# An Adiabatic Method to Train Binarized Artificial Neural Networks

**Jiang Xiao**
Fudan University
**Yuansheng Zhao** ( zhao@issp.u-tokyo.ac.jp )
Fudan University

**Research Article**

**Keywords:** artificial neural network, adiabatic method, neuron output

**Posted Date:** May 13th, 2021

**DOI:** https://doi.org/10.21203/rs.3.rs-515926/v1

# An Adiabatic Method to Train Binarized Artificial Neural Networks

Yuansheng Zhao[1,2] and Jiang Xiao[1,3,4,*]

[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 Research Center for Quantum Sciences, Shanghai 201315, China*

(Dated: February 22, 2021)

**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 XOR operations. In this paper, we demonstrate an adiabatic training method that can successfully binarize the dense neural networks and the convolutional neural networks without modification in terms network structure and with very minimal change in training algorithms. This adiabatic training method is tested in the following four tasks: the recognition of hand-writing numbers using a usual dense network, the cat-dog recognition and the audio recognition using a convolutional neural networks, the image recognition with 10 classes (CIFAR-10) using ResNet20 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 [1–5]. In artificial neural networks, the mostly widely used neuron activation functions include the Sigmoid function $\mathrm{sgmd}(x) = 1/[1+\exp(-x)]$, the Hyperbolic Tangent function $\tanh(x)$, and the Rectified Linear Unit function $\mathrm{reLU}(x) = \max(x, 0)$, *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

* Corresponding author: xiaojiang@fudan.edu.cn

nonlinear activation,

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

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 because of its significant enhancement in computing efficiency, which is one the of major obstacles limiting the applications of artificial neural networks. The pioneering works in binarizing neural networks include the BNN proposed by Hubara *et. al.* [9] and the XNOR-Net by Rastegari *et. al.* [10]

Perviously, it was believed that the network with binary activation and/or binary weights simply cannot be trained because good binary weights simply do not exist as the result of inevitable loss of degrees of freedom. Recently, based on the concept of "binarized feed-forward and full-precision back propagation", it is shown that such binary networks are indeed possible [10–14].

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 $w$:

$$\mathrm{sgmd}_w(x) \equiv [1 + \exp(-x/w)]^{-1} \quad \text{and} \quad \tanh_w(x) \equiv \tanh(x/w) \tag{2}$$

to approach the Heaviside step-function $\mathrm{H}(x)$ and the sign-function $\mathrm{sign}(x)$ 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 $a \tanh_{w=0}(W) = \pm a$ as $w \to 0$. The parametrized function used in Eq. (2) has some similarity with the Differential Soft Quantization (DSQ) method proposed by Gong *et. al.* [13]

