Retraining Quantized Neural Network Models with Unlabeled Data

Kundjanasith Thonglek*, Keichi Takahashi*, Kohei Ichikawa*, Chawanat Nakasan†,
Hidemoto Nakada‡, Ryousei Takano‡ and Hajimu Iida*
* Nara Institute of Science and Technology, Nara, Japan
Email: {thonglek.kundjanasith.ti7, keichi, ichikawa}@is.naist.jp, iida@itc.naist.jp
† Kanazawa University, Ishikawa, Japan
Email: chawanat@staff.kanazawa-u.ac.jp
‡ National Institute of Advanced Industrial Science and Technology, Tsukuba, Japan
Email: {hide-nakada,takano-ryousei}@aist.go.jp

***Abstract*—Running neural network models on edge devices is attracting much attention by neural network researchers since edge computing technology is becoming more powerful than ever. However, deploying large neural network models on edge devices is challenging due to the limitation in available computing resources and storage space. Therefore, model compression techniques have been recently studied to reduce the model size and fit models on resource-limited edge devices. Compressing neural network models reduces the size of a model, but also degrades the accuracy of the model since it reduces the precision of weights in the model. Consequently, a retraining method is required to recover the accuracy of compressed models. Most existing retraining methods require the original labeled training datasets to retrain the models, but labeling is a time-consuming process. In particular, we cannot always access the original labeled datasets because of privacy policies and license limitations. In this paper, we propose a method to retrain a compressed neural network model with an unlabeled dataset that is different from the original labeled dataset. We compress the neural network model using quantization to decrease the size of the model. Subsequently, the compressed model is retrained by our proposed retraining method without using a labeled dataset to recover the accuracy of the model. We compared the proposed retraining method against the conventional retraining. The proposed method reduced the size of VGG-16 and ResNet-50 by 81.10% and 52.45%, respectively without significant accuracy loss. In addition, our proposed retraining method is clearly faster than the conventional retraining method.**

***Index Terms*—Deep Neural Network, Model Compression, Model Quantization, Model Retraining**

I. Introduction

Running neural network models on edge devices is attracting much attention by neural network researchers since edge computing technology has become more efficient of computing power than ever. There are several advantages of running neural network models on edge devices such as better privacy, less network bandwidth, and low-latency. This is because running models on edge devices does not require transferring the training and inference datasets between the edge devices and a centralized server. However, deploying large neural network models on edge devices is challenging due to the limitation in available computing resources and storage space [1].

Therefore, model compression techniques have been recently studied to reduce the model size and fit the model in resource-limited edge devices [2]. Compressing neural network models reduces the size of the model, but also degrades the accuracy of the model since it reduces the precision of weights in the models [3]. It is therefore important to minimize the loss of accuracy while trying to maximize the reduction of model size.

To recover the accuracy of compressed models, retraining is a necessary. Most existing retraining methods require the original labeled datasets to retrain the compressed models. A labeled dataset is a collection of samples that have been marked with labels identifying certain features of objects. Labeling is a crucial step in training neural network models [4], and is a time-consuming process [5]. In particular, we cannot always access the original labeled datasets because of privacy policies and license limitations. Using unlabeled datasets for retraining is highly useful when the original labeled dataset is unavailable.

In this paper, we propose a method to retrain a compressed neural network model with an unlabeled dataset that is different from the original labeled dataset. To retrain the compressed neural network, we use the outputs from the original model instead of using the original labeled dataset. The proposed method is able to reduce the size of the model without significant accuracy loss even when the original labeled data is not publicly available. We first compress the model using quantization to decrease the size of the model. Subsequently, the compressed model is retrained by our proposed retraining method to recover the accuracy of the model.

