CAPACITY OF DEEP NEURAL NETWORKS UNDER PARAMETER QUANTIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Most deep neural networks (DNNs) require complex models to achieve high performance. Parameter quantization is widely used for reducing the implementation complexities. Previous studies on quantization were mostly based on extensive simulation using training data. We choose a different approach and attempt to measure the per-parameter capacity of DNN models and interpret the results to obtain insights on optimum quantization of parameters. This research uses artificially generated data and generic forms of fully connected DNNs, convolutional neural networks, and recurrent neural networks. We conduct memorization and classification tests to study the effects of the number and precision of the parameters on the performance. The model and the per-parameter capacities are assessed by measuring the mutual information between the input and the classified output. We also extend the memorization capacity measurement results to image classification and language modeling tasks. To get insight for parameter quantization when performing real tasks, the training and test performances are compared.

1 INTRODUCTION

Deep neural networks (DNNs) have achieved impressive performance on various machine learning tasks. Several DNN architectures are known, and the most famous ones are fully connected DNNs (FCDNNs), convolutional neural networks (CNNs), and recurrent neural networks (RNNs).

It is known that neural networks do not need full floating-point precision for inference (Dundar & Rose, 1995; Hwang & Sung, 2014; Lin et al., 2016). A 32-bit floating-point parameter can be reduced to 8-bit, 4-bit, 2-bit, or 1-bit, but this can incur performance degradation. Therefore, precision should be optimized, which is primarily conducted by extensive computer simulations using training data. This not only takes much time for optimization but also can incorrectly predict the performance in real environments when the characteristics of input data are different from the training data.

In this study, we attempt to measure the capacity of DNNs, including FCDNN, CNN, and RNN, using a memorization and classification task that applies random binary input data. The per-parameter capacities of various models are estimated by measuring the mutual information between the input data and the classification output. Then, the fixed-point performances of the models are measured to determine the relation between the quantization sensitivity and the per-parameter capacity. The memorization capacity analysis results are extended to real models for performing image classification and language modeling, by which the parameter quantization sensitivity is compared between memorization and generalization tasks.

The contributions of this paper are as follows.

- We experimentally measure the memorization capacity of DNNs and estimate the per-parameter capacity. The capacity per parameter is between 2.3 bits to 3.7 bits, according to the network structure, which is FCDNN, CNN, or RNN. The value is fairly independent of the model size.
- We show that the performance of the quantized networks is closely related to the capacity per parameter, and FCDNNs show the most resilient quantization performance while RNNs suffer most from parameter quantization. The network size hardly effects the quantization performance when DNN models are trained to use full capacity.

- We explain that severe quantization, such as binary or ternary weights, can be employed without much performance degradation when the networks are in the over-parameter region.
- We suggest the sufficient number of bits for representing weights of neural networks, which are approximately 6 bits, 8 bits, and 10 bits for FCDNNs, CNNs, and RNNs, respectively. This estimate of the number of bits for implementing neural networks is very important considering that many accelerators are designed without any specific training data or applications.
- The study with real-models shows that neural networks are more resilient to quantization when performing generalization tasks than conducting memorization. Thus, the optimum bits obtained with the memorization tasks are conservative and safe estimate when solving real problems.

The paper is organized as follows. In Section 2, previous works on neural network capacity and fixed-point optimization are briefly presented. Section 3 explains the capacity measurement methods for DNN models. Section 4 presents parameter capacity measurement results for FCDNNs, CNNs, and RNNs. The quantization performances measured on DNNs are presented in Section 5. Concluding remarks follow in Section 6.

2 Related works and backgrounds

2.1 Neural network capacity

The capacity of neural networks has been studied since the early days of DNN research. Although the capacity can be defined in many ways, it is related to the learnability of networks. The capacity of networks is shown as the number of uncorrelated random samples that can be memorized (Cover, 1965). A single-layer perceptron with n parameters can memorize at least $2n$ random samples (Gardner & Derrida, 1988). In other words, the network can always construct a hyperplane with n parameters that divides $2n$ samples. Additionally, the capacity of a three-layer perceptron is proportional to the number of parameters (Akaho & Amari, 1990). Recently, RNNs were trained with random data to measure the capacity per parameter (Collins et al., 2017). Our study is strongly motivated by this research, and extends it to the quantization performance interpretation of generic DNN models, including FCDNN, CNN, and RNN. Recent studies have showed that neural networks have a generalization ability even if the expressive capacity of the model is sufficiently large (Zhang et al., 2017; Arpit et al., 2017). In this paper, we also discuss the effect of network quantization when performing generalization tasks.

2.2 Fixed-point deep neural networks

Early works on neural network quantization usually employed 16-bit parameters obtained by directly quantizing the floating-point numbers (Dundar & Rose, 1995). Recently, a retraining technique was developed to improve the performance of quantized networks (Hwang & Sung, 2014; Lin et al., 2016). Retraining-based quantization was applied to CNN and RNN models, showing superior performance compared to directly quantized ones (Anwar et al., 2015; Shin et al., 2016). Many studies attempting extreme quantization have been published, such as 2-bit ternary (Hwang & Sung, 2014; Li et al., 2016; Zhu et al., 2017), 1-bit binary weight quantization, and XNOR networks (Courbariaux et al., 2015; Rastegari et al., 2016). Some aggressive model compression techniques also employed vector quantization or table look-up (Han et al., 2015a; Boo & Sung, 2017). However, not all CNNs show the same quantization performance. For example, AlexNet (Krizhevsky et al., 2012) shows almost the same performance with only 1-bit quantized parameters. However, the same quantization technique incurs a very severe performance loss when applied to ResNet (Rastegari et al., 2016). A previous study shows that large sized networks are more resilient to severe quantization than smaller ones (Sung et al., 2015). Theoretical works and many practical implementation optimization techniques have been studied (Han et al., 2015b; Iandola et al., 2016; Kim & Smaragdis, 2016; Sakr et al., 2017; Louizos et al., 2017). Recent work increases the number of network parameters to preserve the performance under low-precision quantization (Mishra et al., 2017). Our works are not targeted to a specific data or model, but introduce the general understanding of parameter quantization.

