BITWISE NEURAL NETWORKS FOR EFFICIENT SINGLE-CHANNEL SOURCE SEPARATION

Minje Kim[*]

Indiana University
Department of Intelligent Systems Engineering
minje@indiana.edu

Paris Smaragdis[†]

University of Illinois at Urbana-Champaign
Adobe Research
paris@illinois.edu

ABSTRACT

We present Bitwise Neural Networks (BNN) as an efficient hardware-friendly solution to single-channel source separation tasks in resource-constrained environments. In the proposed BNN system, we replace all the real-valued operations during the feedforward process of a Deep Neural Network (DNN) with bitwise arithmetic (e.g. the XNOR operation between bipolar binaries in place of multiplications). Thanks to the fully bitwise run-time operations, the BNN system can serve as an alternative solution where efficient real-time processing is critical, for example real-time speech enhancement in embedded systems. Furthermore, we also propose a binarization scheme to convert the input signals into bit strings so that the BNN parameters learn the Boolean mapping between input binarized mixture signals and their target Ideal Binary Masks (IBM). Experiments on the single-channel speech denoising tasks show that the efficient BNN-based source separation system works well with an acceptable performance loss compared to a comprehensive real-valued network, while consuming a minimal amount of resources.

Index Terms— Bitwise neural networks, deep learning, speech enhancement, source separation, low-power computing

1. INTRODUCTION

Recently deep learning has become one of the major forces in improving the performance of machine learning-based source separation tasks, thanks to its powerful multi-layered structure that can learn complex mapping functions between a large amount of training samples (e.g. noisy speech) and their corresponding target values (e.g. clean speech) [1, 2, 3, 4, 5, 6]. In many research areas the Deep Neural Network (DNN) topology is believed to capture a hierarchy of features [7], which eventually provides a better performance in supervised learning tasks. Dictionary-based models by using Non-negative Matrix Factorization (NMF) [8, 9], on the other hand, learn a set of basis vectors that correspond to the weights of a shallow network. In general, we can say that the multiple hidden layers in the deep network structure can learn some more abstraction about the data at the cost of training and maintaining a larger amount of parameters, which can easily amount to a few millions.

A properly trained neural network can do the source separation job in a single feedforward step, which can be recursively defined for t-th data sample as follows:

$$\mathbf{a}_t^{(L+1)} \leftarrow \mathbf{W}^{(L+1)}\mathbf{z}_t^{(L)} + \mathbf{b}^{(L+1)}, \quad \mathbf{z}_t^{(L+1)} \leftarrow f\big(\mathbf{a}_t^{(L+1)}\big),$$
$$\mathbf{a}^{(l)} \leftarrow \mathbf{W}^{(l)}\mathbf{z}_t^{(l-1)} + \mathbf{b}^{(l)}, \quad \mathbf{z}_t^{(l)} \leftarrow g\big(\mathbf{a}_t^{(l)}\big), \tag{1}$$

where $l = \{1, \cdots, L+1\}$ denotes the layer index. Therefore, $\mathbf{z}_t^{(L)}$ stands for the output of the last hidden layer while $\mathbf{z}_t^{(0)}$ denotes the input vector. $g(\cdot)$ and $f(\cdot)$ are the activation functions in the hidden and the final layers, respectively. $K^{(l)}$ is the number of units at l-th layer. In a layer during the feedforward pass the weight matrix $\mathbf{W}^{(l)} \in \mathbb{R}^{K^{(l)} \times K^{(l-1)}}$ is multiplied by the output from the previous layer, and then the bias vector $\mathbf{b}^{(l)} \in \mathbb{R}^{K^{(l)}}$ is added to it. Activation function $g(\cdot)$ saturates the results by adding nonlinearity.

The main contribution of this paper is to develop an efficient feedforward procedure that minimizes the computational and spatial complexity of running and maintaining a DNN-based source separation system. The deep learning advances in source separation systems cost more resources, such as memory and power, due to the enlarged networks in terms of L and $K^{(l)}$. For example, now the network has to compute the linear operation in (1) for more hidden layers with larger $\mathbf{W}^{(l)}$'s. Since those resources can be constrained in embedded devices, it could be prohibitive for them to perform multiplications between large matrices with millions of elements, although it is a fairly typical computation size in many DNNs.

