Resiliency of Deep Neural Networks under Quantization

Article · November 2015

Source: arXiv

3 authors, including:

Wonyong Sung
Seoul National University
207 PUBLICATIONS **3,885** CITATIONS

SEE PROFILE

Sungho Shin
Seoul National University
13 PUBLICATIONS **144** CITATIONS

SEE PROFILE

RESILIENCY OF DEEP NEURAL NETWORKS UNDER QUANTIZATION

Wonyong Sung, Sungho Shin & Kyuyeon Hwang
Department of Electrical and Computer Engineering
Seoul National University
Seoul, 08826 Korea
wysung@snu.ac.kr
shshin@dsp.snu.ac.kr
kyuyeon.hwang@gmail.com

ABSTRACT

The complexity of deep neural network algorithms for hardware implementation can be much lowered by optimizing the word-length of weights and signals. Direct quantization of floating-point weights, however, does not show good performance when the number of bits assigned is small. Retraining of quantized networks has been developed to relieve this problem. In this work, the effects of retraining are analyzed for a feedforward deep neural network (FFDNN) and a convolutional neural network (CNN). The network complexity is controlled to know their effects on the resiliency of quantized networks by retraining. The complexity of the FFDNN is controlled by varying the unit size in each hidden layer and the number of layers, while that of the CNN is done by modifying the feature map configuration. We find that the performance gap between the floating-point and the retrain-based ternary (+1, 0, -1) weight neural networks exists with a fair amount in complexity limited' networks, but the discrepancy almost vanishes in fully complex networks whose capability is limited by the training data, rather than by the number of connections. This research shows that highly complex DNNs have the capability of absorbing the effects of severe weight quantization through retraining, but connection limited networks are less resilient.

1 INTRODUCTION

Deep neural networks begin to find many real-time applications, such as speech recognition, autonomous driving, gesture recognition, and robotic control (Sak et al. (2015); Chen et al. (2015); Jalab et al. (2015); Corradini et al. (2015)). Although most of deep neural networks are implemented using GPUs (Graphics Processing Units) in these days, their implementation in hardware can give many benefits in terms of power consumption and system size (Ovtcharov et al. (2015)). FPGA based implementation examples of CNN show more than 10 times advantage in power consumption (Ovtcharov et al. (2015)).

Neural network algorithms employ many multiply and add (MAC) operations that mimic the operations of biological neurons, which suggests that reconfigurable hardware arrays that contain quite homogeneous hardware blocks, such as MAC units, can give very efficient solution to real-time neural network system design. Our previous works show that the precision required for implementing FFDNN or CNN needs not be very high, especially when the quantized networks are trained again to learn the effects of lowered precision. In the fixed-point optimization examples shown in Hwang & Sung (2014); Anwar et al. (2015), neural networks with ternary weights showed quite good performance which is close to that of floating-point arithmetic, but not always.

In this work, we try to know if retraining can recover the performance of FFDNN and CNN under quantization with only ternary (+1, 0, -1) levels or 3 bits (+3, +2, +1, 0, -1, -2, -3) for weight representation. For this study, the network complexity is changed to analyze their effects on the performance gap between floating-point and low-precision retrained fixed-point deep neural networks.

We conduct our experiments with a feed-forward deep neural network (FFDNN) for phoneme recognition and a convolutional neural network (CNN) for image classification. To control the network size, not only the number of units in each layer but also the number of hidden layers are varied in the FFDNN. For the CNN, the number of feature maps for each layer and the number of layers are both changed. The FFDNN uses the TIMIT corpus and the CNN employs the CIFAR-10 dataset. This analysis intends to find an insight to the knowledge representation capability of highly quantized networks, and also provides a guideline to efficient network size and word-length design for hardware implementation of deep neural networks.

2 RELATED WORK

Fixed-point implementation of signal processing algorithms has long been of interest for VLSI based design of multimedia and communication systems. Some of early works used statistical modeling of quantization noise for application to linear digital filters. The simulation-based word-length optimization method utilized simulation tools to evaluate the fixed-point performance of a system, by which non-linear algorithms can be optimized (Sung & Kum (1995)). Ternary (+1, 0, -1) coefficients based digital filters were used to eliminate multiplications at the cost of higher quantization noise. The implementation of adaptive filters with ternary weights were developed, but it demanded oversampling to remove the quantization effects (Hussain et al. (2007)).

