# HLHLp: Quantized Neural Networks Training for Reaching Flat Minima in Loss Surface

**Sungho Shin,**[1] **Jinhwan Park,**[1] **Yoonho Boo,**[1] **Wonyong Sung**[1]
[1]Department of Electrical and Computer Engineering
Seoul National University
Seoul, 08826 Korea
sungho.develop@gmail.com, wysung@snu.ac.kr

**Abstract**

Quantization of deep neural networks is extremely essential for efficient implementations. Low-precision networks are typically designed to represent original floating-point counterparts with high fidelity, and several elaborate quantization algorithms have been developed. We propose a novel training scheme for quantized neural networks to reach flat minima in the loss surface with the aid of quantization noise. The proposed training scheme employs high-low-high-low precision in an alternating manner for network training. The learning rate is also abruptly changed at each stage for coarse- or fine-tuning. With the proposed training technique, we show quite good performance improvements for convolutional neural networks when compared to the previous fine-tuning based quantization scheme. We achieve the state-of-the-art results for recurrent neural network based language modeling with 2-bit weight and activation.

## 1 Introduction

Deep neural networks (DNNs) are extremely important in various applications. Most DNNs contain a very large number of weights, and their real-time execution typically demands a huge number of operations. In particular, many DNN applications are deployed to mobile and embedded systems that only have limited budgets in terms of computing and memory. Quantized deep neural networks (QDNNs) represent the weights and activations with only a low number of bits as opposed to 32-bit floating-point format to relieve the complexity problem. Specifically, recent studies indicate that most DNN models are over-parameterized (Neyshabur et al. 2018), and they do not necessarily demand 32-bit floating-point arithmetic for full performance (Hwang and Sung 2014).

Many previous QDNN optimization algorithms consist of three steps: training a floating-point network, quantizing the model, and improving the quantized network by fine-tuning. As for the fine-tuning, usually low learning rates are used to limit the deviation from the floating-point model as small as possible (Hubara et al. 2017; Hwang and Sung 2014; Xu et al. 2018; Zhou et al. 2017). However, when only very low-precision weights are employed, the loss surface may differ from that with high precision. Therefore, fine-tuning the QDNN with quantization error feedback is not sufficient to design well-generalized QDNN.

The generalization capability of a DNN has been actively discussed (Hochreiter and Schmidhuber 1997a; Jastrzkebski et al. 2017; Keskar et al. 2017). Hochreiter and Schmidhuber (1997a) related the generalization capability of a DNN with a flat minimum of the error or loss function. The study by Jastrzkebski et al. (2017) reveals that the ratio of learning rate to batch size is a key determinant of flatness of loss surface and generalization. Recent studies schedule the learning rate to improve the generalization capability (Loshchilov and Hutter 2017; Smith 2017).

In this study, we propose a QDNN training algorithm that is intended to avoid sharp minima and reach flat minima in the discrete weight domain. The proposed approach intentionally changes the learning rate and the precision of the parameters in an alternating manner to reach a flat minimum despite abrupt increases in the training error. The experiments exhibit particularly good results in the quantization of parameter-size efficient convolutional neural networks (CNNs) and recurrent neural networks (RNNs). The contributions of the study are as follows:

- We derive that the quantization noise can play a role of escaping sharp minima in the training of QDNN.
- A high-low-high-low precision (HLHLp) training scheme is developed to encourage a QDNN arriving at a flat minimum that exhibits a high generalization capability.
- The proposed method is applied to the quantization of RNNs and CNNs. We achieve the results that significantly exceed those of previous designs for RNNs and competitive results for CNNs.

## 2 Related Works

### 2.1 Quantization of Deep Neural Networks

QDNN has been studied for a long time. However, earlier studies typically employed an 8-bit or higher precision partly because the networks were small and a direct quantization method was used. Hwang and Sung (2014) and Courbariaux, Bengio, and David (2015) successfully quantized

the weights of DNNs to 2-bit ternary or 1-bit binary without significantly affecting the performance. It is difficult to update discrete weights directly because the gradients are much smaller than quantized weight values. Thus, the quantized weights are obtained by the error feedback quantization method. The method retains the high precision weights to accumulate gradients while the quantized weights are used in forward and backward propagation (Courbariaux, Bengio, and David 2015; Hubara et al. 2017; Hwang and Sung 2014; Xu et al. 2018; Zhou et al. 2017).

Several quantization techniques are developed to optimize QDNNs, and these techniques mostly try to reduce quantization errors by considering the distribution of weights. In particular, various elaborate techniques are developed for CNNs, which include weight cluster (Park, Ahn, and Yoo 2017), stochastic rounding (Gupta et al. 2015), data distribution (Zhou et al. 2017), fittable quantization scale (Cai et al. 2017), or trainable quantization (Zhang et al. 2018; Yang et al. 2019).

Ott et al. (2016) showed that weight binarization decreases the performance of RNNs and they introduced stochastic and deterministic ternarization, and pow2-ternarization methods. Parameter-dependent adaptive threshold (He et al. 2016b) or increasing the size of a neural network (Kapur, Mishra, and Marr 2017) is also investigated. Other studies formulated an optimization problem to determine the optimal quantization step size with greedy approximation (Guo et al. 2017) or alternating multi-bit quantization (Xu et al. 2018). HitNet applies a different quantization algorithm to weight and activation (Wang et al. 2018). Ardakani et al. (2019) quantize only the weights using batch normalization between inputs and hidden state vectors.