We employ Bitwise Neural Networks (BNN) [10] to compress the network for an efficient implementation in a resource-constrained environment. BNN's drastically simplified feedforward operation comes from the fact that in a BNN all the input and output signals, weights and biases of the networks, and operations on them are all defined in an efficient bitwise fashion, which we will review more thoroughly in Section 2. If we represent the input signals with bipolar binary numbers, i.e. $+1$ and -1, and we train the bipolar binary parameters as well, the feedforward part can be done using only bitwise logics such as XNOR and bit counting, instead of multiplication, addition, and a nonlinear activation (e.g. tanh) on the usual floating or fixed-point variables. Note that since each weight and node will be encoded with binaries, the space requirement is also reduced compared to the multi-bit encoding schemes.

In all these procedures, we use binarized signals, so that the network can work in a fully bitwise fashion. For example, hidden unit output signals are already binarized thanks to the sign function as the activation, though binarization of input and target variables is an open question. In this paper, we propose a binarization technique, Quantization-and-Dispersion (QaD), which effectively encodes magnitude spectra. As for the target variable, Ideal Binary Masks (IBM) are a natural choice, although the same QaD process

[*]This work was partly supported by Intel Corporation.
[†]Supported by NSF grant 1453104

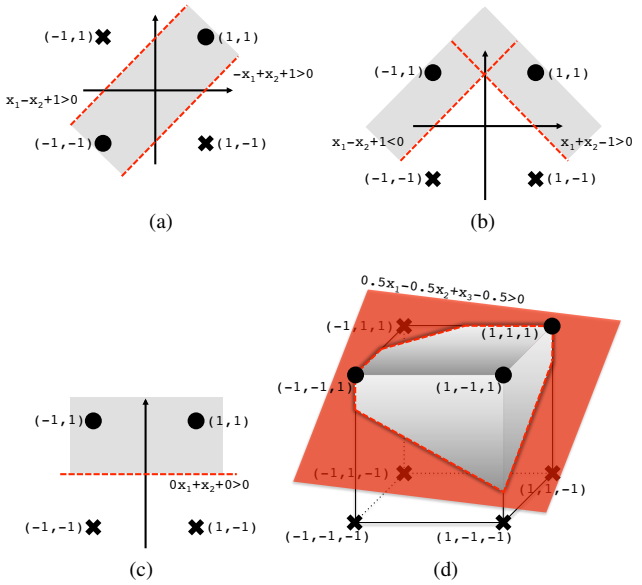

Fig. 1. (a) The XOR problem and a pair of corresponding binary hyperplanes that solves it. (b) A linearly separable problem where bitwise networks need two hyperplanes to solve it. (c) A bitwise network with zero weights to solve the problem with a single hyperplane. (d) Another linearly separable case with real-valued coefficients ($0.5x_1 - 0.5x_2 + x_3 - 0.5 > 0$) that however needs more than one hyperplanes in BNN. Some figures are from [10].

can potentially convert any real-valued target variables. Experimental results on some single-channel speech denoising tasks verify that the proposed BNN system gives comparable results to the similarly structured DNNs with near-continuous encoding strategy.

2. BITWISE NEURAL NETWORKS (BNN)

2.1. Background: Neural Networks with Bitwise Feedforward

Although it has been known that any Boolean function can be represented as a bitwise network with one hidden layer, e.g. by memorizing all the relationships [11], its training is an NP-complete problem [12]. μ-perceptron networks were proposed as a bitwise network, but its topology does not allow a full connection between units [13]. Soudry et. al. recently proposed the Expectation Back Propagation (EBP) algorithm to estimate the posterior probabilities for the bitwise parameters [14]. It is a parameter-free learning algorithm and the discretization is convenient, yet it allows real-valued bias terms and relies on the averaged outputs of multiple networks whose parameters are sampled from the estimated distribution.