Fixed-point neural network design also has been studied with the same purpose of reducing the hardware implementation cost (Moerland & Fiesler (1997)). In Holt & Baker (1991), back propagation simulation with 16-bit integer arithmetic was conducted for several problems, such as NetTalk, Parity, Protein and so on. The authors conducted the experiments while changing the number of hidden units, which was, however, relatively small numbers. The integer simulations showed quite good results for NetTalk and Parity, but not for Protein benchmarks. With direct quantization of trained weights, this work also confirmed satisfactory operation of neural networks with 8-bit precision. An implementation with ternary weights were reported for neural network design with optical fiber networks (Fiesler et al. (1990)). In this ternary network design, the authors employed retraining after direct quantization to improve the performance of a shallow network.

Recently, fixed-point design of deep neural networks is revisited, and FFDNN and CNN with ternary weights show quite good performance that are very close to the floating-point results. The ternary weight based FFDNN and CNN are used for VLSI and FPGA based implementations, by which the algorithms can be operated with only on-chip memory consuming very low power (Kim et al. (2014)). Binary weight based deep neural network design is also studied (Courbariaux et al. (2015)). Pruned floating-point weights are also utilized for efficient GPU based implementations, where small valued weights are forced to zero to reduce the number of arithmetic operations and the memory space for weight storage (Yu et al. (2012b); Han et al. (2015)). A network restructuring technique using singular value decomposition technique is also studied (Xue et al. (2013); Rigamonti et al. (2013)).

3 FIXED-POINT FFDNN AND CNN DESIGN

In this section, the design of FFDNN and CNN with varying network complexity and the fixed-point optimization procedure are explained.

3.1 FFDNN AND CNN DESIGN

A feedforward deep neural network with multiple hidden layers are depicted in Figure 1. Each layer k has a signal vector y_k, which is propagated to the next layer by multiplying the weight matrix W_{k+1}, adding biases b_{k+1}, and applying the activation function $\phi_{k+1}(\cdot)$ as follows:

$$y_{k+1} = \phi_{k+1}\big(W_{k+1}y_k + b_{k+1}\big). \tag{1}$$

One of the most popular activation functions is the rectified linear unit defined as
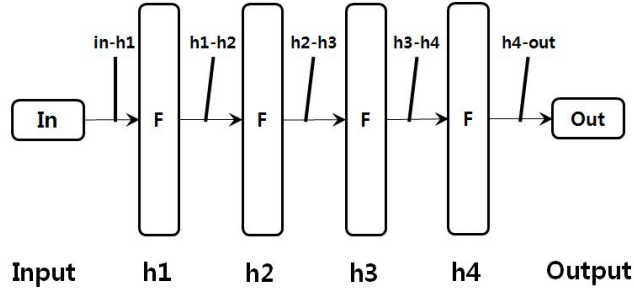
$$Relu(x) = max(0, x). \tag{2}$$

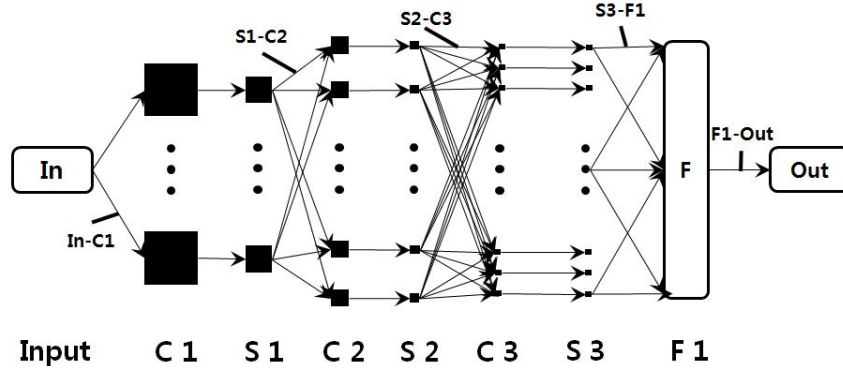Figure 1: Feed-forward deep neural network with 4 hidden layers.

