An adiabatic method to train binarized artificial neural networks

Yuansheng Zhao\(^1,2\) & Jiang Xiao\(^1,3,4,\*)

An artificial neural network consists of neurons and synapses. Neuron gives output based on its input according to non-linear activation functions such as the Sigmoid, Hyperbolic Tangent (Tanh), or Rectified Linear Unit (ReLU) functions, etc.. Synapses connect the neuron outputs to their inputs with tunable real-valued weights. The most resource-demanding operations in realizing such neural networks are the multiplication and accumulate (MAC) operations that compute the dot product between real-valued outputs from neurons and the synapses weights. The efficiency of neural networks can be drastically enhanced if the neuron outputs and/or the weights can be trained to take binary values \(\pm 1\) only, for which the MAC can be replaced by the simple XNOR operations.

In this paper, we demonstrate an adiabatic training method that can binarize the fully-connected neural networks and the convolutional neural networks without modifying the network structure and size. This adiabatic training method only requires very minimal changes in training algorithms, and is tested in the following four tasks: the recognition of hand-writing numbers using a usual fully-connected network, the cat-dog recognition and the audio recognition using convolutional neural networks, the image recognition with 10 classes (CIFAR-10) using ResNet-20 and VGG-Small networks. In all tasks, the performance of the binary neural networks trained by the adiabatic method are almost identical to the networks trained using the conventional ReLU or Sigmoid activations with real-valued activations and weights. This adiabatic method can be easily applied to binarize different types of networks, and will increase the computational efficiency considerably and greatly simplify the deployment of neural networks.

Artificial neural networks have dramatically advanced the capability of artificial intelligence in many aspects\(^3,5\). In artificial neural networks, the most widely used neuron activation functions include the Sigmoid function $sgmd(x) = 1/[1 + \exp(-x)]$, the Hyperbolic Tangent function $\tanh(x)$, and the Rectified Linear Unit function $\text{ReLU}(x) = \max(x, 0)$, etc.\(^6,7\). All of these activation functions require multiple bits for storage or processing. In contrast, the binary Heaviside step-function, as the simplest nonlinear activation,

$$H(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases},$$

only takes 1 bit for storing the output. However, the Heaviside function is not suitable for the activation function because of its non-differentiability at $x = 0$ and its vanishing derivatives for $x \neq 0$, which makes it impossible to use the back-propagation algorithm\(^8\) in the training process. For the same reason, the weights have to be smoothly tunable and cannot take binary values.

Despite of these difficulties, binarization of neural networks is highly desirable. Modern large Deep Neural Networks (DNNs) requires very large memory (hundreds of MB) to store weights and intermediate variables, as well as strong computing power (up to $\sim 10$ G fops per sample). This is one of the major obstacles limiting the applications of artificial neural networks, especially on mobile devices. However, binarizing weights/activations obviously can provide huge performance boost, thus it is of interest to find ways to achieve binary weights/activations.

Previously, it was believed that the network with binary activation and/or binary weights cannot be trained because good binary weights simply do not exist as the result of inevitable loss of degrees of freedom. However, recently, some pioneering works such as the BinaryConnect\(^9\) and BNN\(^10\) proposed by Courbariaux et al. demonstrated that binarization of weights and activation in deep neural network is indeed feasible. During training

\(^1\)Department of Physics and State Key Laboratory of Surface Physics, Fudan University, Shanghai 200433, China. \(^2\)Department of Physics, University of Tokyo, Tokyo, Japan. \(^3\)Institute for Nanoelectronics Devices and Quantum Computing, Fudan University, Shanghai 200433, China. \(^4\)Shanghai Qi Zhi Institute, 200232 Shanghai, China. \(^\ast\)email: xiaojiang@fudan.edu.cn
become binarized because weights are identical. But when the width of the raw weights $W$ for each layer, and the polarized weights instead of raw weights are used for the connections between neurons, i.e. the binary validation accuracy ($\text{Scheme V2}$) reaches target accuracy or the training epochs exceeds 15–30 epochs (exact number depending on tasks), these methods use binarized weights (deterministically or stochastically) and/or activations (deterministically) for forward pass; for backward pass, Straight-Through Estimator (STE) is used for gradient of activations:

$$\frac{d \text{sign}(x)}{dx} = \begin{cases} 1 & |x| < 1 \\ 0 & \text{otherwise} \end{cases}$$