## 2.2 Flat Minima in Loss Surfaces

Most high performance deep neural networks contain a vast number of parameters, and thus the training error almost converges to zero in many cases. The stochastic gradient descent (SGD) algorithm updates the weights to minimize the training error. However, neural network training is non-convex optimization, and low-training error does not necessarily ensure good test performance capability. An early study proposed that the determination of flat minima in the loss surface is important to train high-performance networks (Hochreiter and Schmidhuber 1997a). Recent studies suggested that the increased amount of noise in gradients of a small-batch method aids in reaching a flat minimum in the loss surface (Jastrzkebski et al. 2017). Conversely, large-batch training wherein the gradient noise is low requires an increased learning rate to obtain a good performance (Keskar et al. 2017).

The learning rate is the most important hyper-parameter in the SGD-based training. Typically, the learning rate is designed to monotonically decrease when the training proceeds. At the early stage of training, the weights should be updated coarsely, although they require fine-tuning at the final stage. However, Smith (2017) and Loshchilov and Hutter (2017) indicated that cyclically increasing and decreasing or warm-restarting the learning rate improves test accuracy. It should be noted that the training error is also fluctuating albeit not necessarily decreasing monotonically when the learning rate is alternating. Understanding flat minima is very important in QDNN design because quantization is equivalent to injecting noise to weights, and flat minima imply resiliency in weight distortion.

# 3 Training QDNN for Improved Generalization Capability

In this section, we first briefly explain the conventional neural network quantization algorithm and derive that learning rate to quantization precision ratio controls the stochastic noise. We also present a new QDNN training technique that aids to encourage reaching flat minima in the quantization domain.

## 3.1 Analysis of Training with Quantized Weights

The number of bits representing the quantized values is denoted as $b$. $b$ is usually from 1 to 8 and $b$-bit quantization can support up to $2^b$ levels. The quantization step size, $\Delta$, is inversely proportional to the number of levels, $2^b$. Thus, a low-precision weight needs a large $\Delta$. When $b$ is 2, a weight can be represented as 2-bit ternary, which is +$\Delta$, 0, and -$\Delta$. The $b$-bit symmetric uniform quantization including the 2-bit quantization can be generalized as follows:

$$\mathrm{Q}^b(\mathbf{w}) = \mathrm{sign}(\mathbf{w}) \cdot \Delta \cdot \min\Big\{ \Big\lfloor \Big( \frac{|\mathbf{w}|}{\Delta} + 0.5 \Big) \Big\rfloor, \frac{(M-1)}{2} \Big\} \tag{1}$$

where $M$ is $2^b - 1$. We employ an L2-error minimization between floating and fixed-point weights to obtain the quantization step size $\Delta$ (Hwang and Sung 2014; Rastegari et al. 2016).

Quantization can be interpreted as injecting noise whose range is between $-\frac{\Delta}{2}$ and $+\frac{\Delta}{2}$. Thus, the retraining process is equivalent to injecting noise to weights, which has been known to improve the generalization capability (Wen et al. 2018). As the number of bits, $b$, decreases, the amount of noise injection increases. Following the approach proposed in Jastrzkebski et al. (2017), we analyze the relationship between flatness and precision of weights. Specifically, the weight update procedure in quantization retraining is expressed as follows:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla L(\mathrm{Q}(\mathbf{w}_t)), \tag{2}$$

where $\mathrm{Q}(\cdot)$ is the quantization function, $L$ is the loss, and $\eta$ is the learning rate. Loss surface surrounding the local minimum $\mathbf{w}^*$ is approximated via the Hessian of $L$ at $\mathbf{w}^*$, and this is denoted as $\mathbf{H}$:

$$L(\mathbf{w}) \approx L(\mathbf{w}^*) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^T \times \mathbf{H} \times (\mathbf{w} - \mathbf{w}^*) \tag{3}$$

$$\nabla L(\mathbf{w}) \approx \nabla L(\mathbf{w}^*) + \mathbf{H} \times (\mathbf{w} - \mathbf{w}^*) \tag{4}$$

We rewrite Equation (2) by using Equation (4) as follows:

$$\mathbf{w}_{t+1} \approx \mathbf{w}_t - \eta \mathbf{H} \times (Q(\mathbf{w}_t) - \mathbf{w}^*) \tag{5}$$

$$\approx \mathbf{w}_t - \eta \mathbf{H} \times (\mathbf{w}_t + \mathcal{N}(0, (c^2/2^{2b})\boldsymbol{\mathcal{I}}) - \mathbf{w}^*) \tag{6}$$

$$= \mathbf{w}_t - \eta \mathbf{H} \times (\mathbf{w}_t - \mathbf{w}^*) - \eta \mathbf{H} \mathcal{N}(0, (c^2/2^{2b})\boldsymbol{\mathcal{I}}) \tag{7}$$

$$\approx \mathbf{w}_t - \eta \nabla L(\mathbf{w}_t) - \mathcal{N}(0, (\eta^2 c^2/2^{2b})\mathbf{H}^2), \tag{8}$$

where $c$ is a constant related to models. Therefore, we consider the quantization retraining algorithm as the gradient descent with noisy gradients (Kleinberg, Li, and Yuan 2018), and this corresponds to the Gaussian distribution with a covariance of $(\eta^2 c^2/2^{2b})\mathbf{H}^2$.