Figure 2: CNN structure with 3 convolution layers and 1 fully-connected layers.

In this work, an FFDNN for phoneme recognition is used. The reference DNN has four hidden layers. Each of the hidden layers has N_h units; the value of N_h is changed to control the complexity of the network. The number of hidden layers is also reduced. We conduct experiments with the N_h size of 32, 64, 128, 256, 512, and 1024. The input layer of the network has 1,353 units to accept 11 frames of a Fourier-transform-based filter-bank with 40 coefficients (+energy) distributed on a mel-scale, together with their first and second temporal derivatives. The output layer consists of 61 softmax units which correspond to 61 target phoneme labels. Phoneme recognition experiments were performed on the TIMIT corpus. The standard 462 speaker set with all SA records removed was used for training, and a separate development set of 50 speaker was used for early stopping. Results are reported for the 24-speaker core test set. The network was trained with backpropagation algorithm with 128 mini-batch size. Initial learning rate was 10^{-5} and it was decreased until 10^{-7} during the training. Momentum was 0.9 and RMSProp was adopted for weights update (Tieleman & Hinton (2012)). The dropout technique was employed with 0.2 dropout rate in each layer.

The CNN used is for CIFAR-10 dataset. It contains a training set of 50,000 and a test set of 10,000 $32{\times}32$ RGB color images representing airplanes, automobiles, birds, cats, deers, dogs, frogs, horses, ships and trucks. We divided the training set to 40,000 images for training and 10,000 images for validation. This CNN has 3 convolution and pooling layers and a fully connected hidden layer with 64 units, and the output has 10 softmax units as shown in Figure 2. We control the number of feature maps in each convolution layer. The reference size has 32-32-64 feature maps as used in Krizhevskey (2014). We didn't preform any preprocessing and data augmentation such as ZCA whitening and global contrast normalization. To know the effects of network size variation, the number of feature maps is reduced or increased. The configurations of the feature maps used for the experiments are 8-8-16, 16-16-32, 32-32-64, 64-64-128, 96-96-192, and 128-128-256. The number of feature map layers is also changed, resulting in 32-32-64, 32-64, and 64 map configurations. Note that the fully connected layer is not changed. The network was trained with a backpropagation algorithm with 128 mini-batch size. Initial learning rate was 0.001 and it was decreased to 10^{-8} during the training procedure. Momentum was 0.8 and RMSProp was applied for weights update.

3.2 Fixed-point optimization of DNNs

Reducing the word-length of weights brings several advantages in hardware based implementation of neural networks. First, it lowers the arithmetic precision, and thereby reduces the number of gates needed for multipliers. Second, the size of memory for storing weights is minimized, which would be a big advantage when keeping them on a chip, instead of external DRAM or NAND flash memory. Note that FFDNNs and recurrent neural networks demand a very large number of weights. Third, the reduced arithmetic precision or minimization of off-chip memory accesses leads to low power consumption. However, we need to concern the quantization effects that degrades the system performance when reducing the word-length.

Direct quantization converts a floating-point value to the closest integer number, which is conventionally used in signal processing system design. However, direct quantization usually demands more than 8 bits, and does not show good performance when the number of bits is small. In fixed-point deep neural network design, retraining of quantized weights shows quite good performance.

The fixed-point DNN algorithm design consists of three steps: floating-point training, direct quantization, and retraining of weights. The floating-point training procedure can be any of the state of the art techniques, which may include unsupervised learning and dropout. Note that fixed-point optimization needs to be based on the best performing floating-point weights. Thus, the floating-point weight optimization may need to be conducted several times with different initializations, and this step consumes the most of the time. After the floating-point training, direct quantization is followed.

