Fixed-Point Feedforward Deep Neural Network Design Using Weights +1, 0, and -1

Kyuyeon Hwang and Wonyong Sung
Department of Electrical Engineering and Computer Science
Seoul National University
Seoul 151-744, South Korea
Email: khwang@dsp.snu.ac.kr; wysung@snu.ac.kr

***Abstract*—Feedforward deep neural networks that employ multiple hidden layers show high performance in many applications, but they demand complex hardware for implementation. The hardware complexity can be much lowered by minimizing the word-length of weights and signals, but direct quantization for fixed-point network design does not yield good results. We optimize the fixed-point design by employing backpropagation based retraining. The designed fixed-point networks with ternary weights (+1, 0, and -1) and 3-bit signal show only negligible performance loss when compared to the floating-point counterparts. The backpropagation for retraining uses quantized weights and fixed-point signal to compute the output, but utilizes high precision values for adapting the networks. A character recognition and a phoneme recognition examples are presented.**

I. Introduction

Feedforward deep neural networks (DNNs) with multiple hidden layers have outperformed conventional machine learning algorithms in various fields including speech and image recognition [1]–[3]. Implementation of deep neural networks using VLSI or embedded computing systems is needed for real-time and low-power applications. However, high performance DNN algorithms contain many weights. For example, a DNN for acoustic modeling of speech recognition employs about 20 million weights [3]. VLSI based implementations demand a large chip size for storing the weights, while software based applications suffer from large memory accesses. It is, therefore, very important for efficient implementations to reduce the word-length of weights and internal signals.

In a general feedforward deep neural network with multiple hidden layers as depicted in Fig. 1, each layer k has a signal vector $\mathbf{y}_k$, which is propagated to the next layer by multiplying the weight matrix $\mathbf{W}_{k+1}$, adding biases $\mathbf{b}_{k+1}$, and applying the activation function $\phi_{k+1}(\cdot)$ as follows:

$$\mathbf{y}_{k+1} = \phi_{k+1}\left(\mathbf{W}_{k+1}\,\mathbf{y}_k + \mathbf{b}_{k+1}\right). \tag{1}$$

One of the most general activation functions is the logistic sigmoid function defined as

$$\sigma(x) = \frac{1}{1+e^{-x}}. \tag{2}$$

In fully-connected feedforward deep neural networks, each weight matrix between two layers demands $N_1 \times N_2$ weights, where N_1 and N_2 are the number of units for the anterior layer and the posterior layer, respectively. Considering a network employing hidden layers with 1,024 units, each hidden layer demands about one mega weights. The number of output signals and that of biases are both N_2. The signal word-length affects the complexity of arithmetic units and interconnection networks.

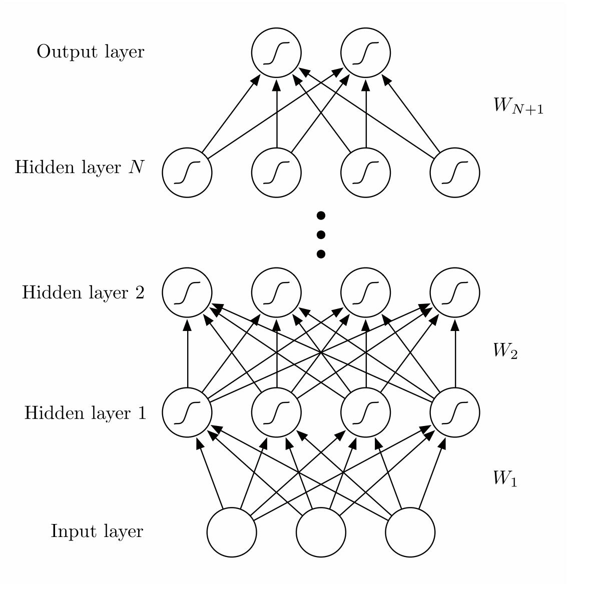

Fig. 1. Feedforward deep neural network with N hidden layers.