In Appendix A, we derive that the precision of weights $b$ determines the trade-off between the expected loss and the squared sum of eigenvalues with

$$\mathbb{E}(L(\mathbf{w}) - L(\mathbf{w}^*))/Tr(\mathbf{H}^2) \propto \eta/2^{2b}. \quad (9)$$

The eigenvalues of Hessian matrix represent the flatness of the loss surface around the local minimum. Therefore, we conclude that the quantization precision also influences the minima in low-precision domain as well as the three factors (learning rate, batch size, and gradient covariance) found in Jastrzkebski et al. (2017).

Based on the above analysis, we propose a new quantization training scheme, high-low-high-low-precision (HLHLp) training, that manipulates the learning rate, $\eta$, and quantization precision, $b$, during training to reach flat minima of the QDNN.

### 3.2 High-low-high-low-precision Training

The proposed HLHLp optimization employs a multi-step training scheme, and this consists of floating-point training of a model from scratch, coarse-tuning on low-precision, fine-tuning on high-precision, and fine-tuning on low-precision. It should be noted that the low-precision means 2-bit weight representation, and the high-precision indicates 8-bit or floating-point weight representation. The coarse-tuning step employs a high learning rate to escape from the current minimum point while the fine-tuning step proceeds with a low learning rate or decreasing learning rate. A detailed explanation of each step is given as follows and the entire algorithm is illustrated in Appendix B.

**High-precision Model Training (H-step)** The first step involves training a neural network in floating-point. Commonly known regularization techniques, such as dropout (Srivastava et al. 2014), and batch normalization (Ioffe and Szegedy 2015) can be employed. The learning rate is selected to obtain the optimal floating-point performance. Pretrained models can also be used. The initial learning rate in this step is denoted as $\eta^{\text{step 1}}$.

**Coarse-tuning on Low-precision (L-step)** The second step performs retraining to 2-bit QDNN using the pretrained model from the first step. Activation quantization can also be employed. The learning rate is $\alpha\eta^{\text{step 1}}$, where $\alpha$ is typically from 0.1 to 0.01 (Shin, Boo, and Sung 2017). We employ relatively high learning rate in this step for the purpose of coarse-tuning, as opposed to fine-tuning. The coarse-tuning aids to escape from sharp minima by increasing the dynamics of $\eta$ to $2^{2b}$ ratio in Equation (9). The new learning rate for this step is selected as approximately $\alpha\eta^{\text{step 1}} \times \frac{\Delta_{\text{2-bit}}}{\Delta_{\text{8-bit}}}$. The ratio is initially designed by considering the quantization step size ratios of 8-bit and 2-bit precision.

As we derived in Section 3.1, the eigenvalues of Hessian matrix represent the flatness of the loss surface. However, computation of the exact Hessian is super inefficient on large neural networks. To handle this problem, we approximate the Hessian by a diagonal matrix from the second moment of gradient $\mathbf{v}$ (Kingma and Ba 2014). Since $\mathbf{v}$ is an estimator of diag($\mathbf{H}^2$), we can obtain the sum of eigenvalues, $s$, exploiting by $Tr(\sqrt{\mathbf{v}})$. It should be noted that, to select the initial parameter of the third step, we measured both $s$ and the validation error rate during the training. More specifically, during the training of the current step, we save three to five model parameters considering validation results [1] and select the one which has the lowest value of $s$ among them.

**Fine-tuning on High-precision (H-step)** The third step performs retraining to 8-bit QDNN using the pretrained model from the second step. The initial learning rate for this step is lower than that in the second step. The fine-tuning decreases the dynamics of $\eta$ to $2^{2b}$ ratio in Equation (9). This step involves descending to the maximum possible extent from the new local minimum. To select the initial model for the next step, we evaluate the validation results.

**Fine-tuning on Low-precision (L-step)** The fourth step involves fine-tuning from the 8-bit weights obtained at the third step. The learning rate for this step is not extremely high and is decreasing. Thus, the final step is intended for fine-tuning and is similar to that in the conventional retraining-based method. We can repeat the second and third steps again. In this case, the total training is represented as HLHLHLp, and this denotes high-precision training, low-precision coarse-tuning, high-precision fine-tuning, low-precision coarse-tuning, high-precision fine-tuning, and final-tuning on low-precision. Additional HL steps may improve performance but increase training time. In our experiments, performance has converged in HLHLp in most cases.

The proposed HLHLp training scheme can employ various quantizers such as the uniform quantizer (Hwang and Sung 2014) and asymmetric quantizer (Zhu et al. 2017). The experimental results that combine the proposed training algorithm with various quantizers are shown in Section 4.

## 4 Experimental Results

We evaluate the proposed HLHLp training scheme on the following three tasks: image classification (CIFAR-10/CIFAR-100 (Krizhevsky and Hinton 2009), ImageNet (Russakovsky et al. 2015)), language modeling (PTB (Marcus et al. 1994) and WikiText-2 (Merity et al. 2016)), and speech recognition (WSJ corpus (Paul and Baker 1992)). The descriptions of these datasets are provided in Appendix C.

### 4.1 Image Classification with CNNs

**Network and Hyper-parameter Configuration:** We evaluate our method on CNNs for image classification. For the CIFAR-10 dataset, we train three different-sized ResNets (He et al. 2016a), namely ResNet-14, -20, and -32. Additionally, the same ResNet-20 and -32, and MobileNetV2 (Sandler et al. 2018) are employed for the

[1] Accuracy for classification problem or perplexity for language modeling.