For direct quantization, uniform quantization function is employed and the function $Q(\cdot)$ is defined as follows :

$$Q(w) = sgn(w) \cdot \Delta \cdot min\left(\left\lfloor \frac{|(w)|}{\Delta} + 0.5 \right\rfloor, \frac{M-1}{2} \right) \tag{3}$$

where $sgn(\cdot)$ is a sign function, Δ is a quantization step size, and M represents the number of quantization levels. Note that M needs to be an odd number since the weight values can be positive or negative. When M is 7, the weights are represented by $-3\cdot\Delta$, $-2\cdot\Delta$, $-1\cdot\Delta$, 0, $+1\cdot\Delta$, $+2\cdot\Delta$, $+3\cdot\Delta$, which can be represented in 3 bits.
The quantization step size Δ is determined to minimize the L2 error, E, depicted as follows.

$$E = \frac{1}{2} \sum_{i=1}^{N} \big(Q(w_i) - w_i \big)^2 \tag{4}$$

where N is the number of weights in each weight group, w_i is the i-th weight value represented in floating-point. This process needs some iteration, but does not take much time.

For network retraining, we maintain both floating-point and quantized weights because the amount of weight updates in each training step is much smaller than the quantization step size Δ. The forward and backward propagation is conducted using quantized weights, but the weight update is applied to the floating-point weights and newly quantized values are generated at each iteration. This retraining procedure usually converges quickly and does not take much time when compared to the floating-point training.

4 Experiments

4.1 Analysis of direct quantization

The performance of the FFDNN and the CNN with directly quantized weights is analyzed while varying the number of units in each layer or the number of feature maps, respectively. In this analysis, the quantization is performed on each weight group, which is illustrated in Figure1 and 2, to know the sensitivity of word-length reduction. In this sub-section, we try to analyze the effects of direct quantization.

The quantized weight can be represented as follows,
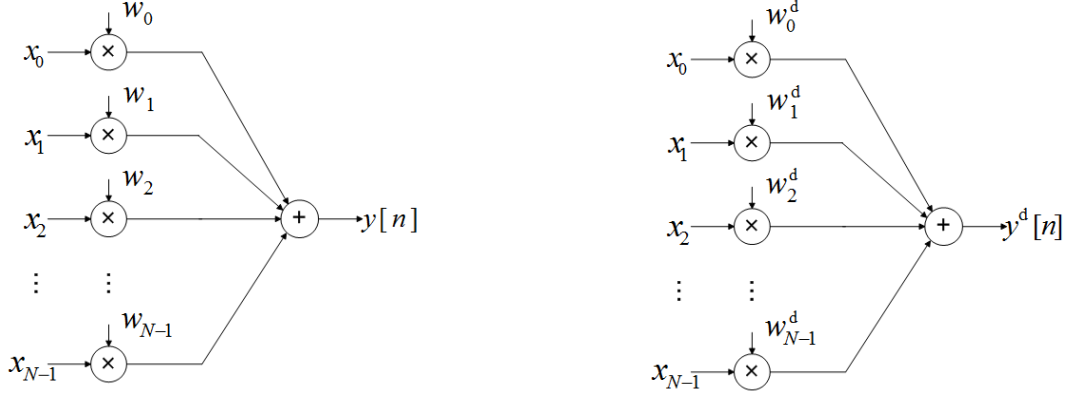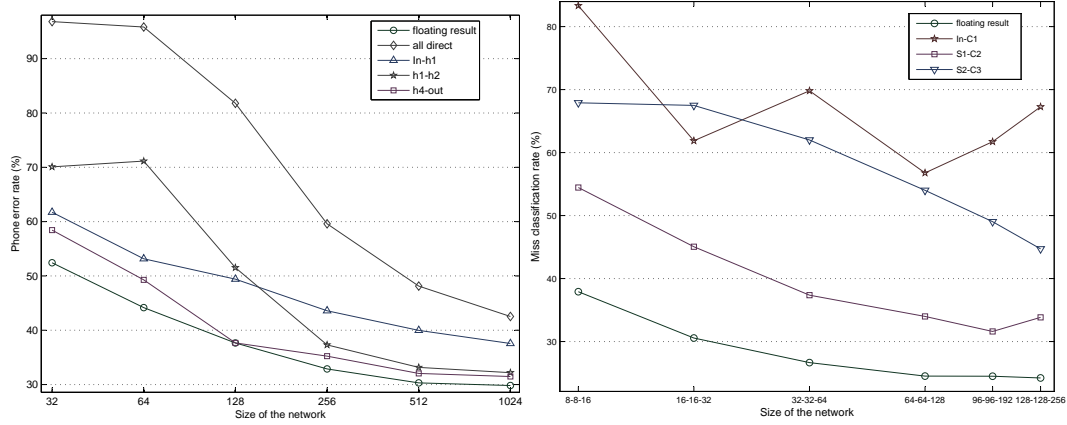
$$w_i^q = w_i + w_i^d \tag{5}$$