BNN can be seen as one of different ways to achieve the fully binary computation during the test time. More recently, there have been more neural networks that learn fully binary network parameters such as BNN [10], BinaryConnect [15], and binarized neural networks [16]. In the early stage of the neural network research, it has been known that if we decrease the quantization level of already trained network parameters, the performance of the network drops significantly. One way to avoid this effect is to inject the quantization error during the training phase, by using the quantized version of the original continuous parameters during the feedforward so that the network is aware of the additional error introduced by the quantization and can fix it during backpropagation [17, 18]. BNN adopts the quantization noise injection technique to estimate its bitwise parameters.

2.2. Feedforward in BNNs

2.2.1. *Notations and setup: bipolar binaries*

Throughout the paper, we use bipolar binaries where the Boolean values are represented as $+1$ and -1. Therefore, we use XNOR (we denote it by $\otimes$) as a corresponding multiplier in this bipolar binary domain. The bipolar binaries are more expressive than 0-1 binaries. For example, in Figure 1 (a), the two hyperplanes (red dashes) can be fully defined with bipolar binary coefficients and biases, while 0-1 binaries fail to do so. Moreover, with the bipolar binaries, we can make use of zeros to explain the sparsity concept. With no loss of generality, in this paper we use the ± 1 bipolar representation.

2.2.2. *The feedforward process*

In a BNN the feedforward pass is defined as follows:

$$\bar{a}_i^{(l)} = \bar{b}_i^{(l)} + \sum_j^{K^{(l-1)}} \bar{w}_{i,j}^{(l)} \otimes \bar{z}_j^{(l-1)}, \quad \bar{z}_i^{(l)} = \text{sign}\big(\bar{a}_i^{(l)}\big), \tag{2}$$

$$\bar{\mathbf{z}}^{(l)} \in \mathbb{B}^{K^{(l)}}, \bar{\mathbf{W}}^{(l)} \in \mathbb{B}^{K^{(l)} \times K^{(l-1)}}, \bar{\mathbf{b}}^{(l)} \in \mathbb{B}^{K^{(l)}}, \bar{\mathbf{a}}^{(l)} \in \mathbb{Z}^{K^{(l)}},$$

where $\mathbb{B} = \{-1, +1\}$ and all lowercase letters are for scalar elements. i and j indicate i-th input and j-th output units of a layer, respectively. Bold characters are for vectors, and matrices are capitalized. We introduce the upper bar notation to distinguish the integer or binary parameters of the BNN from the real-valued parameters in (1). For convenience we drop the sample index. Sign function is used as an activation, which will produce bipolar binaries as the output. The sign function takes an integer $a_i^{(l)}$ as its input, whose value can be from $-K^{(l)}-1$ to $K^{(l)}+1$. $\otimes$ and the sign activation can be seen as special operations designed for the desired speed-up, since they can replace the original real-valued multiplication and the smooth step functions, respectively.

2.2.3. *Linear separability and bitwise hyperplanes*

Although there are cases where both a BNN and a real-valued network can solve the same problem even with the same topology, in general the BNN could need a larger network structure. For example, for the XOR problem in Figure 1 (a), we can see that binary weights and bias are good enough to define the hyperplanes, $x_1 - x_2 + 1 > 0$ and $-x_1 + x_2 + 1 > 0$. On the other hand, in Figure 1 (b) BNN necessitates multiple hyperplanes for a linearly separable problem (for example, we can solve it with a single real-valued hyperplane $-0.1x_1 + x_2 + 0.5 > 0$), because binary coefficients cannot represent those solutions. The BNN solves this problem by combining multiple hyperplanes, but they will eventually increase the complexity.

Sparsity plays a great role not only in providing more freedom to the hyperplanes, but in reducing the model complexity of BNNs, because BNN parameters are often redundant, too. By introducing the sparsity in the coefficient, we can *turn off* some dimensions. Since our coefficients are either $+1$ or -1, zeros are natural choice to express inactivity. For example, we allow zeros in the bipolar weights, the BNN can turn off the x_2 axis to come up with a solution with only one hyperplane as in (c). We can employ either a sparse representation or another bit to encode a BNN with zero weights. First, with the

sparse representation we can only store the non-zero parameters and their locations while skipping zeros. By doing so, in practice (e.g. in hardware chips) these bipolar binaries will be implemented using 0-1 binary values, where the activation in (2) is equivalent to checking whether the number of +1's is bigger than the half of the number of input units plus 1. Or, if the weight matrix is not sparse enough to afford the overhead, we can simply use an additional bit to encode the parameter activity. We will use the sparsity concept in this work as our BNN weight matrices turn out to be 95% sparse. However, in general we expect that there are some problems that BNNs need more hyperplanes, such as in Figure 1 (d), which is linearly separable though no BNN can solve it with a single hyperplane.