Table 1: Test accuracy on CIFAR-10 and CIFAR-100 dataset. The numbers in the parenthesis are the accuracy difference between the floating and the 2-bit models. Both fine-tuning and HLHLp results are an average of five times running.

| Dataset | CIFAR-10 | | | CIFAR-100 | | |
|---|---|---|---|---|---|---|
| Model | ResNet-14 | ResNet-20 | ResNet-32 | ResNet-20 | ResNet-32 | MobileNetV2 |
| # Params | 0.18M | 0.27M | 0.47M | 0.28M | 0.48M | 2.45M |
| Float | 91.35 | 92.15 | 93.65 | 68.01 | 69.97 | 75.98 |
| fine-tuning | 89.20 (-2.15) | 90.86 (-1.29) | 92.32 (-1.33) | 64.47 (-3.54) | 66.90 (-3.07) | 74.97 (-1.01) |
| HLHLp | **90.64 (-0.71)** | **91.58 (-0.57)** | **93.05 (-0.60)** | **66.44 (-1.57)** | **68.66 (-1.31)** | **75.51 (-0.47)** |

CIFAR-100 dataset. All models for both the CIFAR-10 and CIFAR-100 datasets are trained with the same hyper-parameters as follows. The batch size is 128, and the number of epochs trained is 175. An SGD optimizer with a momentum of 0.9 is used. The learning rate starts at 0.1 and decreases by 0.1 times at the 75th and 125th epochs. Additionally, L2-loss is added with the scale of 5e-4. We employ simple symmetric uniform quantizer from Anwar, Hwang, and Sung (2015). The initial learning rate and the weight precision change as mentioned in Section 3.2 during HLHLp training. These changes in learning rate and precision are applied to all experiments in the rest of this paper.

Furthermore, we conduct the weight quantization of ResNet-18 on the ImageNet dataset using the proposed method. We employ a pretrained network as for the full precision model[2]. We set the batch size to 256 and conduct the retrain method for up to 20 epochs for each HLHLp step.

**Results on CIFAR-10/CIFAR-100:** The experimental results of the CIFAR-10 and CIFAR-100 datasets are presented in Table 1. Both the fine-tuned and the HLHLp-trained QDNNs are inherited from the same full-precision models. All layers in the models including the first and the last ones are quantized. In the case of the CIFAR-10 results, the performances of the 2-bit QDNNs improve when the HLHLp training is applied. Specifically, the HLHLp training results on ResNet-14 and ResNet-32 demonstrate 1.44% and 0.73% increase in test accuracy when compared to the existing fine-tuning method (Anwar, Hwang, and Sung 2015). The relative performance degradation of the 2-bit ResNet-32 for the full-precision model is 46% lower (0.6/1.3) when using the HLHLp training method. This small gap is due to the sufficiently large model size for the CIFAR-10 dataset. Large DNN models show a small difference between full-precision and low-precision networks.

The experiments with the more complex dataset (e.g. CIFAR-100) demonstrate more improvements. The test accuracy of the 2-bit ResNet-20 is 66.44% and 64.47% with the HLHLp training and the fine-tuning methods, respectively. The accuracy difference between our HLHLp and conventional training methods is reduced when the model size increases. However, the HLHLp training method demonstrates a comparable accuracy with the floating-point model in the MobileNetV2, which has a large number of parameters, but the conventional training method reports a performance degradation of 1.01%.

[2]https://github.com/facebook/fb.resnet.torch

Table 2: HLHLp training results on ResNet-18 ImageNet. In this experiment, only the weights are quantized in 2-bit. The values in the parentheses are the difference between the full-precision and quantized accuracy (%) in literature. HLHLp result is an average of five times running.

| W2/A32 | Levels | Top-1 Acc | Top-5 Acc |
|---|---|---|---|
| TWN | 3 | 61.8 (N/A) | 84.2 (N/A) |
| TTQ | Asym3 | 66.6 (-3) | 87.2 (-2) |
| LQ-Nets | 4 | 68.0 (-2.3) | 88.0 (-1.5) |
| ADMM | 3 | 67.0 (-2.1) | 87.5 (-1.5) |
| HLHLp (ours) | 3 | 67.2 (**-1.6**) | 87.8 (**-0.8**) |

**Results on ImageNet:** The experimental results of the ImageNet dataset are reported in Table 2. The compared low-precision models include TWN (Fengfu, Bo, and Bin 2016), TTQ (Zhu et al. 2017), LQ-Nets (Zhang et al. 2018), and ADMM (Leng et al. 2018). These models employ 2-bit weights but we do not quantize the activations. Ours, TWN, and ADMM employ the symmetric ternary quantization. However, TTQ employs asymmetric ternary (AT) quantization, whereas LQ-Nets employs 4-level quantization. AT and 4-level quantization help in improving the performance but also make the inference more complex. The experimental results demonstrate that the proposed method is effective and that the top-1 accuracy with ternary weights is better than ADMM. In the comparison of the accuracy difference between the full-precision model and the QDNN, our HLHLp training results demonstrate 1.6% degradation on Top-1 accuracy, which is much better than LQ-Nets (2.3%), TTQ (3%), and ADMM (2.1%). In LQ-Nets, the first and last layers of the model are not quantized, whereas, in ours, all the layers are quantized.

As part of the generalization test, we also evaluate our QDNN model with a contaminated dataset (Hendrycks and Dietterich 2018), which mixes various types of noise in the ImageNet validation set. Our proposed HLHLp training scheme increases the average noise accuracy from 39.08% (HL) to 45.99% (HLHL) in 2-bit QDNNs. Thus, our HLHLp training helps to increase the generalization capability in QDNN. The detailed results for each contaminated dataset are reported in Appendix D.