3 Network capacity measurements of DNNs

3.1 Capacity measurements on a memorization task

We assess the network capacity of DNN models using a random data memorization and classification task (Collins et al., 2017). In this task, N random binary vectors, X, are generated and each is randomly and uniformly assigned to the output label Y. The size of the binary vector depends on the DNN model. For FCDNN, the input X is a one dimensional vector whose size is determined by the hidden layer dimension. In CNN, the input needs to be a 2-D or 3-D tensor. Input samples of CNNs are generated by concatenating and reshaping random binary vectors. During the training process, the DNN is trained to correctly predict the label, which is 0 or 1, of the random input X. As the number of input data size, N, increases, the classification accuracy drops because of the limited memorization capacity. Note that the accuracy for the memorization task refers to the training performance after convergence because there is no proper test dataset for random training samples.

The capacity is measured using the mutual information, defined as a measure of the amount of information that one random variable contains about another random variable (Cover & Thomas, 2012). The mutual information of a trained network with N input samples is calculated as follows:

$$\begin{aligned} I(Y;\hat{Y_\theta}|X) &= H(Y|X) - H(Y|\hat{Y_\theta},X) \\ &= N\left(1-(p\log_2\frac{1}{p}+(1-p)\log_2\frac{1}{(1-p)})\right), \end{aligned} \tag{1}$$

where p is the mean classification accuracy for all samples under trained parameter θ. If the training accuracy is 1, the model memorizes all random samples and the $I(Y;\hat{Y_\theta}|X)$ becomes the number of samples N. If the training accuracy is 0.5, $I(Y;\hat{Y_\theta}|X)$ goes to 0. Please consult Supplementary materials for details of Eq. (1).

The network capacity is defined as

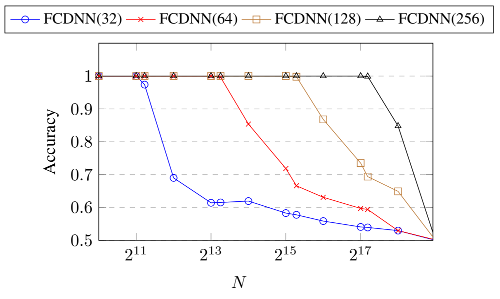
$$C = \max_\theta I(Y;\hat{Y_\theta}|X). \tag{2}$$

The accuracy, p, may vary depending on the training method of the model. We find N and p that maximize the mutual information of the networks by iteratively training the models. This optimization employs both grid search- and Bayesian optimization-based hyper-parameter tuning (Brochu et al., 2010). The optimization procedure consists of three stages. First, we try to find the largest input data size whose accuracy is slightly lower than 1. Second, we perform a grid search to determine the boundary values of the hyper-parameters. The searched hyper-parameters can include initialization, optimizer, initial learning rate, learning rate decay factor, batch size, and optimizer variables. Finally, we conduct hyper-parameter tuning within the search space using Scikit-learn library (Pedregosa et al., 2011). We add the number of training samples N as a hyper-parameter and use the mutual information of Eq. (1) as the metric for the optimization.

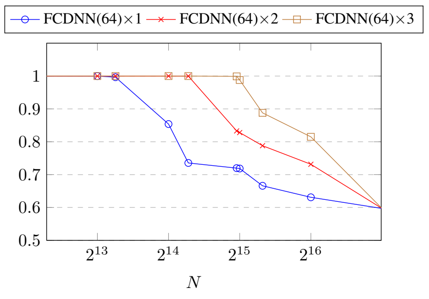
3.2 Network quantization and parameter capacity

Quantization of model parameters perturbs the trained network, therefore, fixed-point training or retraining with full-precision backpropagation is usually needed (Hwang & Sung, 2014; Li et al., 2016; Courbariaux et al., 2015; Zhou et al., 2017). However, the performance of the quantized networks does not always meet that of the floating-point models, even after retraining. This suggests that model capacity is reduced by quantization, especially when the number of bits used is very small.

In this research, we observe the memorization capacity degradation caused by quantization in generic FCDNN, CNN, and RNN models. The uniform quantization is used for the sake of convenient arithmetic, and the same step size is assigned to each layer in the FCDNN, each kernel in the CNN, or each weight matrix in the LSTM layer. The bias values are not quantized, because they have a large dynamic range. It is important to note that the weights connected to the output are not quantized, because their optimum bit-widths depend on the number of labels in the output. Quantization is performed from floating-point to 8-bit, 6-bit, 5-bit, 4-bit, 3-bit, and 2-bit precision, in sequence. Retraining is performed after every quantization, but requires only a small number of epochs, because only fine-tuning is needed (Hwang & Sung, 2014).



(a) Varying width



(b) Varying depth

Figure 1: Memorization performances according to the (a) width and (b) depth of the FCDNN.

3.3 Generalization capability assessment of quantized networks using weight perturbation

We compare the generalization performance of floating-point and fixed-point DNNs by visualizing the loss surface. Loss is measured by applying Gaussian random noise to the parameters of the trained network as shown in Eq. (3).

$$f(\alpha) = L(\theta + \alpha\theta^{noise}). \tag{3}$$

Here, $L(\theta)$ is the loss according to the network parameters. The distribution of weights may vary depending on the model size and learning method. We apply the normalized filter noise to the θ^{noise} for fair comparison on different models (Li et al., 2017).

We employ two real networks, one is for image classification with CIFAR-10 dataset and the other is language modeling with Penn Tree Bank (PTB) dataset. One large and one small model are trained for these networks. We quantize those networks with the precision of 8, 6, 4, and 2 bits and analyze the variation of the surface according to the precision. θ^{noise} is added to the quantized parameters. In order to reduce the error due to randomness of noise, all loss values are measured with 10 different trials and the average values are plotted.