Figure 3: Computation procedure for a unit in the hidden layer j.

Figure 4: Sensitivity analysis of direct quantization.

where w_i^d is the distortion of each weight due to quantization. In the direct quantization, we can assume that the distortion w_i^d is not dependent each other.

Consider a computation procedure for a unit in a hidden layer, the signal from the previous layer is summed up after multiplication with the weights as illustrated in Figure3. We can also assemble a model for distortion, which is shown in Figure3. In the distortion model, since w_i^d is independent each other, we can assume that the effects of the summed distortion is reduced according to the random process theory. This analysis means that the effects of the distortion is reduced when the number of units in the anterior layer increases, but slowly.

Figure4 illustrates the performance of the FFDNN with floating-point arithmetic, 2-bit direct quantization of all the weights, and 2-bit direct quantization only on the weight group 'In-h1', 'h1-h2', and 'h4-out'. Consider the quantization performance of the 'In-h1' layer, the phone-error rate is higher than the floating-point result with an almost constant amount, about 10%. Note that the number of input to the 'In-h1' layer is fixed, 1353, regardless of the hidden unit size. Thus, the amount of distortion delivered to each unit of the hidden layer 1 can be considered unchanged. Figure4 also shows the quantization performance on 'h1-h2', 'h3-h4', and 'h4-out' layers, which informs the trend of reduced gap to the floating-point performance as the network size increases. This can be explained by the sum of increased number of independent distortions when the network size grows. The performance of all 2-bit quantization also shows the similar trend of reduced gap to the floating-point performance. But, apparently, the performance of 2-bit directly quantized networks are not satisfactory.
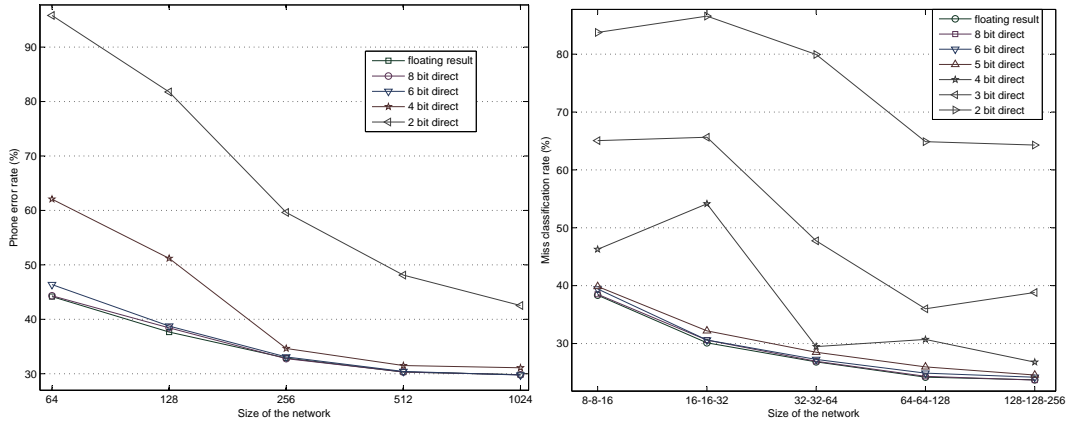
5

Figure 5: Performance of direct quantization with multiple precision.