Table 3: PPL for 2-bit ternary and 2-bit 4-level weight quantized network of LSTM and GRU based language models on PTB test set. The activations are also quantized in 2-bit 4-level. The number in the parenthesis represents that the gap of the PPL between the 2-bit and full-precision in literature. HLHLp result is an average of five times running.

| W2 (Ternary)/A2 | LSTM | GRU | W2 (4-level)/A2 | LSTM | GRU |
|---|---|---|---|---|---|
| He et al. (2016b) | 152 (43) | 150 (50) | Zhou et al. (2017) | 126 (20) | 142 (42) |
| Kapur, Mishra, and Marr (2017) | 152.2 (43.5) | N/A | Guo et al. (2017) | 100.3 (10.5) | 105.1 (12.6) |
| Wang et al. (2018) | 110.3 (13.1) | 113.5 (10.8) | Xu et al. (2018) | 95.8 (6.0) | 101.2 (8.7) |
| HLHLp (Ours) | **97.27 (7.82)** | **95.04 (1.80)** | HLHLp (Ours) | **94.89 (5.44)** | **96.20 (2.96)** |

Table 4: Quantization results on WikiText-2 test set for 2-bit quantized networks. 'FP' means full-precision and 'Difference' represents the gap between the PPL for 2-bit and full-precision in literature. HLHLp result is an average of five times running.

| | Weight Levels | Test PPL for LSTM | | | Test PPL for GRU | | |
|---|---|---|---|---|---|---|---|
| | | FP | 2-bit | Difference | FP | 2-bit | Difference |
| Wang et al. (2018) | Ternary | 114.37 | 126.72 | 12.35 | 124.50 | 132.49 | 7.99 |
| HLHLp (ours) | Ternary | 103.0 | **107.66** | **4.66** | 108.68 | **105.96** | **-2.72** |
| Xu et al. (2018) | 4-level | **100.10** | 106.10 | 6.00 | **106.70** | 113.70 | 7.00 |
| HLHLp (ours) | 4-level | 103.0 | **105.5** | **2.5** | - | - | - |

## 4.2 Language Modeling on PTB and WikiText-2

**Network and Hyper-parameter Configuration:** For the quantitative comparison with previous works (Guo et al. 2017; He et al. 2016b; Kapur, Mishra, and Marr 2017; Wang et al. 2018; Xu et al. 2018; Zhou et al. 2017), we constructed two word-level language models (LMs) containing one long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997b) or one gated recurrent unit (GRU) (Cho et al. 2014). Each LM has a 300-memory cell for PTB and 512-memory cell for WikiText-2. The initial learning rate for the floating-point network is 1.0. After 10 epochs, the learning rate decreases by a factor of 0.9 at each epoch. We clip the norm of the gradients by 1.0 for PTB and 3.5 for WikiText-2. Both the batch-size and unrolling steps are 20 for PTB, while, for WikiText-2, the values are 50 and 30, respectively. We apply dropout (Srivastava et al. 2014) only at non-recurrent connections as suggested in Zaremba, Sutskever, and Vinyals (2015) with a keeping probability of 0.5 for PTB and 0.6 for WikiText-2. The performance of LM is measured via perplexity (PPL). An LM with low PPL is considered a good model.

**Results on PTB:** The comparison of the PPL of our HLHLp scheme and that of previous studies is presented in Table 3 for 2-bit ternary and 2-bit 4-level weight representations. We employ two previously developed quantizers for the 2-bit ternary (Wang et al. 2018) and 2-bit 4-level (Guo et al. 2017) weight respectively. The activations are also quantized in 2-bit. The HLHLp with ternary weights significantly outperforms the previous studies and also exhibits better results for the 2-bit 4-level representation. The GRU results also outperform both the 2-bit ternary and 2-bit 4-level representations. To the best of our knowledge, these results are the state-of-the-art when quantizing both weight and activation in 2-bit. Appendix E presents the results when employing a simple uniform quantizer (Shin, Hwang, and Sung 2016) and the change in PPL during the progress of each step in the HLHLp training scheme. HLHLp yielded significantly improved the results for all three quantizers when compared with those of the conventional training scheme.

**Results on WikiText-2:** The PPL of WikiText-2 is reported in Table 4. We employ a simple uniform quantizer from Shin, Hwang, and Sung (2016). The quantized network trained using HLHLp outperforms the other previous results. The LSTM model quantized with the proposed method shows lower (better) PPL when compared to the previous works in both ternary and 4-level weight. Especially for the 2-bit 4-level result, we achieve the test PPL of 105.5 that is 0.6 lower than the work of Xu et al. (2018) although our full-precision PPL is 2.9 higher (worse) than the compared work. In the case of GRU, our HLHLp quantization scheme demonstrates a much better test PPL than Xu et al. (2018). We have improved the state-of-the-art PPL from 113.7 to 105.96. Surprisingly, the quantized network performs better than the floating-point model, which suggest that the HLHLp scheme works as a regularizer.

## 4.3 Speech Recognition on WSJ Corpus

**Network and Hyper-parameter Configuration:** We construct three unidirectional LSTM layers with 512 memory cells. We employ the connectionist temporal classification (CTC) loss to train an RNN-based acoustic model (AM). We clip the norm of the gradients by 4, and train the AM with the Adam optimizer. A dropout with a keeping probability of 0.5 is applied for all the non-recurrent connections. The initial learning rate for the floating-point training is 3e-4, which decreases by a factor of 0.2 whenever the validation loss does not decrease thrice consecutively.