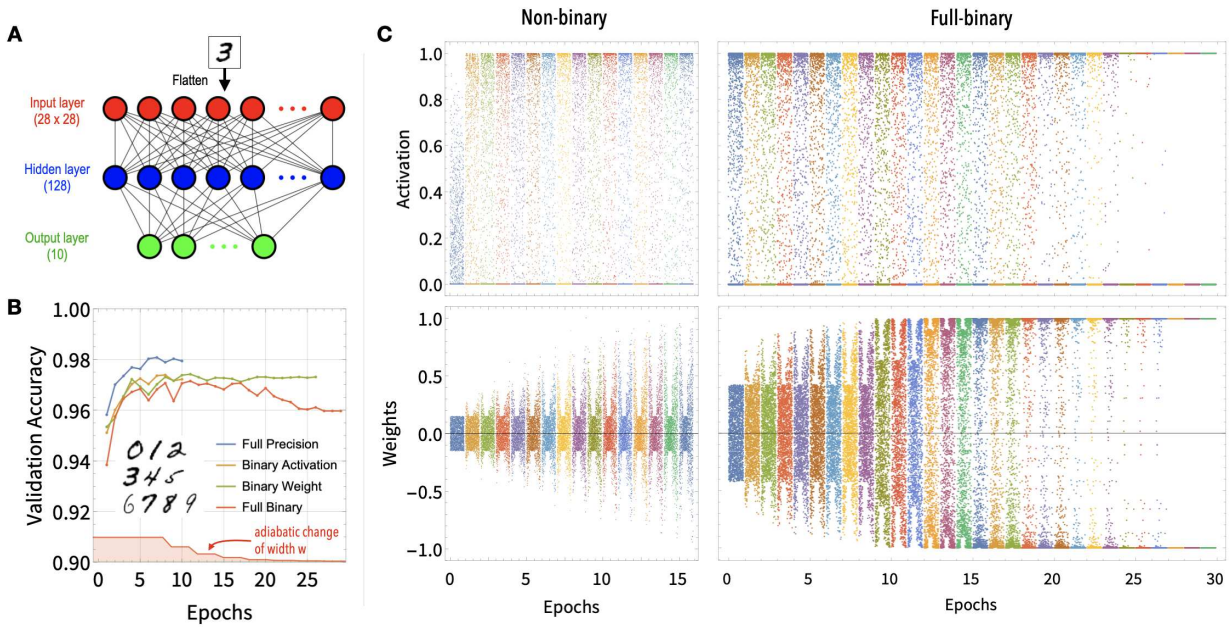


Figure 1. (A) The dense neural network with one hidden layer, used for the recognition of hand-written number images. (B) The comparison of validating accuracies (using the present width) as function of epochs realized by the *non-binary*, *binary-activation*, *binary-weight*, and *full-binary networks* for the numbered image. The adiabatic change of width $w$ as function of epochs for the full-binary case is shown in the bottom of the panel. The sharp dips in the full-binary training curve is due to the sudden change in $w$. (C) The distribution of activation outputs (top) and weights values (bottom) for successive epochs (each color band represents one epoch) for the hand-written numbers recognition task (layer 1 only). The left and right panels are results from the non-binary network (with hybrid Sigmoid-reLU activation function) and the full-binary network, respectively. Each colored band contains 1800 data samples out of $\sim$ 100k weights or 1 million activations. The weights corresponding to the edges of the image remain unchanged during training.

We use the adiabatic training method described above to binarize the following four different tasks based on some typical neural network structures: (1) a dense 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 [15].

**(1) Image recognition of hand-written numbers**: A dense network with one hidden layer (see Fig. 1(A)) is sufficient for this task [16]. And 70k image samples from the `MNIST` dataset [17] are used with 60k for training and 10k for validating. Drop-out [18, 19] is used to alleviate over-fitting. For training the non-binary network, the neurons in the hidden layer use the reLU activation function. After 10 epochs of training, the validation accuracy for a non-binary network reaches over 97%. [20] Since the reLU function can not go smoothly to a Heaviside step-function, to train binary networks, we use a hybrid Sigmoid-reLU activation function as

$$f_w(x) = 2\,\mathrm{sgmd}\left[\mathrm{reLU}(x)/w\right] - 1, \tag{3}$$

which can approaches the Heaviside function as $w \to 0$. The same dense 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 2 epochs using the Heaviside activation. The realized *binary-activation network* with Heaviside activation has the validation accuracy over 97% as shown in Fig. 1(B), similar to the conventional *non-binary network* using reLU activation. It should note that the initial 8 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 the network is trained for 8 epochs using the polarized weight with $w = 1/3$, followed by 3 epochs each for $w = 1/10, 1/20, 1/50, 1/100, 1/300$ and 0, and finally the realized *binary-weight network* reaches over 97% validation 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 $w_{\text{weights}} = 2w_{\text{activation}}$, and $w_{\text{weights}}$ is decreased in the same way as in training the binary-weight network but with an additional step at $w_{\text{weights}} = 1/5$ in the beginning. The final validation accuracy of 96% is reached. Therefore, both the half-binary and full-binary networks can reach almost the same validating accuracy as the non-binary network.