A similar analysis is conducted to the CNN with direct quantization when the number of feature maps increases or decreases. In the CNN, the number of input to each output is determined by the number of input feature maps and the kernel size. For example, at the first layer C1, the number of input signal for computing one output is only 75 (=3×25) regardless of the network size, where the input map size is always 3 and the kernel size is 25. However, at the second layer C2, the number of input feature maps increases as the network size grows. When the feature map of 32-32-64 is considered, the number of input for the C2 layer grows to 800 (=32×25). Thus, we can expect a reduced distortion as the number of feature maps increases.

Figure5 shows the performance of direct quantization with 2, 4, 6, and 8-bit precision when the network complexity varies. In the FFDNN, 6 bit direct quantization seems enough when the size of the network is larger than 128. But, small FFDNNs demand 8 bits for near floating-point performance. The CNN in Figure5 also shows the similar trend. The direct quantization requires about 6 bits when the feature map configuration is 16-16-32 or larger.

4.2 EFFECTS OF RETRAINING ON QUANTIZED NETWORKS

In order to make the DNN to learn the effects of quantization, we conduct retraining on the directly quantized networks using the same data used for floating-point training. The fixed-point performance of the FFDNN is shown in Figure6 when the number of hidden units in each layer varies. The performance of direct 2 bits (ternary levels), direct 3 bits (7-levels), retrain-based 2 bits, and retrain-based 3 bits are compared with the floating-point simulation. We can find that the performance gap between the floating-point and the retrain-based fixed-point networks converges very fast as the network size grows. Although the performance gap between the direct and the floating-point networks also converges, the rate of convergence is significantly different. In this figure, the performance of the floating-point network almost saturates when the network size is about 1024. Note that the TIMIT corpus that is used for training has only 3 hours of data. Thus, the network with 1024 hidden units can be considered in the 'training-data limited region'. Here, the gap between the floating-point and fixed-point networks almost vanishes when the network is in the 'training-data limited region'. However, when the network size is limited, such as 32, 64, 128, or 256, there is some performance gap between the floating-point and highly quantized networks even if retraining on the quantized networks is performed.

The similar experiments are conducted for the CNN with varying feature map sizes and the results are shown in Figure7. The configuration of the feature maps used for the experiments are 8-8-16, 16-16-32, 32-32-64, 64-64-128, 96-96-192, and 128-128-256. The size of the fully connected layer is not changed. In this figure, the floating-point and the fixed-point performances with retraining also converges very fast as the number of feature maps increases. The floating-point performance saturates when the feature map size is 128-128-256, and the gap is less than 1% when comparing the floating-point and the retrain-based 2-bit networks. However, also, there is some performance gap
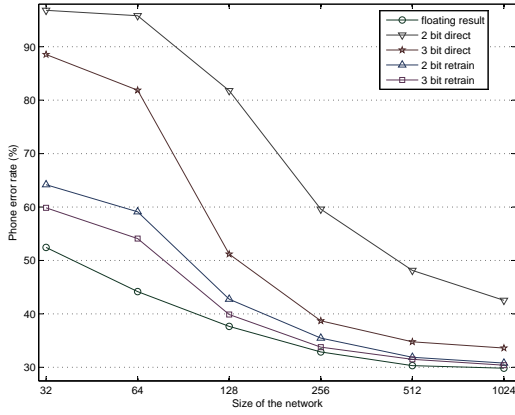
6

Figure 6: Comparison of retrain-based and direct quantization for DNN. All the weights are quantized with ternary and 7-level weights.
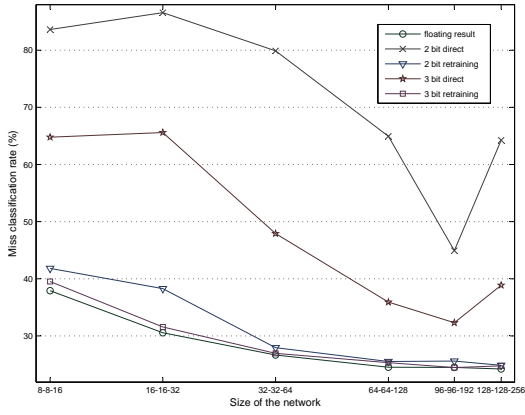
Figure 7: Comparison of retrain-based and direct quantization for CNN. All the weights are quantized with ternary and 7-level weights.

when the number of feature maps is reduced. This suggests that a fairly high performance feature extraction can be designed even using very low-precision weights if the number of feature maps can be increased.