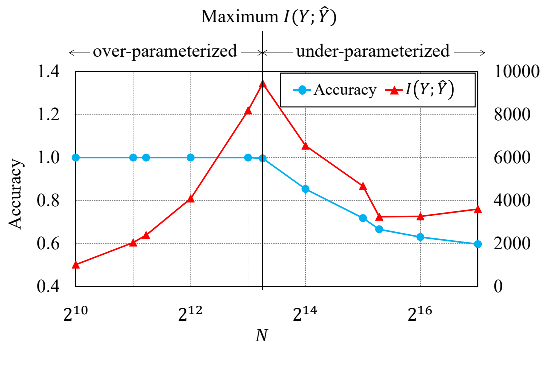
4 Experimental results on capacity of floating-point DNNs

The capacities of FCDNNs, CNNs, and RNNs are measured via the memorization task explained in Section 3.1. The models used for the test employ floating-point parameters.

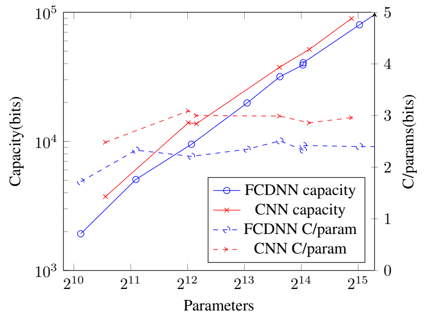
4.1 Capacity of FCDNNs

The training data for FCDNNs is a 1-D vector of size n_{in}. N input data are used as for the training data. The output, Y, is the randomly assigned label, either 0 or 1, for each input. Thus, inputs, X and Y, are represented as $X \in \{0,1\}^{N \times n_{in}}$ and $Y \in \{0,1\}^{N}$, respectively. The input data dimension, n_{in}, should be larger than $\log_2 N$ so that no overlapped data is contained among N input data. In the experiments for FCDNNs, the input vector dimension, n_{in}, is chosen to be equal to the number of units in the hidden layer.

We conduct experiments for FCDNNs with hidden layer dimensions of 32, 64, 128, and 256, and with hidden layer depths of 1, 2, 3, and 4. The initialization method chosen is the 'He' initialization (He et al., 2004) and gradients are updated following the rule in SGD, with momentum, which shows the best performance in our grid search. The initial learning rate for hyper parameter tuning is chosen between 0.001 and 0.05 on the log scale. The decay factor and momentum are set to have even distance values in the linear scale between 0.1 and 0.5 and between 0.6 and 0.99, respectively. For each model, experiments are conducted to measure the accuracy of memorization while increasing the size of the input data, N. Note that only the training error is measured in this memorization task, because there is no unseen data. Experimental results are based upon the best accuracy obtained when attempted with different hyper parameters. The capacity of the model is estimated according to Eq. (1), where p is the training accuracy. The experimentally obtained memorization capacities



(a)



(b)

Figure 2: (a) Mutual information according to the number of inputs N. (b) The relationship between the number of parameters and the capacity of networks in FCDNNs and CNNs.

of the FCDNN models are presented in Fig. 1, where depths of 1, 2, 3, and 4, and widths of 32, 64, 128, and 256 are used. When the number of hidden layers is the same, the amount of data that can be almost perfectly memorized quadruples when the dimension of the hidden layer is doubled. This means that the memorization capacity is linearly proportional to the number of parameters. Similarly, the FCDNN models with 2, 3, or 4 hidden layer depths can memorize 2, 3, or 4 times the input data as compared to the single layer DNN, respectively.

Fig. 2(a) shows the memorization accuracy and the mutual information obtained using Eq. (1) on the FCDNN. The model is composed of three layers and the hidden layer of size 64. Here, we find that the amount of mutual information steadily increases as the input data size grows. However, it begins to drop as the input size grows farther, and the memorization accuracy drops. By analyzing the accuracy trend of the model, it is possible to distinguish the input data size into three regions: the over-parameterized, the maximum performance, and the under-parameterized sections, as shown in Fig. 2(a). For example, if the model is trained to memorize only 10,000 data, it can be regarded as over-parameterized. The number of data that can be memorized by maximally utilizing all the parameters is between 30,000 and 40,000. In over-parameterized regions, performance can be maintained, even if the capacity of the networks is reduced.

The per-parameter capacity of FCDNNs is shown in Fig. 2(b). Regardless of the width or depth, one parameter has a capacity of 1.7 to 2.5 bits, and FCDNNs have an average of 2.3-bit capacity per parameter. This result is consistent with theoretical study (Gardner & Derrida, 1988; Akaho & Amari, 1990). The total capacity of the model may be interpreted as an optimal storage that can store a maximum of random binary samples (Gardner & Derrida, 1988; Bollé et al., 1991).

4.2 Capacity of CNNs

The capacity of CNNs is also measured via a similar memorization task. CNNs can have a variety of structures according to the number of channels, the size of the kernels, and the number of layers. The kernel size of CNNs in this test are either (3×3) or (5×5), which are the same for all layers, the number of convolution layers from 3 to 9. The dimensions of the inputs are $n_{height} = n_{width} = 32$ and $n_{channel} = 1$ for all experiments. Three max-pooling operations are applied to reduce the number of parameters in the fully connected layer. CNN structures used in our experiments are shown in Supplementary materials.

The CNN models contain not only convolution layers but also fully connected layers. Thus, the per-parameter capacity for convolution layers is calculated after subtracting the capacity for fully connected layers from the measured total capacity. We assume the per-parameter capacity of the fully connected layer as 2.3 bits to calculate the capacity for convolution layers. As shown in Fig. 2(b), the convolution layers have the per-parameter capacity of between 2.86 and 3.09 except the smallest model, which is higher than that of FCDNNs. The average capacity per parameter of the tested models is 3.0 bits.