There have been many studies about efficient hardware implementation of neural networks [4]–[8]. Instead of direct weight quantization, retraining with backpropagation was developed in [7], [8]. The designed networks employed usually 3-8 bits for the weights and represented the signals in analog or high precision fixed points using more than 7 bits. Recently, a research work that tries to increase the sparseness of the weights by pruning out small valued ones has been developed in order to reduce the model size and the execution time with a CPU [9].

In this paper, we propose a high performance fixed-point optimization method that can greatly reduce the word-length of weights and signals for implementing DNNs. The proposed scheme allows design of DNNs for real-world problems only with ternary (+1, 0, and -1) weights and 2 or 3 bits of fixed-point signals. The developed training algorithm also retrains fixed-point networks using backpropagation but employs several effective and practical techniques, such as elaborate signal grouping through range and sensitivity analysis, quantization of both weights and signals, optimum quantization parameter search, and consideration of deep neural networks. We also examine the performance of fixed-point DNNs with dropout [10], [11], which is a recently developed strong regularization technique. The early-version of the proposed fixed point optimization method was successfully used in our single-chip VLSI implementation of a DNN [12]. In this hardware, weights are extremely quantized to only have three values (+1, 0, and -1) to reduce the size of memory and simplify processing units.

The paper is organized as follows. In Section II, we describe a direct quantization approach as a baseline. Section III contains the proposed scheme that retrains the network after fixed-point quantization. Both the direct and the proposed quantization schemes are evaluated in Section IV. Concluding remarks follow in Section V.

II. Direct Quantization with Exhaustive Search

A deep neural network usually contains millions of weights and thousands of internal signals. Since applying a different data format for each weight or signal is too complex, it is needed to group them according to their range and the quantization sensitivity [13]. In a deep neural network with several layers, it is convenient to separate each layer for the grouping. Among the weights in each layer, we notice that the biases need to have high precision because their range is usually much larger than that of other weights. Assigning a high precision fixed-point format, such as 8 bits, to the biases does not increase the hardware complexity much because the number of them is small. The quantization sensitivity can also be determined from simulations that apply quantized weights for a specific group while using the floating-point data type for other groups [13]. We found that the quantization sensitivity of signals in the hidden layers is mostly the same and very low, but that in the input of the network depends on applications very much.

Uniform quantizers depicted in Fig. 2 are used for representing weights except for the biases. Since each layer has its own uniform quantizer, the hardware or software based implementations can be conducted mostly using addition or subtraction operations.

Once the number of quantization points for each weight matrix is given, the goal is to minimize the output error of the network. A conventional approach is directly quantizing the trained floating-point weights by finding the optimum step size, Δ. The floating-point weights are obtained by employing unsupervised greedy learning with restricted Boltzmann machines (RBMs) as pre-training followed by fine-tuning with error backpropagation [2]. During RBM learning, a penalty term with L_2 weight cost is applied to prevent weights from becoming too large.

In the direct quantization method, determining the optimum step size, Δ, is however a very difficult problem because there is no clear relation between the parameters and the final output errors resulted in by the quantization. Therefore, the optimum step size is initially determined by using an L_2-error minimizing approach that is similar to Lloyd-Max quantization, and then the quantization step size is fine tuned by using exhaustive search. See Appendix for the details of the L_2-error minimizing approach.

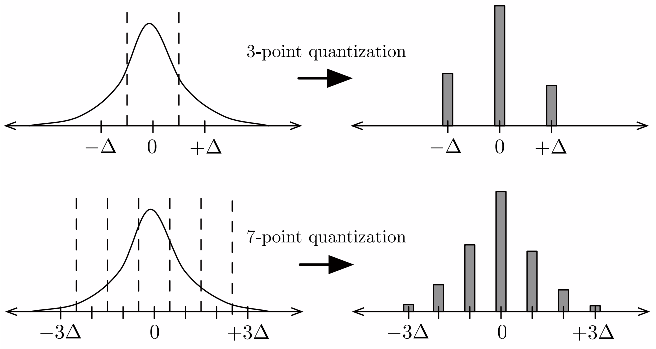

Fig. 2. Histograms of weights before and after applying 3-point and 7-point symmetric uniform quantization where Δ denotes a quantization step size.