Fig. 1(C) 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 $\mathrm{Range}(|W|)$ and the initial value for width $w$ should be $\mathrm{Range}(|W|)/w \gtrsim 1$ (for the

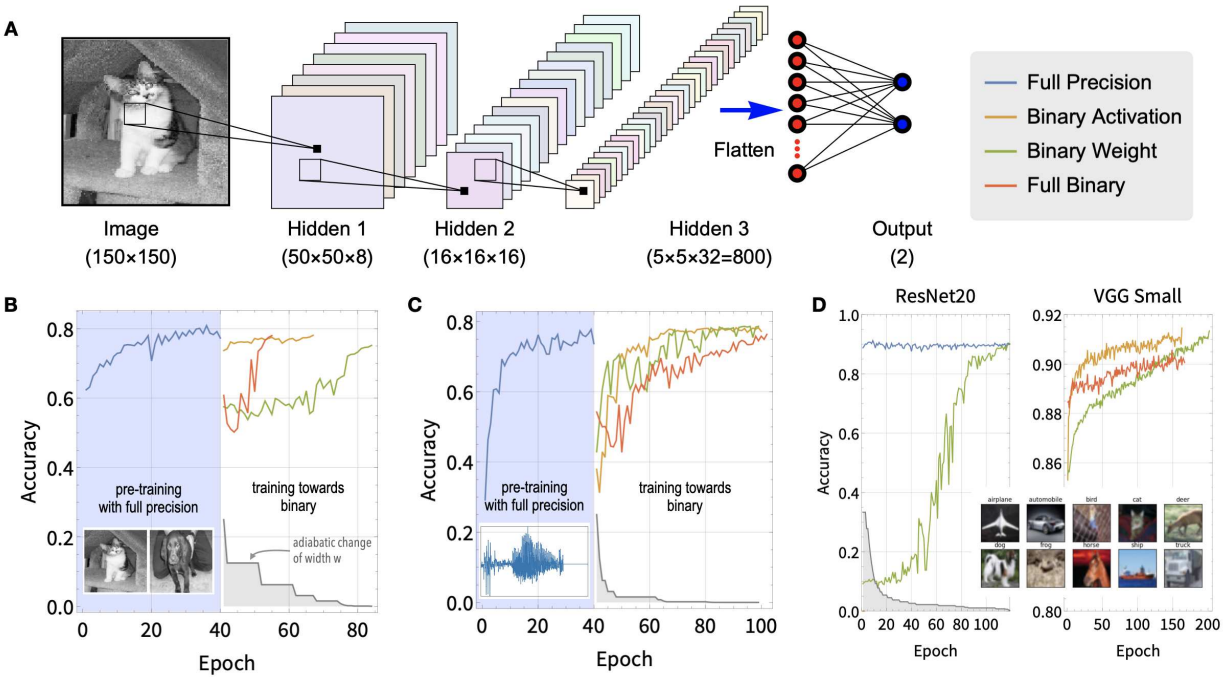


Figure 2. (A) The convoluted neural network with three hidden layers used for dog-cat recognition and the spoken number recognition. (B) The validating accuracies as function of number of epochs for the dog-cat task using networks trained with full precision (blue), binary activation (orange), binary weights (green), and full binary (red). The gray step-wise curve shows the adiabatic change of width $w$ towards zero as function of epochs. (C) Same as (B) for the spoken number task. (D) results for the CIFAR-10 recognition task based on the ResNet20 and VGG-Small networks; all trainings start from pre-trained (not shown) full precision network except full-binary VGG network, which starts from the binary-activation network instead.

first band of Fig. 1(C) for the full-binary panel, $\mathrm{Range}(|W|) \simeq 1/2$ and $w = 1/3$), and in the mean time the learning rate should be moderately large to avoid the center part.

**(2) The dog-cat picture recognition**: A convolutional neural network (CNN) with three hidden layers (see Fig. 1(C)) are used for this task. In this network, both the convolution kernel and the pooling size are $3 \times 3$. And 3k pictures of dogs and cats from the Kaggle dataset [21] are used with 2k for training and 1k for validating. The pictures are resized to $150 \times 150$ pixels and converted to grayscale before training. For the conventional non-binary network, the standard sigmoid activations with max-pooling are used in all hidden layers, and batch normalization [22] is used to accelerate training. Within 40 epochs of training, the validating accuracy saturates to 80% as shown in Fig. 2(B). To train a binary-activation network with the Heaviside activation, the sigmoid activation is replaced by the parametrized sigmoid activation Eq. (2) with varying width $w$ for the hidden layers. 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 Ref. [10]. From this task, the following self-adaptive method is used for adjusting the width: $w$ starts with 1 and is decreased by a factor of $1.2 \sim 2$ (depending on tasks, but not crucial) when the validation accuracy (with current width) reaches the target value and the learning rate is also reduced by a factor of $1 \sim 2$. If the target validation accuracy cannot be reached within $8 \sim 30$ epochs (depending on tasks), $w$ is reduced anyway, and the the target accuracy will be set to the present accuracy. With these operations, the realized binary-activation network equipped with Heaviside activation functions reaches a validating accuracy of 78% (see the orange curve in Fig. 2(B)). For a *binary-weight network*, we use the same method to reduce the width $w$ and a validation accuracy of 75% is achieved (see Fig. 1(D)). 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 [13, 23]. The width $w$ for the activations and weights can be reduced to zero simultaneously to achieve full binarization, and realized a validation accuracy of 78%, close to the non-binary and half-binary networks.