and the gradient of binarized weights are applied to raw (un-binarized) weights. These methods can achieve good results of various tasks. Almost all the works on binarization (quantization) relies on this technique, and some improved variants or extensions are proposed, such as XNOR-Nets$^{11}$ that adds a scaling factor to cancel out quantization error, LQ-Nets$^{12}$ that automatically optimizes quantizers, DSQ$^{13}$ which improves STE itself, and DoReFa-Net$^{14}$ which also uses low bit width during backpropagation to reduce training time. Other methods for binarizing (quantizing) weights of neural networks include expectation backpropagation$^{15}$ and Proxquant$^{16}$. Simons and Lee$^{17}$ and Qin et al.$^{18}$ provide reviews of recent progress in binary neural networks.

Apart from binarization or quantization like our work, there are some other efforts to speed up deep neural networks, they use different method from binarization/quantization of networks, but the aim is essentially the same—to boost the performance of neural network. Some examples are using compact layers$^{19–21}$ and compressing network by removing redundant weights$^{22–25}$.

**Method**

Here, we describe an efficient yet simple method to train existing networks to take binary activation function and/or binary weights. In this method, we use a parameterized Sigmoid and Hyperbolic Tanh functions with width $w$:

$$\text{sgmd}_w(x) \equiv \left[1 + \exp(-x/w)\right]^{-1} \quad \text{and} \quad \tanh_w(x) \equiv \tanh(x/w)$$

(2)

to approach the Heaviside step-function $H(x)$ and the sign-function $\text{sign}(x)$ as $w \to 0$. In order to train the real-valued weights to work with Heaviside-activated neurons, the training process starts with the usual Sigmoid function of finite width $w > 0$, and then $w$ is decreased adiabatically until the Sigmoid function becomes a Heaviside-like function. In the end, the network may be trained once more using the Heaviside-function (binary) activations to further stabilize the weights. To achieve binarized weights, the network is slightly modified by replacing the raw weights $W$ with the polarized weights: $W \mapsto a \tanh_w(W)$, where $a$ is a real-valued constant for each layer, and the polarized weights instead of raw weights are used for the connections between neurons. The raw weights $W$ and multiplier $a$ are trained as usual. When width $w$ is large, the raw weights and polarized weights are identical. But when the width $w$ is gradually decreased during the training, the polarized weights become binarized because $a \tanh_{w\rightarrow 0}(W) = \pm a$ as $w \rightarrow 0$. Unlike most of previous works, our method does not require STE to work and may get rid some of its limitations such as gradient mismatch. The parameterized function used in Eq. (2) has some similarity with the Differentiable Soft Quantization (DSQ) method proposed by Gong et al.$^{13}$, but the tunable width used in this work is viewed as a global non-trainable, time-evolving parameter; Proxquant by Bai et al.$^{16}$ used a time-evolving regularizer to binarize weight, but is unable to regularize activation in a similar manner.

For small tasks such as hand written number below, the width $w$ can be adjusted manually. For other larger tasks, we adopted one of the following two self-adaptive methods to adjust $w$. $\text{Scheme V1}$: a target accuracy is set for each $w$. The network is trained using the current value of $w$. Once the validation accuracy (using the present $w$) reaches target accuracy or the training epochs exceeds 15–30 epochs (exact number depending on tasks), the value of $w$ is decreased by a factor of $1.2–2$ (depending on tasks, but not crucial) and the learning rate is also reduced by a factor of $1–2$. In the mean time, the next target accuracy is set to the present accuracy. $\text{Scheme V2}$: there is no target accuracy in this scheme. The width $w$ and the learning rate are decreased as in Scheme $V1$ once the binary validation accuracy ($i.e., a = w = 0$ instead of at current width) saturates. Note that this method does not require pre-training which is required for $V1$ and may be viewed as more flexible version of $V1$.

**Numerical experiments**

We use the adiabatic training method described above to binarize the following four different tasks based on some typical neural network structures: (1) a fully-connected network for recognizing hand-written numbers; (2) a convolutional neural network for recognizing the dog-cat pictures; (3) a convolutional neural network for recognizing spoken numbers; (4) a ResNet-20 or VGG-Small neural network for recognizing pictures from CIFAR-10 dataset with 10 classes. We compare the networks obtained from four different supervised training procedures: the conventional non-binary network trained with standard Sigmoid or ReLU activations and real-valued weights, the binary-activation network trained with Heaviside (binary) activation and real-valued weights, the binary-weight network trained with Sigmoid or ReLU activation and binary weights, and the full-binary network trained with binary activation and binary weights. All networks are built using the TensorFlow module$^{26}$.