To reduce the search dimension, the greedy approach is applied as follows:

1) Prepare a fully trained floating-point weights.
2) Quantize all input data and signals of hidden layers.
3) Starts with the weight quantizer between the input layer and the first hidden layer, try several step sizes around the initial step size and measure the output error of the network with the training set. The initial step size is determined using the L_2-error minimizing approach.
4) Choose the step size that minimizes the output error and quantize the weights.
5) Perform the third and fourth steps for the next layer until it reaches the last layer.

The output error can be the mean cross-entropy for classification or the mean squared error for regression. We usually obtain the best result when Δ is about 1.2-1.6 times of the value determined by the Lloyd-Max algorithm.

III. Retrain with Error Backpropagation on Quantized Domain

As discussed in Section IV, the direct quantization approach results in high output error when the precision of weights is very low, such as 3 or 7 levels. To address this issue, we use a fixed-point optimization scheme that retrains the quantized neural network. This method reapplies the error backpropagation that is modified to deal with quantized weights and signals.

The error backpropagation is basically a gradient descent method that minimizes the output error, where the error gradient is calculated by propagating the error signal δ backward from the output units to the input units. However, applying this algorithm directly to neural networks with quantized weights usually does not work. This is because the amount of weights to be changed after each step is much smaller than the quantization step size, and most of the weight changes are ignored. Therefore, we need to modify the backpropagation algorithm to properly accumulate the small amount of weight changes. For this, we maintain both the high-precision and low-precision weights and signals to accumulate the effects of small adaptation error.

The mini-batch based backpropagation algorithm [14] updates the weight w_{ij}, the synaptic strength from the unit j to the unit i, by

$$w_{ij,new} = w_{ij} - \alpha \left\langle \frac{\partial E}{\partial w_{ij}} \right\rangle = w_{ij} + \alpha \left\langle \delta_i y_j \right\rangle \tag{3}$$

where E is the output error, α is the learning rate, δ_i is the error signal of the unit i, y_j is the output signal of the unit j, and $< \cdot >$ averages the value over a mini-batch. Note that (3) cannot be directly applied to update the low-precision weights because the amount of update, $\alpha \left\langle \delta_i y_j \right\rangle$, is much smaller than the quantization step size, Δ. Thus, we also store high precision weights for adaptation. The high precision weights are used for accumulating errors and generating quantized ones. The low-precision weights are obtained by quantizing the high-precision weights and used in the forward and backward steps of the backpropagation algorithm.

We further modify the backpropagation algorithm to quantize the signals or the outputs of the units. When propagating the error signals backward, the derivative of the activation function is multiplied to the incoming error signals. For the sigmoid activation function in (2), the derivative is usually calculated by

$$\phi'(x) = \phi(x)(1 - \phi(x)) = y(1 - y) \tag{4}$$

where $\phi(x)$ is the activation function and $y = \phi(x)$ is the output signal of the unit. However, using the quantized signal for y in (4) often fails especially with the small number of quantization points. For example, when the binary quantizer is used for the signals, the derivative of the activation function is always zero. Then, no error signals can be propagated backward because they are multiplied by the derivative, which is always 0. Therefore, the derivative should be calculated using high-precision signal values. The overall algorithm is summarized in Fig. 3.