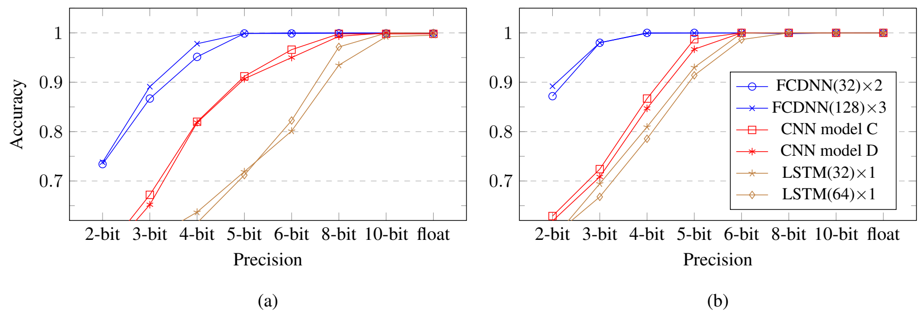


Figure 3: Quantization effect when networks use the (a) full capacity and (b) half capacity.

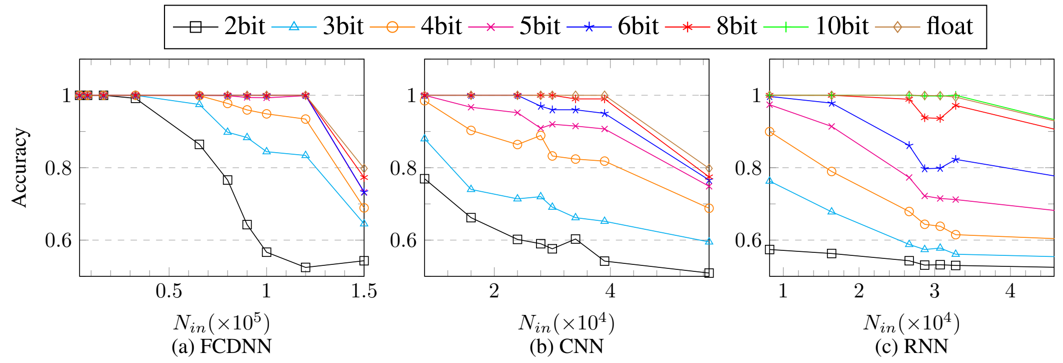


Figure 4: Quantization performance according to the number of data to train. (a) FCDNN, (b) CNN, and (c) RNN.

Results show that the per-parameter capacity of CNNs is higher than that of FCDNNs, even when CNNs memorize uncorrelated data. Note that one parameter of FCDNNs is used only once for each inference. However, the parameter of CNNs is used multiple times. This parameter-sharing nature of CNNs seems to increase the amount of information that one parameter can store.

4.3 Capacity of RNNs

It has been shown that the various structures of RNNs all have similar capacity per parameter of 3 to 6 bits (Collins et al., 2017). We train RNNs with a dataset with no sequence correlation to show the capacity of the parameters. The random input dataset is composed of inputs, $X \in \{0,1\}^{N \times n_{seq} \times n_{in}}$ and labels $Y \in \{0,1\}^N$, which are uniformly set to 0 or 1. The training loss is calculated using the cross-entropy of the label at the output of the last step.

We train RNNs with a single LSTM layer of 32-D. The input dimension, n_{in}, is also 32-D and the amount of unrolling sequence, n_{seq}, is five-step. It has been reported that unrolling of five-step almost saturates the performance in this setup (Collins et al., 2017). We apply 5 input random vectors, X_0, X_1, X_2, X_3, and X_4, each with 32-D, and assign one label to this 160-D vector at the last time step. The error propagates from the last step only, and the outputs at intermediate time-steps are ignored. The number of parameters in the network is 8,386. In this case, the maximum mutual information is obtained when the number of samples is 32K, and the memorization accuracy is 99.52 %. Therefore, the per-parameter capacity of the model is 3.7 bits. The RNN shows higher per-parameter capacity than FCDNNs and CNNs.

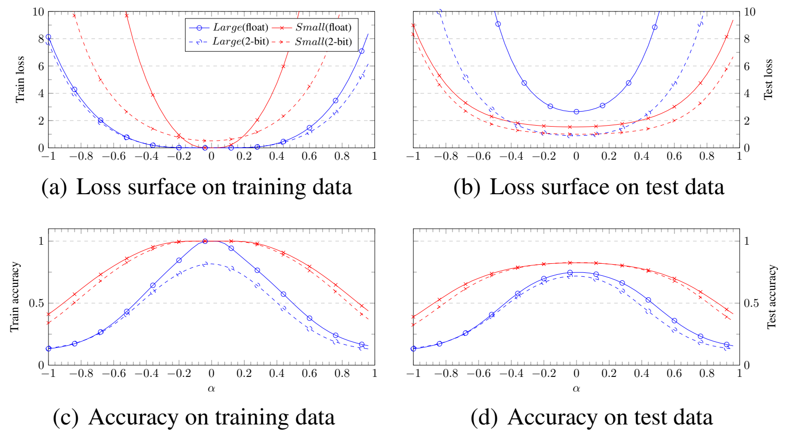


Figure 5: Loss surface and accuracy of floating-point and fixed-point CNNs. α denotes the scale of the parameter noise.

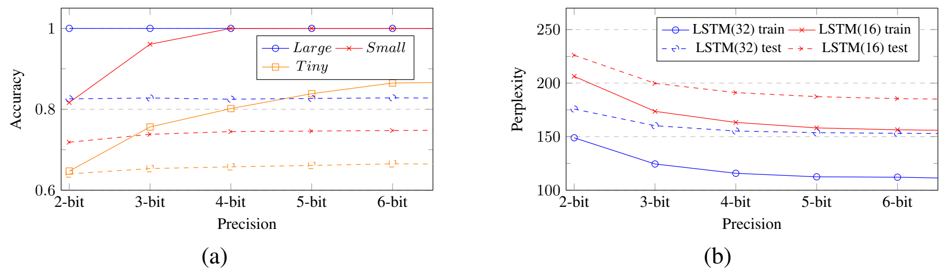


Figure 6: Performance degradation according to the quantization precision on (a) CNN trained with CIFAR-10 dataset and (b) RNN LM trained with PTB dataset.