Note that this does not always mean that a BNN with more network parameters is computationally more complex than a corresponding real-valued one, because a parameter or a unit in BNNs calls for only one bit to encode. Hence, BNN's resource usage can be still lower than its corresponding real-valued network.

2.3. Efficiency of bitwise feedforward in hardware

It has been shown that bitwise feedforward operations with XNOR and bit counting are efficient in hardware implementations. For example, XNOR-Net showed that the bitwise convolutional feedforward for computer vision tasks uses about 1.5% of the memory space that a comprehensive double-precision network spends and 20 to 60 times faster in computation [19]. It is also shown that binarized weights can lead to 7 times faster feedforward than an ordinary floating-point network for MNIST classification, or 23 faster large matrix multiplication tasks on GPU [16]. Also, compared to the single precision multiplier, XNOR operation is 200 times cheaper in FPGA implementations [20, 21]. Based on this literature, we assume that the proposed BNN will also be significantly efficient in hardware implementations.

2.4. Training BNNs

We can train a BNN by using two sequential runs of the Stochastic Gradient Descent (SGD) procedure. We first train an ordinary real-valued network, whose network structure is the same with the desired BNN. We use its weights to initialize the BNN parameters in the subsequent step. In the second phase of training we finally learn the bitwise weights using a noisy feedforward pass.

2.4.1. The first round: learning a weight-compressed network

First, we train a real-valued network that takes binary inputs (we will see how to binarize audio signals in Secion 3). In this first-round, we wrap the weights and bias with the hyperbolic tangent function, $\tanh(\cdot)$, so that the weights are bounded between -1 and $+1$. Since $\tanh(\cdot)$ is used for the activation as well, the network can be seen as a relaxed version of the corresponding bipolar BNN. The weight-compressed forward pass is defined as follows:

$$a_i^l = \tanh(b_i^{(l)}) + \sum_j^{K^{(l-1)}} \tanh(w_{i,j}^{(l)}) z_j^{(l-1)},\ z_i^l = \tanh\left(a_i^l\right), \quad (3)$$

where all the values and operations are real-valued. Therefore, the only difference between (3) and (1) is the use of $\tanh$ for both weight compression and in place of all the activation functions.

During the backpropagation procedure weight compression introduces an additional factor. In the l-th layer the backpropagation error for the n-th training sample is calculated using the compressed versions of the weights:

$$\delta_j^{(l)}(n) = \Big(\sum_i^{K^{(l+1)}} \tanh(w_{i,j}^{(l+1)}) \delta_i^{(l+1)}(n) \Big) \cdot \Big(1 - \tanh^2\left(a_j^l\right) \Big). \quad (4)$$

Finally, the gradients are defined as follows:

$$\nabla w_{i,j}^l = \Big(\sum_n \delta_i^l(n) z_j^{(l-1)} \Big) \cdot \Big(1 - \tanh^2\left(w_{i,j}^l\right) \Big), \quad (5)$$

$$\nabla b_i^l = \Big(\sum_n \delta_i^l(n) \Big) \cdot \Big(1 - \tanh^2\left(b_i^l\right) \Big). \quad (6)$$

Note that from the chain rule the derivatives of tanh is additionally multiplied. The summation over n is defined by the minibatch size.

2.4.2. The second round: noisy feedforward

Now that we have trained a real-valued network with a properly ranged weights, the next step is to train the actual bitwise network. The training procedure is similar to the ones with quantized weights [17, 18], but now all the input signals and weights are constrained to be bipolar binaries, so that the operations on them are bitwise. The full binary setting is of particular importance since it can avoid the computations in-between fixed-point or floating-point variables.