Definition of error signal:

$$\delta_i = -\frac{\partial E}{\partial net_i}$$

Forward step:

$$net_i = \sum_{j \in A_i} w_{ij}^{(q)} y_j^{(q)}$$

$$y_i^{(q)} = R_i \left(\phi_i \left(net_i \right) \right)$$

Backward step:

$$\delta_j = \phi_j' \left(net_j \right) \sum_{i \in P_j} \delta_i w_{ij}^{(q)}$$

Gradient calculation:

$$\frac{\partial E}{\partial w_{ij}} = -\delta_i y_j^{(q)}$$

Weight update:

$$w_{ij,new} = w_{ij} - \alpha \left\langle \frac{\partial E}{\partial w_{ij}} \right\rangle$$

$$w_{ij,new}^{(q)} = Q_{ij} \left(w_{ij,new} \right)$$

Fig. 3. Summarization of the proposed retrain algorithm, where E is the output error, net_i is the summed input value of unit i, δ_i is the error signal of unit i, w_{ij} is the weight from unit j to unit i, y_j is the output signal of unit j, α is the learning rate, A_i is the set of units anterior to unit i, P_j is the set of units posterior to unit j, $R(\cdot)$ is the signal quantizer, $Q(\cdot)$ is the weight quantizer, and $\phi(\cdot)$ is the activation function. The superscript (q) indicates that the value is quantized and $< \cdot >$ averages the value over a mini-batch. Biases can be regarded as weights from the constant input value of 1.

IV. Evaluation

The proposed quantization strategy is evaluated with two neural network examples: handwritten digit recognition and phoneme recognition. We also examine the effects of dropout [10], [11], which is known as very strong regularization technique, and show that dropout can be used together to further increases the performance of the quantized neural network. In each experiment, the input signal is quantized with fixed 8-bit word-length. Output signals of the networks are not quantized because they barely increase the hardware complexity. For every logistic unit, the quantized output is equally divided between 0 and 1. For example, if the word-length is 2 bits, the quantization points are 0/3, 1/3, 2/3, and 3/3. However, signals of linear units are not bounded and their quantization range should be determined carefully. In our phoneme recognition example, each component of the input data is normalized to have zero mean and unit variance over the training set. The input range is chosen to be from -3 to 3.

A. Handwritten Digit Recognition

1) MNIST Database: The MNIST database consists of 28 by 28 grey level images of handwritten digits. A training set has 60,000 examples and a test set has 10,000 examples. This database has been widely used for evaluation of various image classifiers [2], [15], [16].

2) Neural Network Configuration: The neural network configuration is the same as in [2]. The input layer has 784 units, which is followed by two 500-unit and one 2,000-unit hidden layers. The output layer has 10 units, which correspond to 10 target digit labels. All layers contain logistic units.

3) Training: The network is pre-trained with unsupervised greedy RBM learning. Each RBM is trained by 50 epochs of 10-step contrastive-divergence based stochastic gradient descent with the mini-batch size of 100, the fixed learning rate of 0.1, and the momentum of 0.9. Then, we ran 100 epochs of the backpropagation with stochastic gradient descent using

TABLE I
Miss classification rate (%) on the test set with the MNIST handwritten digit recognition example using M-point weight quantization.

Approach	Signal word-length	$M = 3$	$M = 7$	$M = 15$
Direct	1 bit	7.60	1.72	1.38
	2 bits	4.85	1.28	1.06
	3 bits	4.28	1.20	0.99
	8 bits	4.20	1.21	0.95
Retrain	1 bit	1.45	1.27	1.10
	2 bits	1.11	0.99	1.00
	3 bits	1.08	0.95	0.94
	8 bits	1.11	0.95	0.95

TABLE II
Weight distribution of the MNIST handwritten digit recognition example before and after retraining with 3-point weight quantization and 3-bit signal quantization.

Weight distribution (%)		After retraining			Sum
		$-\Delta$	0	$+\Delta$	
Before retraining	$-\Delta$	7.7	0.4	0.0	8.1
	0	1.2	85.2	1.2	87.6
	$+\Delta$	0.0	0.2	4.1	4.3
Sum		8.9	85.8	5.3	100.0

the mini-batch size of 100, the fixed learning rate of 0.1, and the momentum of 0.9. Also, the same parameters are used for the proposed retraining algorithm.