The rest of this paper is organized as follows. Section II describes related work on compression and retraining of neural network models. Section III explains our proposed methodology to compress and retrain a model to reduce the model size while retaining accuracy. Section IV shows the experimental results when applying our proposed method to real-world neural network models. Section V show the comparison between the proposed retraining method and the conventional retraining method. Section VI concludes this paper and discusses future work.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

TABLE I: Comparison of model compression techniques

Model compression technique	**Uses pre-trained models**	**Supports fully connected layers**	**Reduces redundant parameters**	**Impacts accuracy**
Parameter pruning and sharing	✓	✓	✓	✗
Low-rank factorization	✓	✓	✗	✓
Transferred/compact convolutional filters	✗	✗	✗	✗
Knowledge distillation	✗	✓	✓	✓

II. Background

This section gives a brief overview of existing techniques for compressing neural network models to reduce the size of models and retraining compressed neural network models to recover the accuracy of the models.

A. Compression of Neural Networks

Reducing storage space and computational cost of neural network models is necessary to compute models with more layers and nodes. Such expensive models are required to enhance the performance of large-scale data processing in real-time applications. Early works showed several techniques to decrease the size of neural network models. The challenge in model compression is understanding the structure of neural network models, finding which parameters are unnecessary or redundant, and reducing those parameters without sacrificing model accuracy.

Cheng et al. investigated different approaches for compressing neural network models and classified them into four approaches [6]: (1) parameter pruning and sharing, which reduce redundant parameters that are not critical for accuracy [7] (2) low-rank factorization, which uses matrix or tensor decomposition to estimate the informative parameters [8] (3) transferred/compact convolutional filters, which are special structural convolutional filters to save parameters [9], and (4) knowledge distillation, which trains a compact neural network with knowledge distilled from a larger model [10].

Table I shows the comparison of model compression techniques. Parameter pruning and low-rank factorization can support training from both pre-trained and scratch models while the transferred/compact filter and knowledge distillation techniques can only support training models from scratch. The main difference between parameter pruning and low-rank factorization is the impact on model accuracy [11]. In this paper, we focus on parameter pruning and sharing since we aim to deploy pre-trained models on resource-limited edge computing devices.

Network quantization is one method of parameter pruning and sharing approach. It prunes and shares the parameter in each layer of neural network models by quantizing the parameters. It aims to reduce the number of bits required to represent each weight parameter. Consequently, quantization reduces the size of neural network models, but also degrades the accuracy.

In a neural network, quantization means that parameters will be stacked into clusters. As a result, the parameters in the same cluster will share the same single value called *centroid* [12]. Vector quantization is applied to quantize multidimensional parameters, whereas scalar quantization is applied to quantize scalar values. Almost all of neural network models apply vector quantization to quantize their model since the stored parameters are multidimensional.

Zhao et al. improved quantization using outlier channel splitting. Channel is the dimension of data in each layer of neural network models. They proposed to quantize neural network models by duplicating channels of outliers and dividing each layer into two channels. Afterward, the model is still functionally identical, but the center of the parameter distribution in each layer is changed [13]. The size of the model was decreased because the center of the parameter distribution in each layer was changed to a smaller value. Outlier channel splitting takes time to compress models than quantization because this method needs to process every connection in a model.

Gong et al. found that vector quantization has a clear gain over existing matrix factorization methods when compressing densely connected layers, which are the most storage-demanding type of layers. They achieved 16–24$\times$ compression ratio with only 1% loss of accuracy [14]. Therefore, quantization is a promising compression technique due to its compatibility with densely connected layers and its low impact to the accuracy of models.

B. Retraining Compressed Neural Networks

To improve the accuracy of neural network models, it is important to recover the accuracy of compressed models. Retraining is a method to repeat the training process on an already trained model to further improve its accuracy.

Sung et al. showed that highly complex models can absorb the effects of severe weight quantization through retraining, whereas networks with limited number of connections cannot [15]. They confirmed effects on the resiliency of quantized networks by retraining method. Quantization of floating-point weights does not show efficient performance.

Lee et al. proposed layer-wise training, which is a technique to train a neural network model layer-by-layer [16]. Chen et al. introduced an algorithm to split the layers in a model to analyzer the structure in each layer [17]. Then, the weights in each layer are updated to minimize the final error of the model. They formulated and solved quantization of parameters as a discrete optimization problem. We also apply layer-wise training to retrain compressed neural network models because it reduces the number of trainable layers in the model to decrease the training time.