Noisy feedforward pass: To this end, we first initialize all the real-valued parameters, $\mathbf{W}$ and $\mathbf{b}$, with the ones learned from section 2.4.1, but using their compressed version, e.g. $\mathbf{W} \leftarrow \tanh(\mathbf{W})$. At every epoch, based on the pre-defined sparsity value ρ, the boundary $\beta^{(l)}$ is calculated from the following equation:

$$(K^{(l-1)}+1)K^{(l)}\rho = \sum_{i,j} \mathcal{I}(|w_{i,j}^{(l)}| < \beta^{(l)}) + \sum_i \mathcal{I}(|b_i^{(l)}| < \beta^{(l)}), \quad (7)$$

Where $\mathcal{I}(\cdot)$ is an indicator function. Using $\beta^{(l)}$, we refresh the binarized parameters $\bar{\mathbf{W}}$ and $\bar{\mathbf{b}}$ as follows:

$$\bar{w}_{i,j}^{(l)} = \begin{cases} +1 & \text{if } w_{i,j}^{(l)} > \beta \\ -1 & \text{if } w_{i,j}^{(l)} \leq -\beta \\ 0 & \text{otherwise} \end{cases},\ \bar{b}_i^{(l)} = \begin{cases} +1 & \text{if } b_i^{(l)} > \beta \\ -1 & \text{if } b_i^{(l)} \leq -\beta \\ 0 & \text{otherwise} \end{cases}. \quad (8)$$

By doing so, we can achieve the desired sparsity ρ. This noisy feedforward step can be seen as a way to let the network be aware of the additional error introduced by the binarization procedure, so that the backpropagation step can fix it. Note that we use the upper bar notation to signify binarized parameters. Once the binarization of the parameters is done, we use them for the (noisy) feedforward process during training instead of the real-valued versions.

Backpropagation: In the l-th layer we calculate the backpropagation error and the gradients:

$$\delta_j^{(l)}(n) = \Big(\sum_i^{K^{(l+1)}} \bar{w}_{i,j}^{(l+1)} \delta_i^{(l+1)}(n) \Big) \cdot \Big(1 - \tanh^2\left(\bar{a}_j^l\right) \Big), \quad (9)$$

$$\nabla w_{i,j}^{(l)} = \sum_n \delta_i^{(l)}(n) z_j^{(l-1)}, \quad \nabla b_i^{(l)} = \sum_n \delta_i^{(l)}(n). \quad (10)$$

Note that in the calculation the bitwise parameters with a bar on the top are used. In this way, the gradients and errors properly take the binarization of the weights and the signals into account. Also, the non-differentiable sign activation function is relaxed with $\tanh$, which introduces the derivative of $\tanh$. Since the gradients can

get too small to update the binary parameters $\mathbf{W}$ and $\mathbf{b}$, we instead update their corresponding real-valued parameters,

$$w_{i,j}^{(l)} \leftarrow w_{i,j}^{(l)} - \eta \nabla w_{i,j}^{(l)}, \quad b_i^{(l)} \leftarrow b_i^{(l)} - \eta \nabla b_i^{(l)}, \qquad (11)$$

with η as a learning rate parameter. Once they are updated, in the next epoch they are used to define the binarized parameters using the newly calculated $\beta^{(l)}$.

3. BINARIZATION OF SIGNALS

3.1. Quantization and Dispersion (QaD)

Since a BNN takes a bit pattern as the input, we need a binarization technique to encode any real-valued input signals. We found that Lloyd-Max's quantization [22] is useful, which could convert an input magnitude into a fixed-point value and feed it into an input unit. Instead of this integer input string, we treat each bit of the fixed-point quantized value as a binary feature. For example, after encoding the f-th magnitude coefficient of a noisy speech spectrum $\mathbf{x}_t$ into 4 bits, we disperse the 4 bits into 4 corresponding input units. Therefore, the BNN takes a $4F$-dimensional binary vector, where F is the original number of coefficients in the spectrum.