**Image recognition of hand-written numbers.** A fully-connected network with one hidden layer (see Fig. 1A) is sufficient for this task$^{27}$. 70k image samples from the MNIST dataset$^{28}$ are used with 60k for training/validating and 10k for testing. drop-out$^{29,30}$ is used to alleviate over-fitting, and no data augmentation is used. For training the non-binary network, the neurons in the hidden layer use the ReLU activation function and 98.2% accuracy can be reached. Since the ReLU function cannot go smoothly to a Heaviside step-function, to train binary networks, we use a hybrid Sigmoid-ReLU activation function as

$$f_w(x) = 2\text{sgmd} \left[\text{ReLU} \left(\frac{x}{w}\right)\right] - 1,$$

(3)
which can approach the Heaviside function as $w \to 0$. The width $w$ is adjusted manually for this task. The same fully-connected network (of the same size and structure) is trained for 8 epochs using this hybrid activation with $w = 1/3$ for neurons in the hidden layer, followed by 3 epochs using $w = 1/12$ and 2 epochs with the Heaviside activation. The realized binary-activation network with Heaviside activation has the testing accuracy 97.1% as shown in Fig. 1B, similar to the conventional non-binary network using ReLU activation. It should be noted that the initial epochs of training with finite $w$ is crucial, otherwise the validation accuracy can only reach $\sim 85\%$ if using Heaviside activation from start. For the binary-weight network, the network is trained for 8 epochs using the polarized weight with $w = 1/3$, followed by 3 epochs each for $w = 1/10, 1/20, 1/50, 1/100, 1/300, 1/500$ and 0, and finally the realized binary-weight network reaches 97.2% testing accuracy as well. The exact sequence of $w$ is not important as long as it decreases. For the full-binary network, the network is trained with $w$ weights $\propto w$ activation, and $w$ weights is decreased in the same way as in training the binary-weight network. The final validation accuracy of 96.0% is reached. Therefore, both the half-binary and full-binary networks can reach almost the same validation accuracy as the non-binary network.

Figure 1C shows the distributions of the activation outputs and the weights from the non-binary and full-binary networks. It shows that for the full-binary networks, both the activation outputs and the weights become progressively more and more binarized, and eventually purely binarized at the end of the training process. In comparison, both activations and weights in the non-binary network spread out across the full range for all epochs as expected. In the full-binary case, the weight scaling factor $a$ also needs to be trained and is in general a real-valued number, but at the end of the training, $a$ can be set to be unity and all activations and/or weights become purely binary. We should mention that it is not desirable to have many weights or activation values close to the center (near zero) of the Sigmoid/Tanh function during training, because the center part cannot be accessed in binarized networks. Therefore, the ratio between the range of the initial weights $\text{Range}(|W|)$ and the initial value for width $w$ should be $\text{Range}(|W|)/w \gtrsim 1$ (for the first band of Fig. 1C for the full-binary panel, $\text{Range}(|W|) \simeq 1/2$ and $w = 1/3$), and in the meantime the learning rate should be moderately large to avoid the center part.

**The dog-cat picture recognition.** A convolutional neural network (CNN) with three hidden layers (see Fig. 2A) are used for this task. In this network, the convolution kernel is $3 \times 3$ and the pooling size is $2 \times 2$. And 25 k pictures of dogs and cats from the Kaggle dataset\(^a\) (subset of Asirra\(^b\)) are used with 23.4 k for training/validating and 2.6 k for testing. The pictures are resized to 64 $\times$ 64 pixels and converted to grayscale before
training, the training pictures are randomly flipped horizontally and shifted by a maximum of 8 pixels for data augmentation.