**Results:** The experiment results for WSJ are reported in Table 5. For comparison, we report another quantization result of the same RNN trained with the method suggested

Table 5: HLHLp training results on WSJ corpus. We quantize both weight and activations in 2-bit. 'CER' is character error rate (%) and 'WER' means word error rate (%). 'Clean' represents the results on Aurora-4 clean set, and 'Noisy' means the results on average of all noisy set.

| W2/A2 | WSJ | | Aurora-4 | |
|---|---|---|---|---|
| | Test CER | Test WER | Clean CER | Noisy CER |
| Float | 8.18 | 11.16 | 7.37 | 51.95 |
| Lee et al. (2016) | 9.76 | 11.32 | 8.58 | 51.65 |
| HLHLp (ours) | **8.21** | **11.27** | **6.78** | **48.6** |

Table 6: Ablation study on GRU PTB LM. The results are reported in PPL. Results in the same column represent obtained PPL with the exactly same epochs.

| | | | | |
|---|---|---|---|---|
| (A) | Float (H) | 2-bit (L) | 8-bit (H) | 2-bit (L) |
| | 95.32 | 99.12 | 92.25 | 96.97 |
| (B) | Float (H) | Float (H) | Float (H) | 2-bit (L) |
| | 95.32 | 100.98 | 98.13 | 115.92 |
| (C) | Float (H) | 2-bit (L) | 2-bit (L) | 2-bit (L) |
| | 95.32 | 99.46 | 97.48 | 97.84 |
| (D) | Float (H) | Float (H) | Float (H) | 2-bit (L) |
| | - | - | 95.06 | 112.08 |

in Lee et al. (2016). The character error rate (CER) is measured using greedy decoding. To obtain the word error rate (WER), we follow the method used in Miao, Gowayyed, and Metze (2015) to decode the output of the CTC-AM using the weighted finite-state transducers (WFST) network. We used a retrained trigram LM with an extended vocabulary for decoding. The CER of the full-precision model is 8.18%, and that of the 2-bit quantized model measured after the HLHLp training is 8.21%, which demonstrates almost no degradation. As part of the generalization test, we evaluate on the Aurora-4 noisy test corpus (Parihar et al. 2004), which mixes the noises of a car, babble, restaurant, street, airport, and train to the WSJ eval clean test set. The evaluation results of this test are presented in Table 5. When full-precision was quantized to 2 bits using Lee et al. (2016) method, CER increases by 1.21% on the clean set, however, our HLHLp training shows 0.59% higher accuracy than the full-precision result. A similar tendency is observed in the noise test. The performance of the 2-bit weight representation obtained by the HLHLp training is 3.35% better than the full-precision performance based on the average value of the noise test. In Appendix F, we present the detailed results for all the noise entries and each step in HLHLp method.

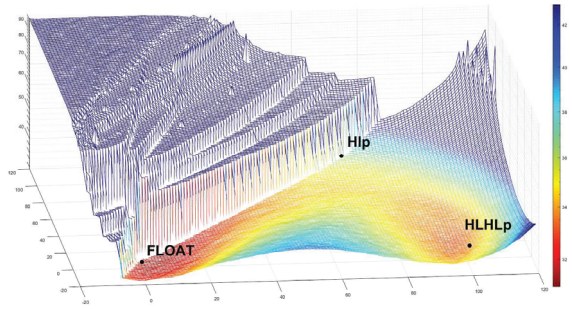
### 4.4 Discussion

**Ablation Study:** The experimental results demonstrate that the proposed HLHLp training method works exceptionally well for all experiments, including image classification, language modeling, and speech recognition. However, some questions still exist, such as "Are the improved results due to the longer training time?" and "Which part helps in increasing the performance?". To answer these questions, we conduct ablation experiments that optimizes an LM with a GRU using four different approaches as follows:

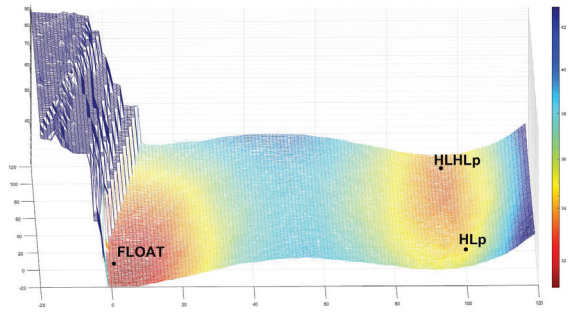
- (A) is the proposed HLHLp training employing floating-point training for 50 epochs, 2-bit retraining for 30 epochs with high a learning rate, 8-bit retraining for 30 epochs with a low learning rate, and 2-bit retraining for 30 epochs with a low learning rate.
- (B) employs 110 (=50+30+30) epochs of floating-point training with a cyclic learning rate which is exactly the same as the learning rate of (A). Additionally, 2-bit retraining for 30 epochs is conducted with the same learning rate as that of the last step in (A).
- (C) adopts 50 epochs of floating-point training and 90 (=30+30+30) epochs of 2-bit retraining with exactly the same learning rate as that of (A). Therefore, this setting converts the high-precision in the third step of (A) into low-precision.
- (D) conducts floating-point training but monotonically decreases the learning rate during 110 epochs and performs the retraining of 30 epochs with 2-bit weights representations. Thus, this method uses only fine-tuning.

The detailed results are presented in Table 6. The results clearly indicate that HLHLp ((A)) training performs much better than training with cyclical learning rate ((B)) or monotonic decreasing learning rate ((D)) without converting precision in the middle of steps in HLHLp training. The gap in PPL between (A) and (C) also indicates that fine-tuning in high-precision aids in improving the performance.