III. Methodology

This section describes the proposed retraining method to recover the accuracy of quantized neural network models. Figure 1 shows the overview of the proposed method. We first apply quantization to the model. Subsequently, the proposed retraining method is applied to the compressed model.

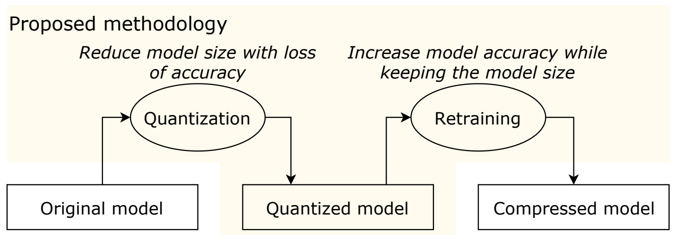
Proposed methodology
Reduce model size with loss of accuracy
Increase model accuracy while keeping the model size
Quantization
Retraining
Original model
Quantized model
Compressed model

Fig. 1: Overview of proposed methodology

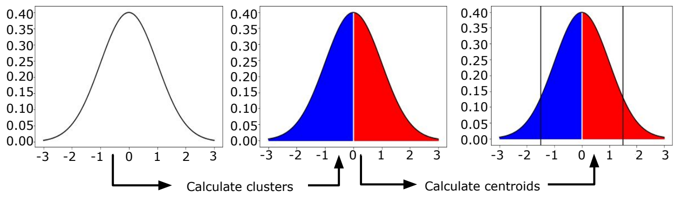

Fig. 2: Compression using quantization

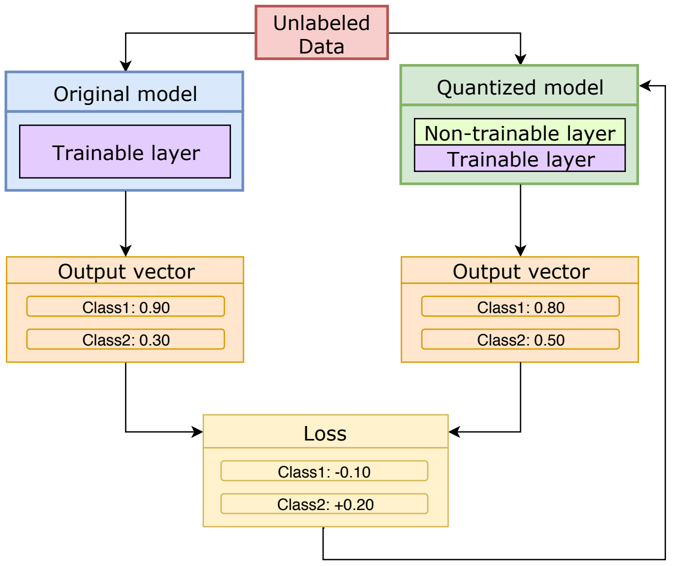

Fig. 3: Proposed retraining method

A. Quantization

Quantization is a lossy data compression technique where a range of data is grouped or binned into a single value or centroid. It is a nonlinear map that partitions the whole space and represents the values in each subspace by their centroid.

Figure 2 shows an example of quantization when the number of clusters is two. The parameter space is separated into two clusters such that each cluster has the same number of parameters. The centroid of each cluster is then calculated and the values in each cluster is represented using their centroid.

Each layer in a neural network model has a different number and distribution of parameters. Thus, different number of centroids must be chosen for each individual layer. To find the best number of centroids for each layer, we quantized each layer separately and balance between the reduced model size and the accuracy loss.

We varied the number of centroids using the powers of two from 1 to 256 because the number of centroids is based on the number of stored data bits. We defined the maximum number of stored data bit is 8 then the maximum number of centroids is 256.

The accuracy of a quantized model is an important indicator for deciding which layer to quantize. If the accuracy of the quantized model decreases significantly when quantizing a layer, it suggests that the layer contains important information that we should not eliminate. In many cases, deeper layers can be quantized without causing severe loss of accuracy, while the shallower layers cannot be quantized. We keep those non-quantized layers for the later retraining method.