For the conventional non-binary network, the standard ReLU activations with max-pooling are used in all hidden layers, and batch normalization is used to accelerate training. Within 120 epochs of training, the validation accuracy saturates to 89.7% as shown in Fig. 2B. To train a binary-activation network with the Heaviside activation, the ReLU activation is replaced by the parametrized sigmoid activation $\text{Eq. } (2)$ with varying width $w$ for the hidden layers except last one which uses the ReLU-sigmoid hybrid activation (using sigmoid here causes mostly 0 output when binarized thus not ideal). Because the max-pooling makes no sense with numbers of 0s and 1s, the order of pooling and activation is changed as convolution $\rightarrow$ batch Normalization $\rightarrow$ pooling $\rightarrow$ activation, the same as previously the width is adjusted by hand, but this is not practical for large network. From this task, we use the self-adaptive method V1 mentioned above. With these operations, the realized binary-activation network equipped with Heaviside activation functions reaches a validation accuracy of 84.6% (see the orange curve in Fig. 2B). For a binary-weight network, we use the same method to reduce the width $w$ and a validation accuracy of 86.0% is achieved (see the green curve Fig. 2B). In order to realize the full binarization for both activations and weights, we have to leave out the first (weights only) and last layer from being binarized, as in most other methods on binarization. The width $w$ for the activations and weights can be reduced to zero alternatively (or simultaneously) to achieve full binarization, and realized a validation accuracy of 85.5% (the red curve in Fig. 2B), close to the non-binary and half-binary networks.

**Voice recognition of spoken numbers.** This task uses a similar convolutional neural network (see Fig. 2A) as the one used for the dog-cat recognition task above. The kernel length is 30 and the pool sizes for the three layers are 10, 8, and 5, respectively. And 14 k audio waveforms of human spoken numbers (each resized to array of length 8000) from Speech Commands dataset are used with 13 k for training/validating and 1 k for testing; no data augmentation is used for this task. The non-binary network trained using the ReLU activation and real-valued weights along with max-pooling reaches a validation accuracy of 93.7% (see the blue curve in Fig. 2C). In order to realize the full binarization for both activations and weights, we have to leave out the first (weights only) and last layer from being binarized, as in most other methods on binarization like. The width $w$ for the activations and weights can be reduced to zero alternatively (or simultaneously) to achieve full binarization, and realized a validation accuracy of 85.5% (the red curve in Fig. 2C), close to the non-binary and half-binary networks.
In this task, we train a standard ResNet-20\cite{he2016deep} or VGG-Small network\cite{simonyan2014very} (with similar structure as the CNN shown in Fig. 2A) to recognize 60 K (50 K for training/validation and 10 K for testing) 32 × 32 color images belonging to 10 classes from the CIFAR-10 dataset\cite{krizhevsky2009learning,cifar}. This task is much more challenging than previous ones. However, the adiabatic method can be applied directly without modification, except that the first and last layer are kept un-binarized for both half- and full-binarization. For the data augmentation, as in\cite{ulyanov2018deep}, we randomly flip the picture horizontally and shift the image by the maximum of 4 pixels.

Using the adiabatic method and starting from pre-trained full-precision network with ReLU activations, we succeed in training both ResNet-20 and VGG-Small network to function with binary-weight as shown in Fig. 2D, where the validation accuracy of binary-weight network approaches to the accuracy of the full-precision network. The test accuracy reached is 90.2% and 93.3% respectively. Compared with VGG-Small, ResNet-20 is much deeper but with much less feature maps per layer, thus the binarization may cause more information loss. It is known to be very hard to binarize activations. As keeping accuracy while pushing activations to binary value is practically impossible, scheme V2 is used to adjust the width $w$. We successfully train the binary activation and full binary ResNet-20 to testing accuracy 85.7% and 83.0%. A slight modification in network structure by adding shortcut at every conventional layer\cite{he2016identity} can increase the accuracy to 86.2% and 84.1%. On the other hand, VGG-Small networks are more friendly to activation binarization, and binary-activation can be trained using the adiabatic method (adjusting width using method V1) starting with $w = 1$ (or pre-training with sigmoid activation) and reached 92.4%. The full-binary VGG-Small networks (see the right panel of Fig. 2D) can also be trained using the same method with accuracy 90.7%.

