch

4.2. Related work

Recent work on network binarization majorly focuses on dealing with two issues mentioned above, namely the robustness issue and the stability issue.

As for the robustness issue, one obvious way is to have more bits per weight and per activation value. Some researchers believe that network binarization is too extreme and adding more bits can ensure a satisfactory accuracy rate. The ternary model in [27] values weights and activations with $\{-1, 0, +1\}$. It is further found that activations play a more significant role in the model's robustness than weights do. In [21], researchers believe that a quantified model with a combination of 2-bit weights and 4-bit activations is the best-of-breed choice that achieves a balance between efficiency and accuracy. In [28], researchers change the thought of more bits per network to the idea of more networks per bit so they propose a network that leverages ensemble methods.

In [25], researchers provide two useful ways to deal with the robustness issue. Research shows that higher learning rate causes turbulence inside the binarized model so they believe that the binarized model needs finer tuning and a small learning rate is a better choice. They also develop a binarized-model-oriented regularization that can help keep the model away from overfitting.

As for the stability issue, $Htanh(x)$ is used as the activation function in Binarized Neural Network. To solve the problem that as the error signal goes deeper down, the gradient mismatch exacerbates, [19] iteratively binarizes the model, as shown in Figure 5.

	Phase 1		Phase 2		Phase 3	
	Acts	Wgts	Acts	Wgts	Acts	Wgts
Layer4	Float	-	Float	-	Float	update
Layer3	Float	-	Float	update	FixPt	-
Layer2	Float	update	FixPt	-	FixPt	-
Layer1	FixPt	-	FixPt	-	FixPt	-

Figure 5. The idea of iterative binarization in the model

One possible misconception about binarized models is that its resource consumption is optimum. However, evidence has shown that they can be further improved in resource consumption.

Researchers in [18] explore the redundancy in Binarized Neural Network and built a Compact Binarized Neural Network based on sensitivity analysis and data pruning, as shown in Figure 6. The Binarized Neural Network is pretrained as usual. Then, N^{th} bit slices are substituted with binary random bit slices. If the error brought is acceptable, the N^{th} bit slices are classified as prunable. Therefore, a collection of insensitive bit slices can be pruned. Results show that in this way, the Binarized Neural Network can be further compressed while limited accuracy rate is lost. Besides, researchers in [12] quantized the error gradients in the backward propagation and improves the execution efficiency greatly.

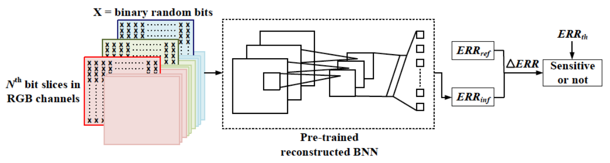

Figure 6. The idea of Compact Binarized Neural Network

Remember that during backward propagation in training, Binarized Neural Network stills requires full-precise values to update its weights. Researchers in [9] greatly improve storage efficiency by recursively recycling data storage of weights during training, which makes on-chip learning a reality. The idea can be illustrated in Figure 7 where a $1 \times 2 \times 1$ network is trained conventionally at first, followed by all weights being binarized to one bit. Afterwards, the second iteration of training is continued with the (n-1)-bit storage space for new weights. This process is repeated until the end of the training. Experimental results show that it can achieve a 2.28 $bit/weight$ storage requirement for training, which makes on-chip learning a reality.

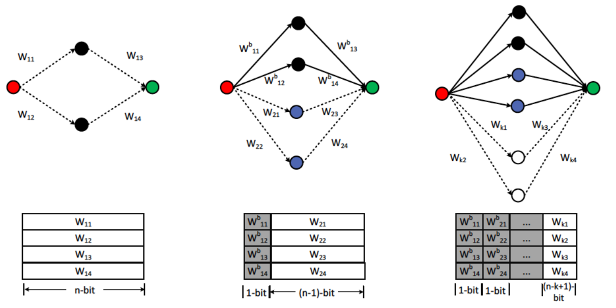

Figure 7. The idea of recycling data storage of weights recursively

4.3. Future work

In this survey, we firstly analyze three techniques for network binarization. BinaryConnect in [6] binarizes weights to $\{-1, +1\}$ with stochastic binarization. Binarized Neural Network in [7] binarizes both weights and activations to $\{-1, +1\}$ with deterministic binarization. XNOR-Net in [23] furthers by adding scaling factors to each convolutional layer. Majorly considering resource consumption, we select Binarized Neural Network and our experiments show that there is a big accuracy rate drop in the model which is also understandable. Then we discuss two major issues that binarized models are facing, namely the robustness issue and the stability issue, followed with related work.