**Visualization of Loss Surface:** We employ 3-dimensional graphical visualization to demonstrate that our proposed HLHLp training scheme aids to reach flat minima in loss surface. We employ the method developed by Garipov et al. (2018). This method can help compare the training or test loss of three neural network models on the same 3-D surface, while the previous visualization method in (Li et al. 2018) only shows the loss surface of one model. Figure 1 depicts the loss surface of test error on ResNet20 using CIFAR-100 dataset. Figure 1 (a) compares the test loss of three models: full-precision ('FLOAT'), 2-bit quantized network retrained using a very small learning rate or fine-tuning ('Hlp'), and 2-bit quantized network trained with HLHLp ('HLHLp'). We can find a path connecting 'FLOAT' and 'Hlp'. Note that 'Hlp' point is located very close to the steep loss wall, suggesting a poor generalization capability. On the other hand, connecting 'FLOAT' and 'HLHLp' seems more difficult because the loss surface between them is not flat. However, 'HLHLp' is located near the center of a wide basin or a flat minimum. Apparently, 'HLHLp' should be preferred for good generalization. In Figure 1 (b), we compare the full-precision ('FLOAT'), 2-bit QDNN trained with HLHLp ('HLHLp'), and 2-bit QDNN with HLp ('HLp'). Note that 'HLp' means 2-bit QDNN after the second step of HLHLp training scheme. 'HLp' employs a very large learning rate for coarse tuning. Here, we can find that 'HLp' is




(a)




(b)

Figure 1: 3-D loss surface for test error on ResNet20 CIFAR-100. The three points in (a) indicate full-precision (FLOAT), 2-bit QDNN that trained with fine-tuning (Hlp), and 2-bit QDNN that trained with HLHLp (HLHLp). The three points in (b) represent full-precision (FLOAT), 2-bit QDNN that trained with HLHLp (HLHLp), and 2-bit QDNN that trained with HLp (HLp). Note that HLp means 2-bit QDNN after the second step of HLHLp training scheme.

at the same basin with the 'HLHLp', but is at the boundary. The remaining steps of HLHLp training help move 'HLp' to the near center of the basin.

In Figure 1 (a) and (b), we show the test loss surface. The training loss surface can be found in Appendix H. Appendix H contains more results using $\epsilon$-sharpness (Keskar et al. 2017) and figures produced with another visualization method (Li et al. 2018).

## 5 Concluding Remarks

In this study, we developed a HLHLp training scheme to obtain high-performance quantized neural networks. At each training step, we employed different precisions and abruptly changing learning rates during training to escape from sharp minima and reach a flatter loss surface. Thus, the proposed approach significantly differs from conventional methods, wherein training with quantized networks is conducted for fine-tuning and the learning rates typically monotonically decrease. We applied the training scheme to the quantization of RNNs and CNNs and obtained very good performance closing the gap between full-precision and low-precision models. Specifically, the method exhibited extremely good results with respect to the quantization of RNNs.

## Acknowledgments

This work was supported by Samsung Advanced Institute of Technology through Neural Processing Research Center (NPRC) in Seoul National University. This work was also supported in part by the Brain Korea 21 Plus Project and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No.2018R1A2A1A05079504).

## References

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

Ardakani, A.; Ji, Z.; Smithson, S. C.; Meyer, B. H.; and Gross, W. J. 2019. Learning recurrent binary/ternary weights. *International Conference on Learning Representations (ICLR)*.

Cai, Z.; He, X.; Sun, J.; and Vasconcelos, N. 2017. Deep learning with low precision by half-wave gaussian quantization. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5406–5414. IEEE.

Cho, K.; Van Merriënboer, B.; Bahdanau, D.; and Bengio, Y. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Courbariaux, M.; Bengio, Y.; and David, J.-P. 2015. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems (NIPS)*, 3123–3131.

Fengfu, L.; Bo, Z.; and Bin, L. 2016. Ternary weight networks. In *NIPS Workshop on EMDNN*, volume 118, 119.

Garipov, T.; Izmailov, P.; Podoprikhin, D.; Vetrov, D. P.; and Wilson, A. G. 2018. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Advances in Neural Information Processing Systems*, 8789–8798.

Guo, Y.; Yao, A.; Zhao, H.; and Chen, Y. 2017. Network sketching: Exploiting binary structure in deep CNNs. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2. IEEE.

Gupta, S.; Agrawal, A.; Gopalakrishnan, K.; and Narayanan, P. 2015. Deep learning with limited numerical precision. In *International Conference on Machine Learning (ICML)*, 1737–1746.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016a. Deep residual learning for image recognition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. IEEE.

He, Q.; Wen, H.; Zhou, S.; Wu, Y.; Yao, C.; Zhou, X.; and Zou, Y. 2016b. Effective quantization methods for recurrent neural networks. *arXiv preprint arXiv:1611.10176*.

Hendrycks, D., and Dietterich, T. G. 2018. Benchmarking neural network robustness to common corruptions and surface variations. *arXiv preprint arXiv:1807.01697*.

Hochreiter, S., and Schmidhuber, J. 1997a. Flat minima. *Neural Computation* 9(1):1–42.

Hochreiter, S., and Schmidhuber, J. 1997b. Long short-term memory. *Neural computation* 9(8):1735–1780.

Hubara, I.; Courbariaux, M.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2017. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research* 18(187):1–30.