| Tasks | Networks | Method | Binarization | $w$-Reduction | Accuracy (%) |
|-------|----------|--------|--------------|---------------|--------------|
| MNIST | 2-layer DNN | Full precision | Manual | 98.2 (0.1) |
|       | Ours | Weights only | Manual | 97.2 (0.1) |
|       | Ours | Activation only | Manual | 97.1 (0.1) |
|       | Ours | Weights and activations | Manual | 96.0 (0.2) |
| Dogs vs Cats | 4-layer CNN | Full precision | V1 | 93.7 (0.2) |
|       | Ours | Weights only | V1 | 91.5 (0.5) |
|       | Ours | Activation only | V1 | 91.4 (0.8) |
|       | Ours | Weights and activations | V1 | 93.0 (0.2) |
| Spoken numbers | 4-layer CNN | Full precision | V2 | 86.2 (0.3) |
|       | Ours | Activation only | V2 | 84.1 (0.2) |
|       | Ours | Weights and activations | V2 | 84.1 (0.2) |
|       | Ours | Weights and activations | V1 | 90.7 |
| ResNet-20 | 4-layer CNN | Full precision | V1 | 92.4 (0.1) |
|       | DoReFa\cite{zhou2016dorefa} | Weights only | V1 | 90.0 |
|       | LQ-Net\cite{lebedev2016learning} | Weights only | V1 | 90.1 |
|       | DSQ\cite{watanabe2017dsq} | Weights only | V1 | 90.2 |
|       | ProxQuant\cite{kim2016proxquant} | Weights only | V1 | 90.7 |
|       | Ours | Activation only | V2 | 86.2 (0.3) |
|       | Ours | Weights and activations | V2 | 84.1 (0.2) |
|       | DoReFa\cite{zhou2016dorefa} | Weights and activations | V1 | 79.9 |
|       | DSQ\cite{watanabe2017dsq} | Weights and activations | V1 | 84.1 |
|       | CL-BCNN\cite{wu2018ternary} | Weights and activations | V1 | 91.1 |
| VGG-Small | 4-layer CNN | Full precision | V1 | 93.8 (0.1) |
|       | Ours | Weights only | V1 | 93.3 (0.1) |
|       | BWN\cite{li2017binary} | Weights only | V1 | 90.1 |
|       | BinaryConnect\cite{rastegari2016xnor} | Weights only | V1 | 91.7 |
|       | LQ-Net\cite{lebedev2016learning} | Weights only | V1 | 93.5 |
|       | Ours | Activation only | V1 | 92.4 (0.2) |
|       | Ours | Weights and activations | V1 | 90.7 (0.2) |
|       | XNOR-Net\cite{wu2018ternary} | Weights and activations | V1 | 89.8 |
|       | BNN\cite{courbariaux2016binaryconnect} | Weights and activations | V1 | 89.9 |
|       | DSQ\cite{watanabe2017dsq} | Weights and activations | V1 | 91.7 |
|       | CL-BCNN\cite{wu2018ternary} | Weights and activations | V1 | 92.5 |

Table 1. Comparison of accuracy for all tasks. The classes in all datasets are completely balanced.
starting with weights of binary-activation network can boost accuracy (with ~ 1% accuracy gain) as binarization of activation is the bottleneck of whole task. Table 1 lists the accuracies for the CIFAR-10 task trained from our method and other existing methods, which shows that the accuracies from our method are better or comparable to the accuracies from other methods, and approach the accuracies from the full-precision network. In the table, methods like BNN and XNOR-Net use relatively vanilla STE for gradient of activation and thus has relatively low accuracy. Improvement on STE like DSQ is more complicated but can increase the accuracy. While CL-BCNN introduces channel wise interaction, effectively changing the network structure, and can achieve highest accuracy. Our method can be viewed as an independent method from STE, simple but still have relatively good performance.

Here, we discuss some tricks and analysis on binarization. Firstly, for parameterized binarizing function for activation, there can be many choice: sigmoid, sigmoid-ReLU hybrid, \( \text{clip}(x/w, 0, 1) \), etc.. In early layers, symmetric function usually works better as the mean value and shape of distribution of its output is not largely affected. Secondly, it should be mentioned that the L2 regularization (or weight decay)\(^{40}\), often used in ResNet-20 or VGG small networks, may reduce the magnitude of raw weights over time or even flip their signs, or may make the network use the center part of sigmoid function which is not accessible after binarized. This may weaken the effect of reducing \( w \) used in the adiabatic method. From our experiment, when binarizing weights only, this is not problematic at all, but when binarizing activations, the regularizations such as drop-out is recommended in the adiabatic method.