**(3) Voice recognition of spoken numbers**: This task uses the same convolutional neural network (see Fig. 2(A)) 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 6, respectively. And 3k audio waveforms of human spoken numbers (each is an array of length 16000) from Kaggle dataset are used with 2k for training and 1k for validating. [24]. The non-binary network trained using the reLU-sigmoid hybrid activation and real-valued weights along with max-pooling reachs a validating accuracy of $\sim$79% (see the blue curve in Fig. 2(C)). Following the same procedure as that used in the dog-cat case, the validating accuracies for the binary-activation (the orange curve in Fig. 2(C)), binary-weight (the green curve), and full-binary networks (the red curve) trained using the adiabatic method are 78%, 78%, and 76% , all of which are close to the non-binary network.

**(4) CIFAR-10 classification**: In this task, we train a standard ResNet-20 [25] or VGG-Small network [2, 23] (with similar structure as the CNN shown in Fig. 2(A)) to recognize 60K (50K for training and 10K for validation) $32 \times 32$ color images belonging to 10 classes from the CIFAR-10 dataset [26]. This task is much more challenging than previously 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 Ref. [25], 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. 2(D), where the validation accuracy of binary-weight network approaches to the accuracy of the full-precision network. However, we do not succeed in training a binary-activation or full-binary ResNet-20 network, which may be related to short-cut mechanism in the network [25] so that full precision network

Table I. Comparison of accuracy based on different methods on CIFAR-10 task

| Networks | Method | Binarization | Accuracy (%) |
|---|---|---|---|
| ResNet20 | Ours | full precision | 92.2 |
| | **Ours** | **weights only** | **90.1** |
| | DoReFa [28] | weights only | 90.0 |
| | LQ-Net [23] | weights only | 90.1 |
| | DSQ [13] | weights only | 90.2 |
| VGG-Small | Ours | full precision | 92.4 |
| | **Ours** | **weights only** | **91.5** |
| | BinaryConnect [29] | weights only | 91.7 |
| | BWN [10] | weights only | 90.1 |
| | LQ-Net [23] | weights only | 93.5 |
| | **Ours** | **activation only** | **91.3** |
| | **Ours** | **weights and activations** | **90.6** |
| | BNN [12] | weights and activations | 89.9 |
| | XNOR-Net [10] | weights and activations | 89.8 |
| | DSQ [13] | weights and activations | 91.7 |

cannot be smoothly transformed into a binary one. On the other hand, VGG-Small networks with binary-activation can be trained using the adiabatic method if we start with random weights (no pre-training using reLU activations), or with pre-training weights using the sigmoid activation function instead of reLU. Furthermore, the full-binary VGG-Small networks (see the right panel of Fig. 2(D)) can also be trained using the adiabatic method, and the best accuracy (with $\sim 1\%$ accuracy gain) is achieved by starting with the weights from the binary-activation network then followed by reducing the $w_{\text{weight}}$ and $w_{\text{activation}}$ alternatively. The realized validation accuracies for the binary-activation, binary-weight, and full-binary networks for the VGG-Small network using the adiabatic method are 91.5%, 91.3%, and 90.6%, respectively. All are close to the full-precision network result (92.4%). Table I 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. It should be mentioned that the L2 regularization (or weight decay [27]), often used in ResNet20 or VGG small networks, may reduce the magnitude of raw weights over time or even flip their signs. This weakens the effect of reducing $w$ used in the adiabatic method. Therefore, the regularizations such as Drop-out is recommended in the adiabatic method.