4) Experimental Results: The experimental results with various weight and signal quantization are summarized in TABLE I. The original miss classification rate for the test set was 0.97% with floating-point arithmetic. The direct quantization shows a miss rate of 4.28% with 3-point weights, and 1.20% with 7-point weights. On the other hand, the retraining approach shows the result that is quite close to the original one. The miss rate with 3-point weights and 3-bit signal quantization is 1.08%. Applying 8-bit signal quantization does not show significant difference compared to 3-bit signal quantization, which means 3 bits for signal word-length is enough for hidden layers. Even 2-bit quantization for signals does not show much increase in the miss classification rates when the developed retrain algorithm is used. From this table, we can notice that the retrain algorithm helps reducing not only the word-length of weights but also that of signals.

Distribution of all the weights in the network before and after retraining is shown in TABLE II in the case of 3-point weight and 3-bit signal quantization. Note that weights are changed in both direction, that is, some weights are changed from 0 to Δ whereas other weights are changed in the opposite direction. This clearly shows that fixed-point optimization with retraining is not just adjustment of the quantization boundaries, but slightly moving weights around the quantization boundaries in both directions.

TABLE III
Frame-level phone error rate (%) on the test set with the TIMIT phoneme recognition example using M-point weight quantization.

Approach	Signal word-length	$M = 3$	$M = 7$	$M = 15$
Direct	1 bit	66.58	46.53	38.34
	2 bits	56.15	34.56	30.44
	3 bits	54.10	33.36	28.85
	8 bits	50.20	32.85	28.55
Retrain	1 bit	29.97	29.76	29.67
	2 bits	28.35	28.46	28.02
	3 bits	27.63	27.90	27.73
	8 bits	27.37	27.87	27.84

B. Phoneme Recognition

1) Speech Database: We used the TIMIT corpus that contains a training set from 462 speakers and a test set from 168 speakers. All SA recordings, utterances of the same sentences from every speakers, were removed since they might bias the results. We used 12th-order Mel-frequency cepstral coefficients (MFCCs), energy, and their first and second temporal derivatives, which are extracted using 25-ms Hamming window with 10-ms frame rate. Each component of the feature vectors is normalized over the training set to have zero mean and unit variance. For the input to the neural network, 11 consecutive frames are used, which results in total 429 dimensional input features [17]. For evaluation, the original 61 phones are folded into standard 39 classes as proposed in [18].

2) Neural Network Configuration: The input layer consists of 429 linear units to deal with real value inputs. Four hidden layers have the same number of 1,024 logistic units. The output layer consists of 61 logistic units that correspond to 61 target phoneme labels. For simplicity, no other optimizations nor a language model is used. A similar structure with Gaussian input units, softmax output units, three-state monophone classes, and a bigram language model over phones can be found in [17].

3) Training: The network is pre-trained with unsupervised greedy RBM learning. Each RBM except the linear-logistic RBM is trained by 20 epochs of 1-step contrastive-divergence based stochastic gradient descent with the mini-batch size of 128, the fixed learning rate of 0.05, and the momentum of 0.9. For the linear-logistic RBM, we used 40 epochs with the fixed learning rate of 0.005. For fine-tuning, we ran 10 epochs of the backpropagation with stochastic gradient descent with the mini-batch size of 128, the fixed learning rate of 0.05, and the momentum of 0.9, which are the same for the retraining algorithm.

4) Experimental Results: The original frame error rate of phonemes was 26.24% before any quantization. As in TABLE III, the proposed approach results in very close performance to the original one, even when the network employs only 3-point weights and 3-bit signals for the hidden layers. This shows that two bits are sufficient to represent a single

TABLE IV

COMPARISON OF FRAME-LEVEL PHONE ERROR RATES AND THE PERCENTAGES OF NONZERO WEIGHTS AFTER QUANTIZATION WITH VARYING NUMBER OF UNITS PER HIDDEN LAYER WHEN 3-POINT WEIGHT AND 3-BIT SIGNAL QUANTIZATION IS APPLIED.