To see whether the binarized neural network is identifying the same defining part of the image as the full-precision network, we investigated the Gradient-weighted Class Activation Mapping (Grad-CAM)\(^{41}\) (for VGG-Small) and usual CAM (for ResNet-20) of full-precision and full-binary network, as shown in Fig. 3. We find that the neural network indeed focuses on the defining part of the image that it correctly identifies. However, for full-binary ResNet-20, we find that the all the weights in the last layer (fully connective layer) are negative, and the heat map is reversed. This suggests that the structure of weights in binary ResNet-20 may be completely different from its full-precision counterpart and can be the reason that ResNet-20 is very difficult to binarize activations.

Discussion and conclusion
With the four different tasks demonstrated above, we showed that, by adiabatically varying the width of the activation and weight distribution towards zero, the existing neural networks can be trained to work with binarized Heaviside activated neurons and/or binarized weights. And the realized binary networks have the validation accuracy approaching that obtained from the conventional non-binary neural networks. The most distinguishing advantage of this adiabatic approach for training binary networks is its applicability to many (if not all) existing neural networks with no or slight change in terms of network size or structure. Reflecting in the coding, this corresponds to a few lines of code change in the training programs (see the program codes\(^{43}\)). Our method can also be applied in conjunction with other existing tricks (e.g. adding channel wise interaction like\(^{42}\)) to obtain better results.

Compared with the non-binary neural networks, the benefits of binary network are obvious. Firstly, a Heaviside activation simplifies the neuron to a 1-bit yes or no decision maker, which would greatly speed-up and simplify both software- and hardware-implemented neuron realizations. For example, a CMOS based hardware-implemented neuron takes several dozens of transistors to construct a sigmoid-activated neuron, but it only requires one transistor for a Heaviside-activated neuron. Binarized weight can heavily reduce the storage along with computation burdens. Compared with a float32 type weight, binary weights only takes about 1/32 of
works10,39. Because the Heaviside activation function simply turns on or off the contributions from some of the last layers as well as batch normalization. The speed up is tested to be around 7 times faster according to other weights, there is no need to read out the weights corresponding to the off-neuron, which reduces the read-out operation (Strictly speaking, only multiplication between ±1 is equal to XNOR, however, multiplication between ±1 weights and any binary activation only differs by a bias term, which can be absorbed into batch normalization layer.). Full binarized VGG-Small network only requires $3 \times 10^3$ flops per sample for un-binarized first and last layers as well as batch normalization. The speed up is tested to be around 7 times faster according to other works10,39. Because the Heaviside activation function simply turns on or off the contributions from some of the weights, there is no need to read out the weights corresponding to the off-neuron, which reduces the read-out operations from $N$ to $\sim N/2$ on average. Because of these obvious benefits, the binarized neural network can have huge advantage in the deployment.

Received: 12 May 2021; Accepted: 20 September 2021
Published online: 05 October 2021