4.3 FIXED-POINT PERFORMANCES WHEN VARYING THE DEPTH

It is well known that increasing the depth usually results in positive effects on the performance of a DNN (Yu et al. (2012a)). The network complexity of a DNN is changed by increasing or reducing the number of hidden layers or feature map levels. The result of fixed-point and floating-point performances when varying the number of hidden layers for the FFDNN is summarized in Table 1. This table shows that both the floating-point and the fixed-point performances of the FFDNN increase when adding hidden layers from 0 to 4. The performance gap between the floating-point and the fixed-point networks shrinks as the number of levels increases.

Table 1: Depth change in DNN

Number of layers (Floating-point result)	Quant Level	Direct	Retraining	Difference
4	3-level	48.13%	31.86%	1.55%
(30.31%)	7-level	34.77%	31.49%	1.18%
3	3-level	49.27%	33.05%	2.24%
(30.81%)	7-level	36.58%	31.72%	0.91%
2	3-level	47.74%	33.89%	2.38%
(31.51%)	7-level	36.99%	33.04%	1.53%
1	3-level	69.88%	38.58%	3.91%
(34.67%)	7-level	56.81%	36.57%	1.90%

The network complexity of the CNN is also varied by reducing the level of feature maps as shown in Table 2. As expected, the performance of both the floating-point and retrain-based low-precision networks degrades as the number of levels is reduced. The performance gap between them is very small with 7-level quantization for all feature map levels.

These results for the FFDNN and the CNN with varied number of levels also show that the effects of quantization can be much reduced by retraining when the network contains some redundant complexity.

Table 2: Depth change in CNN

Layer (Floating-point result)	Quant Level	Direct	Retraining	Difference
64	3-level	72.95%	35.37%	1.18%
(34.19%)	7-level	46.60%	34.15%	-0.04%
32-64	3-level	55.30%	29.51%	0.22%
(29.29%)	7-level	39.80%	29.32%	0.03%
32-32-64	3-level	79.88%	27.94%	1.07%
(26.87%)	7-level	47.91%	26.95%	0.08%

5 DISCUSSION

In this study, we control the network size by changing the number of units in the hidden layers, the number of feature maps, or the number of levels. At any case, reduced complexity lowers the resiliency to quantization. We are conducting similar experiments to the recurrent neural networks that are known to be more sensitive to quantization. This work seems to be directly related to several network optimization methods, such as pruning, fault tolerance, and decomposition. In the pruning, retraining of weights is conducted after zeroing small valued weights. The effects of pruning, fault tolerance, and network decomposition efficiency would be dependent on the redundant representation capability of DNNs.

This study can be applied to hardware efficient DNN design. For design with limited hardware resources, when the size of the reference DNN is relatively small, it is advised to employ a very low-precision arithmetic and, instead, increasing the network complexity as much as the hardware capacity allows. But, when the DNNs are in the performance saturation region, increasing the arithmetic precision gains slightly more because growing the 'alread-big' network size brings almost no performance advantages.

6 CONCLUSION

We analyze the performance of fixed-point deep neural networks, an FFDNN for phoneme recognition and a CNN for image classification, while not only changing the arithmetic precision but also varying their network complexity. The low-precision networks for this analysis are obtained by using the retrain based quantization methods, and the network complexity is controlled by changing the configurations of the hidden layers or feature maps. The performance gap between the floating-point and the fixed-point neural networks with ternary weights (+1, 0, -1) almost vanishes when the DNNs are in the performance saturation region for the given training data. However, when the complexity of DNNs are reduced, by lowering either the number of units, feature maps, or hidden layers, the performance gap between them increases. In other words, a large size network that may contain redundant representation capability for the given training data does not hurt by the lowered precision, but a very compact network does.