Hidden layer size	Frame-level phone error rate (%)		Nonzeros (%)
	Floating-point	Fixed-point	
128	30.38	37.35 (+6.97)	56.26
256	28.19	32.53 (+4.34)	47.56
512	26.84	28.91 (+2.07)	40.04
1024	26.24	27.63 (+1.39)	37.19

TABLE V

COMPARISON OF FRAME-LEVEL PHONE ERROR RATES WITH AND WITHOUT DROPOUT WHEN THE HIDDEN LAYER SIZE IS 1024. FOR FIXED-POINT EXPERIMENTS, 3-POINT WEIGHT AND 3-BIT SIGNAL QUANTIZATION IS APPLIED.

Dropout	Frame-level phone error rate (%)	
	Floating-point	Fixed-point
No	26.24	27.63 (+1.39)
Yes	22.04	22.70 (+0.66)

weight in neural networks of which application is as complex as the phoneme recognition.

TABLE IV compares the performance of floating-point and fixed-point DNNs for phoneme recognition when the number of units in each hidden layer varies from 128 to 1024. For DNNs of hidden layer size 128 and 256, we ran 30 and 20 epochs, respectively, for both fine-tuning and retraining thanks to less severe overfitting. The fixed-point networks employ 3-level weights and 3-bit signal representation. Note that deep neural networks usually employ a relatively large number of units in each layer compared to shallow neural networks. We can find that the gap between the floating-point and fixed-point networks rapidly shrinks as the number of units in each layer increases. The results are not surprising because summing many signals from the previous layer helps lowering the effects of quantization errors. We consider that increasing the dimensionality of a network contributes to correctly identifying patterns when the weights are severely quantized. Also, the percentage of nonzero weights decreases when the hidden layers become larger. The sparseness of weights can allow further optimization in hardware and software based implementations.

5) Experimental Results with Dropout: Dropout is applied to prevent the overfitting problem, which usually observed when training large-sized neural networks with not enough training data. For floating-point training, we applied 20% dropout for the input layer 40% dropout for the hidden layers with the increased learning rate of 0.02. The floating-point training is terminated after 200 epochs of backpropagation. For the fixed-point retraining, we reduce the number of epochs to 50, and the dropout rate to 0% and 20% for the input and the hidden layers, respectively. Overfitting is hardly observed in both floating-point training and fixed-point retraining stage. The frame-level phone error rates of resulting floating-point and fixed-point networks are shown in TABLE V when the hidden layer size is 1024. With dropout, the error rate increases only 0.66%, which is smaller than the result when dropout is not used (+1.39%). This is because dropout also prevents overfitting in retraining stage. Therefore, with proper dropout rates, the proposed fixed-point optimization method can be efficiently used with dropout.

V. CONCLUDING REMARKS

We have developed a training procedure to reduce the word-length of weights and that of signals in deep neural networks. The proposed procedure yields superior results compared to the direct quantization method, especially when only 3-point (+1, 0, and -1) weights are used. The signal word-length that affects the complexity of interconnection and arithmetic units can also be reduced to 3 bits without sacrificing the performance much. We find that the performance gap between the floating-point and fixed-point networks shrinks as the number of units in each layer increases. Also it is shown that dropout can be employed together to generalize the network and further increase the performance. This research is useful for not only hardware based implementations but also real-time software development.

APPENDIX

The L_2-error minimization approach is similar to Lloyd-Max quantization except the quantizer is uniform. To apply Lloyd-Max algorithm to the uniform quantizer, we first divide the quantization function $Q(x)$ into two functions using the integer membership z as

$$Q(x) = f(z) \tag{5}$$

$$z = g(x). \tag{6}$$

For M-point symmetric uniform quantizer with the quantization step size Δ and the odd integer M,

$$f(z) = \Delta \cdot z \tag{7}$$

$$g(x) = \mathrm{sgn}(x) \cdot \min\left\{ \left\lfloor \frac{|x|}{\Delta} + 0.5 \right\rfloor, \frac{M-1}{2} \right\}. \tag{8}$$

Before performing the exhaustive search for the weight quantizers, the quantization step size Δ is determined to minimize the error,

$$E = \frac{1}{2}\sum_{i=1}^{N}(Q(x_i) - x_i)^2 = \frac{1}{2}\sum_{i=1}^{N}(\Delta \cdot z_i - x_i)^2 \tag{9}$$

where N is the number of weight coefficients, x_i is the original i-th weight value, and z_i is the membership of x_i determined by the quantizer.