References
1. Krizhevsky, A., Sutskever, I. & Hinton, G. E. Imagenet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. 20, 1097–1102 (2010).
2. Simonyan, K., & Zisserman, A. Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556 (arXiv preprint) (2014).
3. Hinton, G. et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Process. Mag. 29(6), 82–97 (2012).
4. Bahdanau, D., Cho, K. & Bengio, Y. Neural machine translation by jointly learning to align and translate. arXiv:1409.0473 (arXiv preprint) (2014).
5. Silver, D. et al. Mastering the game of go with deep neural networks and tree search. Nature 529(7587), 484–489 (2016).
6. Kröse, B. & van der Smagt, P. An Introduction to Neural Networks (Citepeer, 1993).
7. Goodfellow, I., Bengio, Y., Courville, A. & Bengio, Y. Deep Learning (MIT Press, Cambridge, 2016).
8. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. Nature 323(6088), 533–536 (1986).
9. Courbariaux, M., Bengio, Y. & David, J.-P. Binaryconnect: Training deep neural networks with binary weights during propagations. Adv. Neural Inf. Process. Syst. 20, 3123–3131 (2015).
10. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R. & Bengio, Y. Binarized neural networks. Adv. Neural Inf. Process. Syst. 29, 25 (2016).
11. Rastegari, M., Ordonez, V., Redmon, J. & Farhadi, A. Xnor-net: Imagenet classification using binary convolutional neural networks. In European Conference on Computer Vision, 525–542 (Springer, 2016).
12. Zhang, D., Yang, J., Ye, D. & Hua, G. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In Proceedings of the European Conference on Computer Vision (ECCV), pp. 365–382 (2018).
13. Gong, R. et al. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In Proceedings of the IEEE International Conference on Computer Vision, pp. 4852–4861 (2019).
14. Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H. & Zou, Y. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. arXiv:1606.06160 (arXiv preprint) (2016).
15. Soudry, D., Hubara, I. & Meir, R. Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. Adv. Neural Inf. Process. Syst. 2, 963–971 (2014).
16. Bai, Y., Wang, Y.-X. & Liberty, E. Proxquant: Quantized neural networks via proximal operators. arXiv:1810.00861 (arXiv preprint) (2018).
17. Simons, T. & Lee, D.-J. A review of binarized neural networks. Electronics 8, 6 (2019).
18. Qin, H. et al. Binary neural networks: A survey. Pattern Recogn. 20, 107281 (2020).
19. Howard, A. G., et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. CoRR, arXiv:abs/1704.04861 (2017).
20. Iandola, F. N., et al. SqueezeNet: Alexnet-level accuracy with 50x fewer parameters and < 1mb model size. CoRR, arXiv:abs/1602.07360 (2016).
21. Szegedy, C., et al. Going deeper with convolutions. CoRR, arXiv:abs/1409.4842 (2014).
22. Denil, M., Shakibi, B., Dinh, L., Razooz, M. A. & de Freitas, N. Predicting parameters in deep learning. In Advances in Neural Information Processing Systems Vol. 26 (eds Burges, C. J. C. et al.) (Curran Associates Inc., 2013).
23. Han, S., Pool, J., Tran, J. & Dally, W. J. Learning both weights and connections for efficient neural networks. arXiv:1506.02626 (arXiv preprint) (2015).
24. He, Y., Zhang, X. & Sun, J. Channel pruning for accelerating very deep neural networks. In Proceedings of the IEEE International Conference on Computer Vision, pp. 1389–1397 (2017).
25. Liu, B., Wang, M., Foroosh, H., Tappen, M. & Pensky, M. Sparse convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 806–814 (2015).
26. TensorFlow: https://www.tensorflow.org/.
27. LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. Proc. IEEE 86(11), 2278–2324 (1998).
28. http://yann.lecun.com/exdb/mnist/.
29. Srivastava, N. Improving neural networks with dropout. Univ. Toronto 182(566), 7 (2013).
30. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15(1), 1929–1958 (2014).
31. https://www.kaggle.com/c/dogs-vs-cats/data.
32. Elson, J., Douceur, J. J., Howell, J. & Saul, J. Asirra: A captcha that exploits interest-aligned manual image categorization. In Proceedings of 14th ACM Conference on Computer and Communications Security (CCS). Association for Computing Machinery, Inc. (2007).
33. Ioffe, S. & Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv:1502.03167 (arXiv preprint) (2015).
34. Warden, P. Speech commands: A dataset for limited-vocabulary speech recognition. arXiv e-prints (2018).
35. https://www.kaggle.com/khol Auds/speech commands-v002.
36. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016).
37. https://www.cs.toronto.edu/~kriz/cifar.html.
38. Krizhevsky, A. & Hinton, G. Learning Multiple Layers of Features from Tiny Images. Technical Report, University of Toronto, Toronto (2009).
39. Liu, Z., et al. 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), pp. 722–737 (2018).
40. Hanson, S. J. & Pratt, L. Y. Comparing biases for minimal network construction with back-propagation. In Proceedings of the 1st International Conference on Neural Information Processing Systems, pp. 177–185 (1988).
41. Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D. & Batra, D Grad-cam: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE International Conference on Computer Vision, pp. 618–626 (2017).
42. Wang, Z., Lu, J., Tao, C., Zhou, J. & Tian, Q. Learning channel-wise interactions for binary convolutional neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 568–577 (2019).
43. https://github.com/YuangshengZhaoadiabaticbinary.
44. Sze, V., Chen, Y.-H., Yang, T.-J. & Emer, J. S. Efficient processing of deep neural networks: A tutorial and survey. Proc. IEEE 105(12), 2295–2329 (2017).

Acknowledgements

This work is supported by Science and Technology Commission of Shanghai Municipality (Grant no. 20JC1415900) and Shanghai Municipal Science and Technology Major Project (Grant no. 2019SHZDZX01).

Author contributions

J.X. conceived the main idea, Y.Z. devised the method and carried out the computation. Both authors analyzed the results and wrote the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to J.X.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

© The Author(s) 2021