*Discussion & Conclusion:* With the four different tasks demonstrated above, we showed that, by adia-

batically 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 without any 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 in the Supplementary Materials).

Compare with the non-binary neural networks, the benefits of binary network are obvious. Firstly, a Heaviside activation simplifies the neuron to an 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, a binary weights only takes about $1/32$ of storage space. Secondly but more dramatically, the most time/area/energy-consuming part of a conventional non-binary neural network is the weight readout and the multiply-accumulate (MAC) operation [30] which calculates the dot product of the outputs of previous layer neurons and the weights. If the number of neurons involved is $N$, then one MAC operation includes $N$ multiplication and $N$ addition operations. However, for a binary-activation neural network, since the neuron outputs are either $0$ or $1$, the matrix multiplication operation is completely unnecessary, one needs only sum over those weights associated with neuron output $1$ and omitting all the weights corresponding to zero outputs. The binarized weights in a binary-weight network can reduce matrix multiplication for a similar reason. For full-binary neural networks, since both neuron outputs and the weights are binary, the addition operation is further simplified to a single type of counting (XOR) operation. 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.

This work was supported National Science Foundation of China (Grant No. 11722430) and Shanghai Municipal Science and Technology Major Project (Grant No. 2019SHZDZX01).

---

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, in *Advances in neural information processing systems* (2012) pp. 1097–1105.

[2] K. Simonyan and A. Zisserman, arXiv preprint arXiv:1409.1556 (2014).

[3] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, *et al.*, IEEE Signal processing magazine **29**, 82 (2012).

[4] D. Bahdanau, K. Cho, and Y. Bengio, arXiv preprint arXiv:1409.0473 (2014).

[5] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, nature **529**, 484 (2016).

[6] B. Kröse and P. van der Smagt, *An introduction to neural networks* (Citeseer, 1993).

[7] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning* (MIT press Cambridge, 2016).

[8] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, nature **323**, 533 (1986).

[9] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, in *Proceedings of the 30th international conference on neural information processing systems* (2016) pp. 4114–4122.

[10] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, in *European conference on computer vision* (Springer, 2016) pp. 525–542.

[11] D. Soudry, I. Hubara, and R. Meir, in *Advances in neural information processing systems* (2014) pp. 963–971.

[12] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, arXiv preprint arXiv:1602.02830 (2016).

[13] R. Gong, X. Liu, S. Jiang, T. Li, P. Hu, J. Lin, F. Yu, and J. Yan, in *Proceedings of the IEEE International Conference on Computer Vision* (2019) pp. 4852–4861.

[14] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, and N. Sebe, Pattern Recognition , 107281 (2020).

[15] TensorFlow: `https://www.tensorflow.org/`.

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Proceedings of the IEEE **86**, 2278 (1998).

[17] `http://yann.lecun.com/exdb/mnist/`.

[18] N. Srivastava, University of Toronto **182**, 7 (2013).

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, The journal of machine learning research **15**, 1929 (2014).

[20] In the optimization, the `adam` optimizer [31] and `sparse_categorical_crossentropy` loss are used and other parameters are set to default values.

[21] `https://www.kaggle.com/c/dogs-vs-cats/data`.

[22] S. Ioffe and C. Szegedy, arXiv preprint arXiv:1502.03167 (2015).

[23] D. Zhang, J. Yang, D. Ye, and G. Hua, in *Proceedings of the European conference on computer vision (ECCV)* (2018) pp. 365–382.

[24] `https://www.kaggle.com/mok0na/speech-commands-v002`.

[25] K. He, X. Zhang, S. Ren, and J. Sun, in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016) pp. 770–778.

[26] `https://www.cs.toronto.edu/~kriz/cifar.html`.

[27] S. J. Hanson and L. Y. Pratt, in *Proceedings of the 1st International Conference on Neural Information Processing Systems* (1988) pp. 177–185.

[28] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, arXiv preprint arXiv:1606.06160 (2016).

[29] M. Courbariaux, Y. Bengio, and J.-P. David, in *Advances in neural information processing systems* (2015) pp. 3123–3131.

[30] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, Proceedings of the IEEE **105**, 2295 (2017).

[31] D. P. Kingma and J. Ba, arXiv preprint arXiv:1412.6980 (2014).

# Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- SupplementaryMaterials.zip