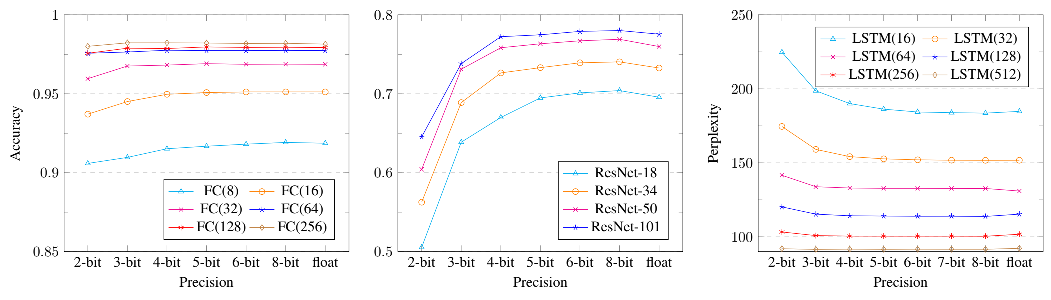
5 Experimental results of parameter quantization

5.1 Capacity under parameter quantization

We have shown that FCDNNs, CNNs, and RNNs have different per-parameter capacities. According to the parameter-data ratio, a trained DNN can be an over-parameterized, max-capacity, or under-parameterized model. Thus, we can assume that the DNN performance under quantization would depend on not only the network structure, such as FCDNN, CNN, or RNN, but also the parameter-data ratio. The experiments are divided into two cases. The first is to measure performance degradation via quantization precision when each model is in the maximum capacity region. The second analyzes performance when the models are in the over-parameterized region.

When the FCDNN, CNN, and RNN are trained to have the maximum memorization capacity, the performances with parameter quantization are shown in Fig. 3(a). The FCDNN, CNN, and RNN models are shown. The fixed-point performances of two FCDNNs, two CNNs, and two RNNs are illustrated. With 6-bit parameter quantization, the FCDNN shows no accuracy drop. However, those for CNNs and RNNs are 5 % and 18 %, respectively. Because the RNN contains the largest amount of information at each parameter, the loss caused by parameter quantization seems to be the most severe. We also find that there is no decline in performance until the parameter precision is lowered to 6-bit for FCDNNs, 8-bit for CNNs, and 10-bit for RNNs, even when all models use full capacity.

Next, we show the fixed-point performance of DNNs when they are trained to be in the over-parameterized region. Note that the per-parameter capacity is lowered in the over-parameterized region. We conducted simulation with half size of the maximum number of data that can be memorized. For example, an FCDNN used for the measurement has 3 hidden layers with a hidden-layer dimension of 128; the capacity of the corresponding model is about 2^{17} bits. The network is



(a) (b) (c)

Figure 7: (a) Accuracy of fixed-point FCDNN on MNIST dataset. (b) Top-1 test accuracy of fixed-point ResNets trained on ImageNet dataset. (c) Perplexity of fixed-point WLMs on PTB testset.

over-parameterized when the number of memorized samples is 2^{16}. Fig. 3(b) shows that the FCDNN model memorizes all samples even with 4-bit parameter quantization when the model uses half of the capacity. Also, over-parmeterized model is less sensitive to bit-precision on CNNs and RNNs. The performances of fixed-point DNNs with the number of samples are shown in Fig. 4. The result shows that DNNs are more robust when the networks are more over-parameterized.

5.2 Quantization experiments on real tasks

We have assessed the required precision of networks for performing memorization tasks. The memorization test only uses the training data that are artificially generated. However, most neural networks should conduct more than memorization because the test data are not seen during the training. In this section, we analyze the effects of network quantization for performing real tasks. We train two different sized CNN models with CIFAR-10 data. The structures of the two models are as follows:

$$\begin{aligned} &Small: 2 \times 16C - MP2 - 2 \times 32C - MP2 - 2 \times 64C - MP2 - 2 \times 128FC - 10out \\ &Large: 2 \times 64C - MP2 - 2 \times 128C - MP2 - 2 \times 256C - MP2 - 2 \times 512FC - 10out. \end{aligned} \tag{4}$$

The size of kernels of both models is (3×3), $16C$ represents a convolution layer with 16 channels and $128FC$ means a fully connected layer with 128-dimension. The number of parameters is 0.22M for the small model and 3.5M for the large one. Both models were trained with the same hyper-parameter setting.

To analyze the impact of network quantization on the test performance, we plot the loss and accuracy surfaces of floating-point and quantized CNN models in Fig. 5. For simplicity, the results of floating-point and 2-bit fixed-point CNN are given. Please refer to the Supplementary materials for other results. When applying the training data that may have been memorized during the training phase, the $large$ model shows indifferent performance surface regardless of the parameter precision. But, for the $small$ model, the 2-bit model shows quite degraded performance when compared to the floating-point network. However, the test accuracy of $small$ 2-bit model is not much lowered. We can notice that the loss surface of the 2-bit model shown in Fig. 5 is much wider than that of the floating-point model. This result is consistent with recent studies on generalization (Keskar et al., 2017; Li et al., 2017). This observation suggests that the quantized networks are more resilient when performing generalization tasks. Thus, the required precision of the network obtained with the memorization task can be considered a conservative estimate.

Fig. 6 shows the training and test data based performance of fixed-point CNN and RNN on real data. $Tiny$ model has the same structure as the $small$ model, but reducing the number of channels by half and the size of the fully connected layers by a quarter. The RNNs are trained for language modeling with Penn Treebank corpus (Marcus et al., 1993), and the models consists of two LSTM layers with the same dimension. Here, both for CNN and RNN models, we can confirm that large networks are more robust to quantization. Also, the networks need more parameter precision when conducting memorization tasks using the training set, rather than solving real problems using the test set.