3.2. Ideal Binary Masks (IBM)

As for the output units it is natural to form a softmax layer to solve classification problems as in [10]. On the other hand, the source separation problems often form continuous target variables such as the Ideal Ratio Masks (IRM) [23], for which we believe that the same QaD technique can be employed to convert them into bit patterns, too. In this paper, we conveniently make use of the IBM as our target, which is a natural choice for us to binarize the target [3]. We leave the investigation of QaD for the continuous target variables to future work. Finally, the prediction error $\mathcal{E}$ can be measured as follows: $\mathcal{E}(n) = \frac{1}{2}\sum_i^{K^{(L+1)}} \left(t_i(n) - \bar{z}_i^{(L+1)}(n)\right)^2$, where $t_i(n)$ is an element of the bipolarized IBM mask.

4. EXPERIMENTS

4.1. Experimental Setups

We prepare 121,280 training spectra (18,020 of them are used for validation). Twelve gender-balanced TIMIT speakers are chosen for training, each of which contributes five chosen utterances, totalling 60 clean speech signals. They are then mixed with ten different non-stationary noise signals proposed in [24] with 0 dB Signal-to-Noise Ratio (SNR) to form 600 noisy utterances. Among them, we set aside 100 utterances as a validation set. By applying Short-Time Fourier Transform (STFT) with a Hann window of 1024 points and a 75% of overlap. For testing, another four speakers are chosen and mixed with the same set of noise signals, but from different parts to make sure the test mixtures are not seen during training. We apply a 4-bit QaD procedure to these spectra as an input to the BNN systems. We found $\rho = 0.95$ optimal via validation. All signals are with sampling rate 16 kHz. We train three different kinds of neural networks that predict the IBM of the given magnitude spectrum:

• *Baseline*: The baseline networks take the ordinary 513 dimensional real-valued magnitude spectrum as its input. For training this network the first round training algorithm is used, where the additional weight compression works like max-norm feature in [25]. It employs dropout with 0.95 for the first layer (dropping 5%) and 0.8 for the other layers as the parameter.

Systems	Topology	SDR	SIR	SAR	STOI
Baseline with original input	1024×2	10.17	26.69	10.45	0.7880
	2048×2	10.57	26.25	10.88	0.8060
Baseline with binary input	1024×2	9.80	27.00	10.08	0.7790
	2048×2	10.11	26.61	10.43	0.7946
BNN	1024×2	9.35	23.38	9.82	0.7819
	2048×2	9.82	23.62	10.30	0.7861

Table 1. Speech denoising performance of the proposed BNN-based separation system compared with the other real-valued networks.

• *Baseline with binary input*: This setup is equivalent to the first round of the BNN training. The difference between this and the baseline is that the first round networks take the 4×513 4-bit QaD vectors as their input. The prediction results from these first round networks will serve as an upper bound of the separation performance of a real-valued network with the binarized input. Also, the learned parameters will be reused to intialize the second round parameters.

• *The proposed BNN*: Here we combine the first round results (baseline with binary input) and the second round.

Validation determines the learning rate which usually starts from 10^{-7} or 10^{-6} and gradually decreases. Minibatch and the momentum parameter are set to be 100 frames and 0.95, respectively.

4.2. Discussion

Table 1 lists the speech denoising performance of the systems in terms of Signal-to-Distortion Ratio (SDR), Signal-to-Interference Ratio (SIR), Signal-to-Artifact Ratio (SAR) [26], and Short-Time Objective Intelligibility (STOI) [27]. First, we can see that doubling up the number of hidden units generally improves SAR, which eventually improves SDR as well. STOI gets better with more hidden units, too. For example, BNN with 2048 hidden units catches up the performance of 1024×2 DNN with binary input (9.82 versus 9.80 dB in SDR and 0.7861 versus 0.7790 in STOI). If we compare the effect of the QaD binarization (the baseline systems with original magnitudes versus with the QaD input), we see a slight performance drop in both SDR (0.37 dB and 0.47 dB for the 1024 and 2048 hidden units, respectively) and STOI (0.009 and 0.0114). Starting from there, BNN catches up with the performance of the baseline system with binarized input by a margin 0.45 dB and 0.29 dB for the 1024 and 2048 units, respectively. The 2048×2 BNN lost 0.0045, but 1024×2 happens to improve STOI by 0.0029.

Overall, we see that the proposed BNN that is fully binarized from its input to the output shows reasonable performance compared to its corresponding real-valued networks. We believe that our experiments on the Fourier spectra and IBM prove the concept well enough and are ready to be extended to the other types of features, target variables, and the choice of context window, since the QaD technique and the BNN training methods work for general purposes.