The error can be minimized by the following two-step iterative approach, which is similar to Lloyd-Max algorithm.

$$\mathbf{z}^{(t)} = \arg\min_{\mathbf{z}} E(\mathbf{x}, \mathbf{z}, \Delta^{(t-1)}) \tag{10}$$

$$\Delta^{(t)} = \arg\min_{\boldsymbol{\Delta}} E(\mathbf{x}, \mathbf{z}^{(t)}, \Delta) \tag{11}$$

where the superscript (t) indicates the iteration step. By Lloyd-Max algorithm, the first step is equivalent to $\mathbf{z}^{(t)} = g^{(t-1)}(\mathbf{x})$ where $g^{(t-1)}(\mathbf{x})$ is the vector version of $g(x)$ in (8) with the previous step size $\Delta^{(t-1)}$. The second equation can be solved by letting the derivative of the error with respect to $\Delta^{(t)}$ to zero, which leads to

$$\Delta^{(t)} = \frac{\sum_{i=1}^{N} x_i z_i^{(t)}}{\sum_{i=1}^{N} \left(z_i^{(t)}\right)^2}. \tag{12}$$

The iteration stops when $\Delta^{(t)}$ converges to one of the local minima.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2012R1A2A2A06047297).

References

[1] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[2] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[3] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 1, pp. 30–42, 2012.

[4] P. Moerland and E. Fiesler, "Neural network adaptations to hardware implementations," *Handbook of neural computation*, vol. 1, p. 2, 1997.

[5] J. Holi and J.-N. Hwang, "Finite precision error analysis of neural network hardware implementations," *Computers, IEEE Transactions on*, vol. 42, no. 3, pp. 281–290, 1993.

[6] G. Dundar and K. Rose, "The effects of quantization on multilayer neural networks," *Neural Networks, IEEE Transactions on*, vol. 6, no. 6, pp. 1446–1451, 1995.

[7] 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.

[8] C. Z. Tang and H. K. Kwan, "Multilayer feedforward neural networks with single powers-of-two weights," *Signal Processing, IEEE Transactions on*, vol. 41, no. 8, pp. 2724–2727, 1993.

[9] D. Yu, F. Seide, G. Li, and L. Deng, "Exploiting sparseness in deep neural networks for large vocabulary speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4409–4412.

[10] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[11] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout," in *Proc. ICASSP*, 2013.

[12] J. Kim, K. Hwang, and W. Sung, "x1000 real-time phoneme recognition VLSI using feed-forward deep neural networks," in *Proc. ICASSP*, 2014.

[13] W. Sung and K.-I. Kum, "Simulation-based word-length optimization method for fixed-point digital signal processing systems," *Signal Processing, IEEE Transactions on*, vol. 43, no. 12, pp. 3087–3090, 1995.

[14] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural Networks, 1989. IJCNN., International Joint Conference on*. IEEE, 1989, pp. 593–605.

[15] L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, L. D. Jackel, Y. LeCun, U. A. Muller, E. Sackinger, P. Simard *et al.*, "Comparison of classifier methods: a case study in handwritten digit recognition," in *Pattern Recognition, 1994. Vol. 2-Conference B: Computer Vision & Image Processing., Proceedings of the 12th IAPR International. Conference on*, vol. 2. IEEE, 1994, pp. 77–82.

[16] K. Labusch, E. Barth, and T. Martinetz, "Simple method for high-performance digit recognition based on sparse coding," *Neural Networks, IEEE Transactions on*, vol. 19, no. 11, pp. 1985–1989, 2008.

[17] A.-r. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 1, pp. 14–22, 2012.

[18] K.-F. Lee and H.-W. Hon, "Speaker-independent phone recognition using hidden markov models," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 37, no. 11, pp. 1641–1648, 1989.