We have measured fixed-point DNN performance on real tasks and results are shown in Fig. 7. FCDNN models are trained with MNIST dataset, ResNet (He et al., 2016) models are trained using ILSVRC-2012 dataset (Russakovsky et al., 2015) and RNN based word-level language models (WLMs) are designed using PTB dataset. Experimented FCDNNs and RNNs are composed of two FC layers and two LSTM layers with the same dimension, respectively. 4-bit quantized FCDNNs shows almost same performance compared to the floating-point networks even when the number of neurons are only 8. Performances are preserved up to 6 bits on ResNets and RNN WLMs.Their resiliency to quantization increases as networks become larger.

6 Concluding remarks

Quantization of parameters is a straightforward way of reducing the complexity of DNN implementations, especially when VLSI or special purpose neural processing engines are used. Our study employed simulations on varying sizes of generic forms of DNN models. Memorization tests using random binary input data were conducted to determine the total capacity by measuring the mutual information. Our simulation results show that the per-parameter capacity is not sensitive to the model size, but is dependent on the structure of the network models, such as FCDNN, CNN, and RNN. The maximum per-parameter capacities of FCDNNs, CNNs, and RNNs are approximately 2.3 bits, 3.0 bits, and 3.7 bits per parameter. Thus, RNNs have the tendency of demanding more bits when compared to FCDNNs. We quantized DNNs under various capacity-utilization regions and showed that the capacity of parameters are preserved up to 6 bits, 8 bits, and 10 bits on FCDNNs, CNN, and RNNs, respectively. The performance of the quantized networks was also tested with image classification and language modeling tasks. The results show that networks need more parameter precision when conducting memorization tasks, rather than inferencing with unseen data. Thus, the precision obtained through the memorization test can be considered a conservative estimate in implementing neural networks for solving real problems. This research not only gives valuable insights on the capacity of parameters but also provides practical strategies for training and optimizing DNN models.

References

S Akaho and S Amari. On the capacity of three-layer networks. In *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*, pp. 1–6. IEEE, 1990.

Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. 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.

Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *International Conference on Machine Learning*, pp. 233–242, 2017.

D Bollé, P Dupont, and J Van Mourik. The optimal storage capacity for a neural network with multi-state neurons. *EPL (Europhysics Letters)*, 15(8):893, 1991.

Yoonho Boo and Wonyong Sung. Structured sparse ternary weight coding of deep neural networks for efficient hardware implementations. In *Signal Processing Systems (SiPS), 2017 IEEE International Workshop on*. IEEE, 2017.

Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

Jasmine Collins, Jascha Sohl-Dickstein, and David Sussillo. Capacity and trainability in recurrent neural networks. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pp. 3123–3131, 2015.

Thomas M Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE transactions on electronic computers*, (3):326–334, 1965.

Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

Gunhan Dundar and Kenneth Rose. The effects of quantization on multilayer neural networks. *IEEE Transactions on Neural Networks*, 6(6):1446–1451, 1995.

E Gardner and B Derrida. Optimal storage properties of neural network models. *Journal of Physics A: Mathematical and general*, 21(1):271, 1988.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015a.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pp. 1135–1143, 2015b.

Ji He, Man Lan, Chew-Lim Tan, Sam-Yuan Sung, and Hwee-Boon Low. Initialization of cluster refinement algorithms: A review and comparative study. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 1, pp. 297–302. IEEE, 2004.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kyuyeon Hwang and Wonyong Sung. 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.

Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Minje Kim and Paris Smaragdis. Bitwise neural networks. *arXiv preprint arXiv:1601.06071*, 2016.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

Hao Li, Zheng Xu, Gavin Taylor, and Tom Goldstein. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*, 2017.

Darryl Lin, Sachin Talathi, and Sreekanth Annapureddy. Fixed point quantization of deep convolutional networks. In *International Conference on Machine Learning*, pp. 2849–2858, 2016.

Christos Louizos, Karen Ullrich, and Max Welling. Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems*, pp. 3290–3300, 2017.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

Asit Mishra, Eriko Nurvitadhi, Jeffrey J Cook, and Debbie Marr. Wrpn: Wide reduced-precision networks. *arXiv preprint arXiv:1709.01134*, 2017.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pp. 525–542. Springer, 2016.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115 (3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Charbel Sakr, Yongjune Kim, and Naresh Shanbhag. Analytical guarantees on numerical precision of deep neural networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3007–3016, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

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

Wonyong Sung, Sungho Shin, and Kyuyeon Hwang. Resiliency of deep neural networks under quantization. *arXiv preprint arXiv:1511.06488*, 2015.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Appendix

Table 1: CNN architectures for memorization task.

Model type	A	B	C	D	E	F
Architecture	Input - (32×32×1)					
	(3×3,4)×1	(5×5,4)×1	(5×5,4)×2	(3×3,4)×2	(3×3,8)×2	(3×3,8)×3
	3×3 max pool, stride 2					
	(3×3,8)×1	(5×5,8)×1	(5×5,8)×2	(3×3,8)×2	(3×3,16)×2	(3×3,16)×3
	3×3 max pool, stride 2					
	(3×3,16)×1	(5×5,16)×1	(5×5,16)×2	(3×3,16)×2	(3×3,32)×2	(3×3,32)×3
	3×3 max pool, stride 2					
	2-d fully connected					
# params	2018	4642	13070	5070	19066	31218
# params(conv)	1504	4128	12556	4556	18040	30192
Capacity (C)	4916.6	13944.8	38674.5	14878.0	53920.5	91712.9
C / param(conv)	2.483	3.09	2.99	3.00	2.86	2.96

A FCDNN, CNN AND RNN MODELS

Generic FCDNN, CNN, and RNN models are used for model capacity measurement and quantization performance estimation. The FCDNN consists of one or multiple fully connected layers. Each hidden layer, l, propagates a hidden vector h_l, by multiplying the weight matrix, $\mathbf{W}_l$, to the previous hidden vector, h_{l-1}, adding biases, b_l, and applying nonlinear activation, $\sigma_l(\cdot)$ as follows:

$$h_l = \sigma_l(\mathbf{W}_l h_{l-1} + b_l) \tag{5}$$

In FCDNNs, each weight matrix, W_l between two layers, demands $|h_{l-1}| \times |h_l|$ weights, where $|h|$ is the number of units for the layer, l.