Network binarization is an "extreme" way to compress and accelerate the deep neural network. So, how to narrow the gap of the accuracy rate between original networks and binarized networks is what many researchers are studying on, focusing on two major issues. Here, for our future work,

- We insist that no value with full-precision should be introduced to the network. In forward propagation, both weights and activations should be binary for computational acceleration and storage efficiency. In backward propagation, gradients can be quantized to update the weights, which can save much space and enable learning on some small devices.

- We propose that in order to remedy the accuracy loss due to network binarization, more weights can be introduced to the new network. This idea is inspired by [21] which adds more filters to the quantized network and makes it wider to improve the accuracy rate. As far as we know, few work has been conducted on enlarging or widening the structure of binarized networks to improve the accuracy.

- We believe that redundancy in the binarized network should also be removed as much as possible. When we change the structure of the network and add more weights, the network becomes more bloated. In this case, network pruning techniques like [18] can be used to make the model more efficient.

Theoretically, by using network binarization, deep neural networks can be compressed by 32 times and accelerated by replacing all multiply-accumulation operations with XNOR and bitcount operations. However, in practice, the network can not be compressed by exactly 32 times since we need to remedy the accuracy loss usually by changing some of the weights from 1-bit precision to more-bit precision. We speculate that adding more 1-bit-precise weights to the network might be better than adding more precision to some of weights in the network based on two assumptions.

- The first assumption is that with a same memory size, the network added with 1-bit weights achieves a higher accuracy rate of image classification and a lower resource consumption in computation than the network where some weights are more-bit-precise.

- The second assumption is that with a same memory size, it removes more redundancy in the binarized network added with 1-bit weights than in the binarized network where some weights are more-bit-precise.

Therefore, in our future work, we are trying to figure out a general way of transforming one deep neural network to a binarized network with a few weights added but without changing the original structure much. Then, we would like to test our hypotheses above and evaluate this technique in model compression and acceleration by looking at its performance carefully.

Acknowledgement

The first author would like to thank Fumin Shen for his advice on the topic selection and his help to conduct the survey. The first author would also like to thank Marco Pedersoli for his offering an internship to collaborate on this topic for future work.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] J. M. Alvarez and M. Salzmann. Learning the number of neurons in deep networks. In *Advances in Neural Information Processing Systems*, pages 2270–2278, 2016.

[3] Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[4] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[5] Z. Cai, X. He, J. Sun, and N. Vasconcelos. Deep learning with low precision by half-wave gaussian quantization. *arXiv preprint arXiv:1702.00953*, 2017.

[6] M. Courbariaux, Y. Bengio, and J.-P. David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

[7] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009.

[9] T. Guan, X. Zeng, and M. Seok. Recursive binary neural network learning model with 2.28 b/weight storage requirement. *arXiv preprint arXiv:1709.05306*, 2017.

[10] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[11] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[12] I. Hubara, E. Hoffer, and D. Soudry. Quantized back-propagation: Training binarized neural networks with quantized gradients. 2018.

[13] X. Jia, E. Gavves, B. Fernando, and T. Tuytelaars. Guiding the long-short term memory model for image caption generation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2407–2415, 2015.

[14] A. Krizhevsky, V. Nair, and G. Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 2014.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[16] Y. LeCun, C. Cortes, and C. Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

[17] Y. LeCun et al. Lenet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet*, page 20, 2015.

[18] Y. Li and F. Ren. Build a compact binary neural network through bit-level sensitivity and data pruning. *arXiv preprint arXiv:1802.00904*, 2018.

[19] D. D. Lin and S. S. Talathi. Overcoming challenges in fixed point training of deep convolutional networks. *arXiv preprint arXiv:1607.02241*, 2016.

[20] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky. Sparse convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 806–814, 2015.

[21] A. Mishra, E. Nurvitadhi, J. J. Cook, and D. Marr. Wrpn: wide reduced-precision networks. *arXiv preprint arXiv:1709.01134*, 2017.

[22] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[23] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

[24] N. Srivastava. Improving neural networks with dropout. *University of Toronto*, 182:566, 2013.

[25] W. Tang, G. Hua, and L. Wang. How to train a compact binary neural network with high accuracy? In *AAAI*, pages 2625–2631, 2017.

[26] J. Xue, J. Li, and Y. Gong. Restructuring of deep neural network acoustic models with singular value decomposition. In *Interspeech*, pages 2365–2369, 2013.

[27] C. Zhu, S. Han, H. Mao, and W. J. Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.

[28] S. Zhu, X. Dong, and H. Su. Binary ensemble neural network: More bits per network or more networks per bit? *arXiv preprint arXiv:1806.07550*, 2018.