REFERENCES

Anwar, Sajid, Hwang, Kyuyeon, and Sung, Wonyong. Fixed point optimization of deep convolutional neural networks for object recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pp. 1131–1135. IEEE, 2015.

Chen, Chenyi, Seff, Ari, Kornhauser, Alain, and Xiao, Jianxiong. Deepdriving: Learning affordance for direct perception in autonomous driving. *arXiv preprint arXiv:1505.00256*, 2015.

Corradini, Maria Letizia, Giantomassi, Andrea, Ippoliti, Gianluca, Longhi, Sauro, and Orlando, Giuseppe. Robust control of robot arms via quasi sliding modes and neural networks. In *Advances and Applications in Sliding Mode Control systems*, pp. 79–105. Springer, 2015.

Courbariaux, Matthieu, Bengio, Yoshua, and David, Jean-Pierre. Binaryconnect: Training deep neural networks with binary weights during propagations. *arXiv preprint arXiv:1511.00363*, 2015.

Fiesler, Emile, Choudry, Amar, and Caulfield, H John. Weight discretization paradigm for optical neural networks. In *The Hague'90, 12-16 April*, pp. 164–173. International Society for Optics and Photonics, 1990.

Han, Song, Mao, Huizi, and Dally, William J. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. 2015.

Holt, Jordan L and Baker, Thomas E. Back propagation simulations using limited precision calculations. In *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, volume 2, pp. 121–126. IEEE, 1991.

Hussain, B Zahir M et al. Short word-length lms filtering. In *Signal Processing and Its Applications, 2007. ISSPA 2007. 9th International Symposium on*, pp. 1–4. IEEE, 2007.

Hwang, Kyuyeon and Sung, Wonyong. Fixed-point feedforward deep neural network design using weights +1, 0, and -1. In *Signal Processing Systems (SiPS), 2014 IEEE Workshop on*, pp. 1–6. IEEE, 2014.

Jalab, Hamid A, Omer, Herman, et al. Human computer interface using hand gesture recognition based on neural network. In *Information Technology: Towards New Smart World (NSITNSW), 2015 5th National Symposium on*, pp. 1–6. IEEE, 2015.

Kim, Jonghong, Hwang, Kyuyeon, and Sung, Wonyong. X1000 real-time phoneme recognition VLSI using feed-forward deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pp. 7510–7514. IEEE, 2014.

Krizhevskey, A. CUDA-convnet, 2014.

Moerland, Perry and Fiesler, Emile. Neural network adaptations to hardware implementations. Technical report, IDIAP, 1997.

Ovtcharov, Kalin, Ruwase, Olatunji, Kim, Joo-Young, Fowers, Jeremy, Strauss, Karin, and Chung, Eric S. Accelerating deep convolutional neural networks using specialized hardware. *Microsoft Research Whitepaper*, 2, 2015.

Rigamonti, Roberto, Sironi, Amos, Lepetit, Vincent, and Fua, Pascal. Learning separable filters. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pp. 2754–2761. IEEE, 2013.

Sak, Haşim, Senior, Andrew, Rao, Kanishka, and Beaufays, Françoise. Fast and accurate recurrent neural network acoustic models for speech recognition. *arXiv preprint arXiv:1507.06947*, 2015.

Sung, Wonyong and Kum, Ki-II. Simulation-based word-length optimization method for fixed-point digital signal processing systems. *Signal Processing, IEEE Transactions on*, 43(12):3087–3090, 1995.

Tieleman, Tijmen and Hinton, Geoffrey. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4, 2012.

Xue, Jian, Li, Jinyu, and Gong, Yifan. Restructuring of deep neural network acoustic models with singular value decomposition. In *INTERSPEECH*, pp. 2365–2369, 2013.

Yu, Dong, Deng, Alex Acero, Dahl, George, Seide, Frank, and Li, Gang. More data + deeper model = better accuracy. In *keynote at International Workshop on Statistical Machine Learning for Speech Processing*, 2012a.

Yu, Dong, Seide, Frank, Li, Gang, and Deng, Li. Exploiting sparseness in deep neural networks for large vocabulary speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pp. 4409–4412. IEEE, 2012b.