CNNs, which are popular for image processing, usually receive 2-dimensional (D) or 3-D input data, whose size is much larger than the filter or kernel size. The set of weights between layers is referred to as the ‘kernel’ and the output is referred to as the ‘feature map’. Because the input size is usually much larger than the kernel size, the CNN parameters are reused many times. A kernel slides over the input feature map and produces an output feature map, and the sliding step is determined by the stride, s.

The convolution weights is denoted as $\mathbf{W}_l \in \mathbb{R}^{k_{l,h} \times k_{l,w} \times n_{l-1} \times n_l}$ and feature map of the layer as $\mathbf{C}_l \in \mathbb{R}^{c_{l,h} \times c_{l,w} \times n_l}$. $k_{l,h}$ and $k_{l,w}$ are height and width of each kernel and $c_{l,h}$, $c_{l,w}$ are height and width of the feature map in layer l, respectively. n_l is the number of feature map in the layer l.

CNNs can have a variety of structures according to the number of channels, the size of the kernels, and the number of layers. We attempted to produce a general setting for CNNs. The experiments models are shown in Table. 1. We constructed CNNs to have twice the number of feature maps and half the height/width after pooling. Also, to minimize side-effects by fully connected layers, the output feature of the last convolution layer is flattened and directly propagated to the $softmax$ layer. The capacity per parameter is measured only for parameters in convolution layers, by subtracting capacity of the fully connected layer.

RNNs have a feedback structure that reflects the information in the previous steps when processing sequence data. RNNs are composed of one or multiple recurrent layers, and each layer computes the output, y_t, and the hidden state, h_t, using the previous hidden state, h_{t-1}, and the input, x_t.

We use LSTM as the recurrent layers, showing stable performance in various applications.

B MUTUAL INFORMATION OF A NETWORK

The mutual information equation of a network can be obtained as follows.

We first re-write our random variables X, Y, and $\hat{Y}$.

$$X = \{X_1, ..., X_N\}, X_i \in (0,1)^{n_{in}}, \tag{6}$$

$$Y = \{Y_1, ..., Y_N\}, Y_i \in (0,1), \tag{7}$$

$$\hat{Y} = \{\hat{Y}_1, ..., \hat{Y}_N\}, \hat{Y}_i = f(\theta, X_i), \tag{8}$$

where $f(\theta, X_i)$ is the predict of a network when the input is X_i. Under our experimental setting, both X and Y have uniform random distribution. Note that X and Y are independent as well as Y_i and Y_j when $i \neq j$. Therefore,

$$P(Y_i|X_i) = P(Y_i) = 1/2, \tag{9}$$

$$H(Y) = H(Y_1, ..., Y_N) = NH(Y_1) = N. \tag{10}$$

And we use the network's average accuracy p as a probability of $Y_i = \hat{Y}_i$, so that

$$P(Y_i|\hat{Y}_i) = \begin{cases} p, & \text{if } (Y_i = \hat{Y}_i) \quad (11) \\ (1-p), & \text{otherwise} \quad (12) \end{cases}$$

Finally, the equation is derived as:

$$\begin{aligned} & I(Y; \hat{Y}|X) \\ & \quad = H(Y|X) - H(Y|\hat{Y}, X) \\ & \quad = H(Y) - H(Y|\hat{Y}) \\ & \quad = N - \sum_i H(Y_i|\hat{Y}_i) \\ & \quad = N - N\left(p \log_2 \frac{1}{p} + (1-p) \log_2 \frac{1}{(1-p)}\right) \end{aligned} \tag{13}$$

C Loss surfaces of fixed-point DNNs

The loss surfaces of fixed-point CNNs are shown in Fig. 8 and Fig. 9. The models are trained with CIFAR-10 dataset. The loss surfaces of RNN LMs for PTB dataset are also shown in Fig. 10 and Fig. 11. The performance degradation on test dataset is lower than the degradation on training dataset in all experiments.

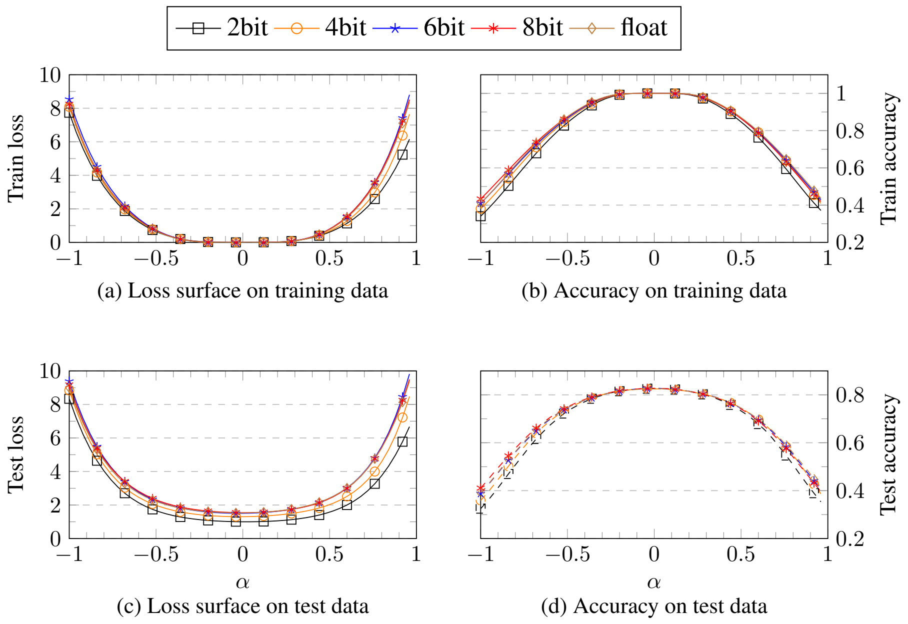


Figure 8: Loss surface of fixed-point *large* model trained with CIFAR-10 dataset. The top and bottom plots are the results for the training and test dataset, respectively.