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

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Jastrzkebski, S.; Kenton, Z.; Arpit, D.; Ballas, N.; Fischer, A.; Bengio, Y.; and Storkey, A. 2017. Three factors influencing minima in SGD. *arXiv preprint arXiv:1711.04623*.

Kapur, S.; Mishra, A.; and Marr, D. 2017. Low precision RNNs: Quantizing RNNs without losing accuracy. *arXiv preprint arXiv:1710.07706*.

Keskar, N. S.; Mudigere, D.; Nocedal, J.; Smelyanskiy, M.; and Tang, P. T. P. 2017. On large-batch training for deep learning: Generalization gap and sharp minima. *International Conference on Learning Representations (ICLR)*.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kleinberg, R.; Li, Y.; and Yuan, Y. 2018. An alternative view: When does sgd escape local minima? *arXiv preprint arXiv:1802.06175*.

Krizhevsky, A., and Hinton, G. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

Lee, M.; Hwang, K.; Park, J.; Choi, S.; Shin, S.; and Sung, W. 2016. Fpga-based low-power speech recognition with recurrent neural networks. In *2016 IEEE International Workshop on Signal Processing Systems (SiPS)*, 230–235. IEEE.

Leng, C.; Dou, Z.; Li, H.; Zhu, S.; and Jin, R. 2018. Extremely low bit neural network: Squeeze the last bit out with admm. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Li, H.; Xu, Z.; Taylor, G.; Studer, C.; and Goldstein, T. 2018. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems (NIPS)*, 6391–6401.

Loshchilov, I., and Hutter, F. 2017. Sgdr: Stochastic gradient descent with warm restarts. *International Conference on Learning Representations (ICLR)*.

Marcus, M.; Kim, G.; Marcinkiewicz, M. A.; MacIntyre, R.; Bies, A.; Ferguson, M.; Katz, K.; and Schasberger, B. 1994. The penn treebank: annotating predicate argument structure. In *Proceedings of the workshop on Human Language Technology*, 114–119. Association for Computational Linguistics.

Merity, S.; Xiong, C.; Bradbury, J.; and Socher, R. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Miao, Y.; Gowayyed, M.; and Metze, F. 2015. EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding. In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*, 167–174. IEEE.

Neyshabur, B.; Li, Z.; Bhojanapalli, S.; LeCun, Y.; and Srebro, N. 2018. Towards understanding the role of over-parametrization in generalization of neural networks. *arXiv preprint arXiv:1805.12076*.

Ott, J.; Lin, Z.; Zhang, Y.; Liu, S.-C.; and Bengio, Y. 2016. Recurrent neural networks with limited numerical precision. *arXiv preprint arXiv:1608.06902*.

Parihar, N.; Picone, J.; Pearce, D.; and Hirsch, H.-G. 2004. Performance analysis of the aurora large vocabulary baseline system. In *2004 12th European Signal Processing Conference*, 553–556. IEEE.

Park, E.; Ahn, J.; and Yoo, S. 2017. Weighted-entropy-based quantization for deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 7197–7205. IEEE.

Paul, D. B., and Baker, J. M. 1992. The design for the Wall Street Journal-based CSR corpus. In *Proceedings of the workshop on Speech and Natural Language*, 357–362. Association for Computational Linguistics.

Rastegari, M.; Ordonez, V.; Redmon, J.; and Farhadi, A. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 525–542. Springer.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision* 115(3):211–252.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4510–4520.

Shin, S.; Boo, Y.; and Sung, W. 2017. Fixed-point optimization of deep neural networks with adaptive step size retraining. In *2017 IEEE International conference on acoustics, speech and signal processing (ICASSP)*, 1203–1207. IEEE.

Shin, S.; Hwang, K.; and Sung, W. 2016. Fixed-point performance analysis of recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, 976–980. IEEE.

Smith, L. N. 2017. Cyclical learning rates for training neural networks. In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, 464–472. IEEE.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Wang, P.; Xie, X.; Deng, L.; Li, G.; Wang, D.; and Xie, Y. 2018. Hitnet: Hybrid ternary recurrent neural network. In *Advances in Neural Information Processing Systems (NIPS)*, 602–612.

Wen, W.; Wang, Y.; Yan, F.; Xu, C.; Wu, C.; Chen, Y.; and Li, H. 2018. Smoothout: Smoothing out sharp minima to improve generalization in deep learning. *arXiv preprint arXiv:1805.07898*.

Xu, C.; Yao, J.; Lin, Z.; Ou, W.; Cao, Y.; Wang, Z.; and Zha, H. 2018. Alternating multi-bit quantization for recurrent neural networks. *International Conference on Learning Representations (ICLR)*.

Yang, J.; Shen, X.; Xing, J.; Tian, X.; Li, H.; Deng, B.; Huang, J.; and Hua, X.-s. 2019. Quantization networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7308–7316.

Zaremba, W.; Sutskever, I.; and Vinyals, O. 2015. Recurrent neural network regularization. *International Conference on Learning Representations (ICLR)*.

Zhang, D.; Yang, J.; Ye, D.; and Hua, G. 2018. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 365–382.

Zhou, S.-C.; Wang, Y.-Z.; Wen, H.; He, Q.-Y.; and Zou, Y.-H. 2017. Balanced quantization: An effective and efficient approach to quantized neural networks. *Journal of Computer Science and Technology* 32(4):667–682.

Zhu, C.; Han, S.; Mao, H.; and Dally, W. J. 2017. Trained ternary quantization. *International Conference on Learning Representations (ICLR)*.