5. CONCLUSION

In this paper we proposed a novel bitwise source separation system by employing BNNs, which redefined the feedforward pass in a bitwise fashion. A two-stage training strategy was introduced to prepare a set of compressed weights, and then to initialize the BNN parameters that are eventually binarized during the feedforward pass. By binarizing the input magnitude spectra with the QaD technique and having IBM as the target, we showed that BNN performs well for the speech denoising job with a minimal computational cost.

6. REFERENCES

[1] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, "An experimental study on speech enhancement based on deep neural networks," *IEEE Signal Processing Letters*, vol. 21, no. 1, pp. 65–68, 2014.

[2] P. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis, "Joint optimization of masks and deep recurrent neural networks for monaural source separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 12, pp. 2136–2147, Dec 2015.

[3] Y. Wang and D. L. Wang, "Towards scaling up classification-based speech separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 7, pp. 1381–1390, July 2013.

[4] H. Erdogan, J. R. Hershey, S. Watanabe, and J. Le Roux, "Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2015.

[5] Felix Weninger, Hakan Erdogan, Shinji Watanabe, Emmanuel Vincent, Jonathan Le Roux, John R. Hershey, and Björn Schuller, "Speech enhancement with lstm recurrent neural networks and its application to noise-robust asr," in *Proceedings of the International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA)*, Aug. 2015.

[6] Jonathan Le Roux, John R. Hershey, and Felix Weninger, "Deep NMF for speech separation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Apr. 2015.

[7] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[8] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, pp. 788–791, 1999.

[9] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in Neural Information Processing Systems (NIPS)*. 2001, vol. 13, MIT Press.

[10] M. Kim and P. Smaragdis, "Bitwise neural networks," in *International Conference on Machine Learning (ICML) Workshop on Resource-Efficient Machine Learning*, Jul 2015.

[11] W. S. McCulloch and W. H. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[12] L. Pitt and L. G. Valiant, "Computational limitations on learning from examples," *Journal of the Association for Computing Machinery*, vol. 35, pp. 965–984, 1988.

[13] M. Golea, M. Marchand, and T. R Hancock, "On learning μ-perceptron networks with binary weights.," in *Advances in Neural Information Processing Systems (NIPS)*, 1992, pp. 591–598.

[14] D. Soudry, I. Hubara, and R. Meir, "Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights," in *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[15] 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 (NIPS)*, 2015, pp. 3105–3113.

[16] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 4107–4115.

[17] E. Fiesler, A. Choudry, and H. J. Caulfield, "Weight discretization paradigm for optical neural networks," in *The Hague'90, 12-16 April*. International Society for Optics and Photonics, 1990, pp. 164–173.

[18] K. Hwang and W. Sung, "Fixed-point feedforward deep neural network design using weights +1, 0, and -1," in *2014 IEEE Workshop on Signal Processing Systems (SiPS)*, Oct 2014.

[19] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," 2016.

[20] G. Govindu, L. Zhuo, S. Choi, and V. Prasanna, "Analysis of high-performance floating-point arithmetic on fpgas," in *IEEE 18th International Parallel and Distributed Processing Symposium*, 2004.

[21] M. J. Beauchamp, S. Hauck, K. D. Underwood, and K. S. Hemmert, "Embedded floating-point units in fpgas," in *ACM/SIGDA 14th international symposium on Field programmable gate arrays*, 2006, pp. 12–20.

[22] S.P. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, Mar 1982.

[23] A. Narayanan and D. L. Wang, "Ideal ratio mask estimation using deep neural networks for robust speech recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2013, pp. 7092–7096.

[24] Z. Duan, G. J. Mysore, and P. Smaragdis, "Online PLCA for real-time semi-supervised source separation," in *Proceedings of the International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA)*, 2012, pp. 34–41.

[25] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.

[26] E. Vincent, C. Fevotte, and R. Gribonval, "Performance measurement in blind audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 4, pp. 1462–1469, 2006.

[27] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "A short-time objective intelligibility measure for time-frequency weighted noisy speech," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2010.