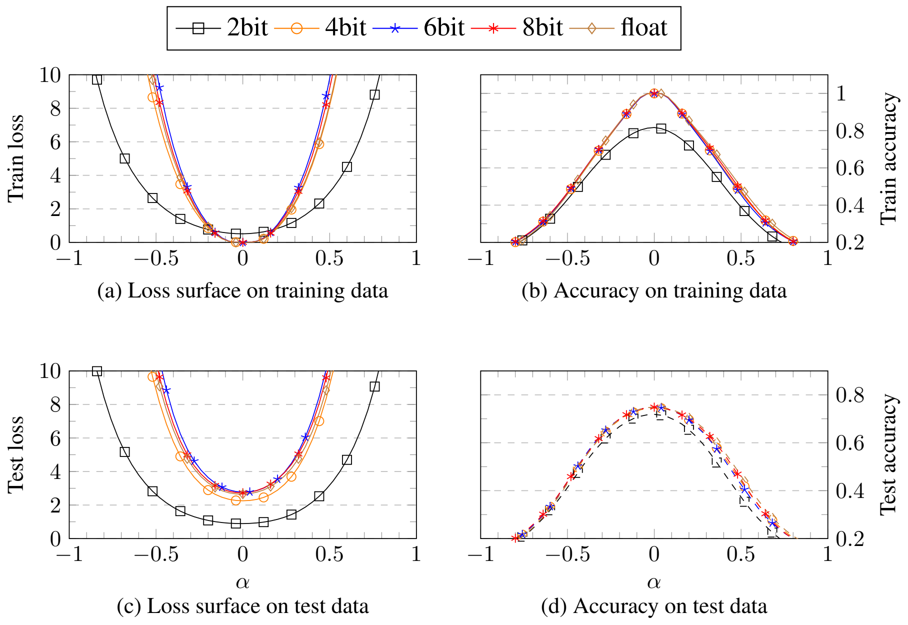
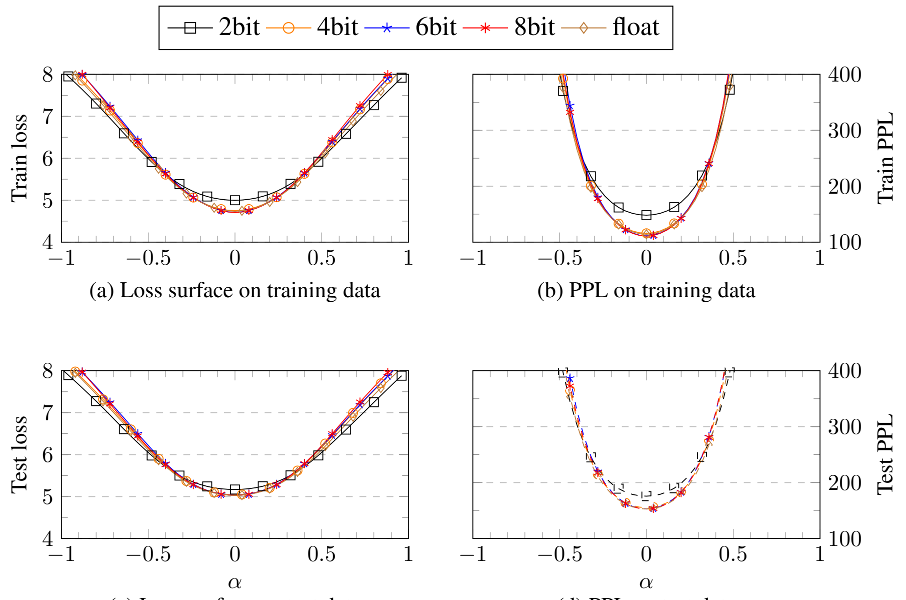


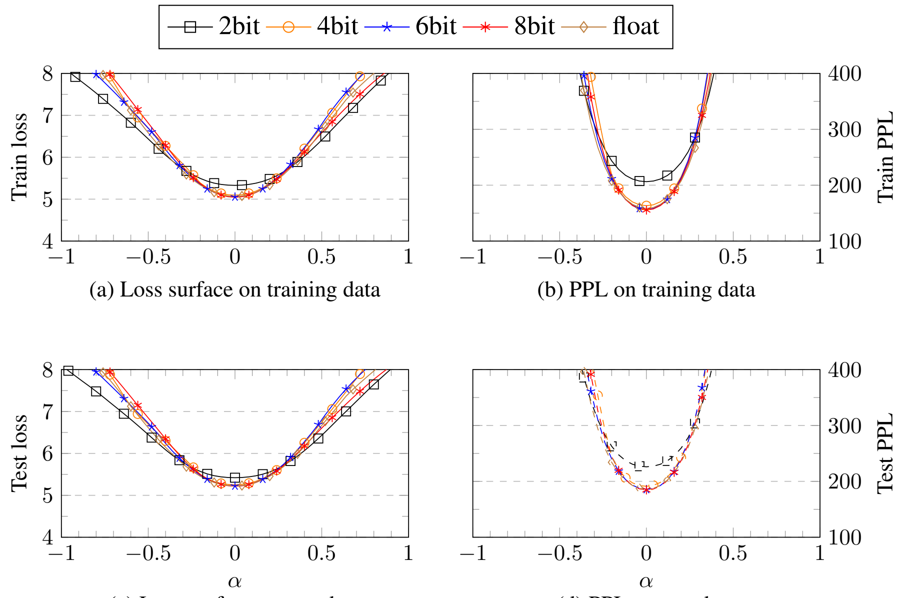
Figure 9: Loss surface of fixed-point *small* model trained with CIFAR-10 dataset.



(a) Loss surface on training data (b) PPL on training data

(c) Loss surface on test data (d) PPL on test data

Figure 10: Loss surface of fixed-point RNN LM with two 32-dimensional LSTM layers trained using PTB dataset.



(a) Loss surface on training data (b) PPL on training data

(c) Loss surface on test data (d) PPL on test data

Figure 11: Loss surface of fixed-point RNN LM with two 16-dimensional LSTM layers trained using PTB dataset.