The quantized model is then compressed using the Hierarchical Data Format 5 (HDF5) with its transparent gzip compression feature. Through the quantization process, the parameter values are grouped into limited number of centroids. Gzip compression finds those redundancies in the quantized data and reduce storage usage.

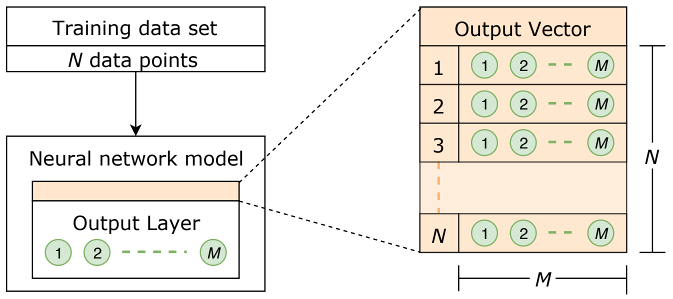

Fig. 4: Output vector structure in a neural network model

B. Retraining

After the quantization and compression processes, the compressed model is retrained to recover the accuracy without using the original labeled dataset. Figure 3 shows the overview of our proposed retraining method. We split the layers of the model into trainable layers and non-trainable layers. We keep the quantized layers as non-trainable layers because the unnecessary parameter values have been already eliminated in the quantized layers, and also the retraining method might be able to produce unnecessary parameter values that will just increase the model size without improving the model accuracy. We therefore choose only non-quantized layers as trainable layers for the retraining method. This approach is also helpful to shorten the retraining time since it takes longer if we handle more layers as trainable.

In our method, we leverage the outputs from the original model to retrain the quantized model instead of using the original labeled dataset. We supply the same unlabeled dataset to both the original model and the quantized model and acquire the output vectors from the two models. Figure 4 illustrates an output vector from the last layer of a model, which represents the probability of each output class, the size of output vector

TABLE II: Hardware specification

Hardware	Specification
CPU	Intel Xeon Gold 6148 ×2
Main Memory	384 GiB
GPU	NVIDIA Tesla V100 SXM2 ×4
GPU Memory	16 GiB

is *(number of samples in the training dataset)*×*(number of output classes)*. The weights in the trainable layers of the quantized model are then updated so as to minimize the loss between the output vectors from the original model and the quantized model. If the provided dataset covers enough variety of samples, the quantized model can be retrained with the outputs from original model as teacher signals. This method is highly effective since we do not need to prepare a new labeled dataset for retraining.

IV. Case Study

This section shows the experimental results when applying our proposed method to VGG-16 and ResNet-50. These two models achieve outstanding performance in image classification tasks [18]. The experiments were conducted using the computational resource of the AI Bridging Cloud Infrastructure (ABCI) [1] provided by the National Institute of Advanced Industrial Science and Technology (AIST). Table II shows the hardware specification of a compute node in ABCI.

Both models were trained using ImageNet, which is a dataset of images established for the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [19]. Although the ImageNet dataset is publicly available, we assume a situation where we cannot access the original labeled dataset for the retraining method, and we use another image dataset, the CIFAR dataset established by the Canadian Institute For Advanced Research [20] instead. Since the CIFAR dataset is compiled by a completely different organization than the original dataset, we can evaluate how our method works with such a different dataset.

A. Case Study of VGG-16 model

VGG-16 is a deep neural network model which won the first place in the ILSVRC 2014 classification task. It is a convolutional neural network designed for image classification tasks [21]. Even though this model has been exceeded by newer network architectures such as Inception or ResNet, it is still one of them most important neural network architectures for image classification tasks since most of the newer neural network architectures in computer vision is inspired by VGG-16.

1) Model analysis: Figure 5 shows the VGG-16 architecture. The 16 layers in the model can be divided into six blocks depending on the size of their inputs and filters.

We counted the number of weights and biases in each layer separately. Figure 6 shows the number of weights in each layer. It indicates that the number of weights in the last block

[1]https://abci.ai/

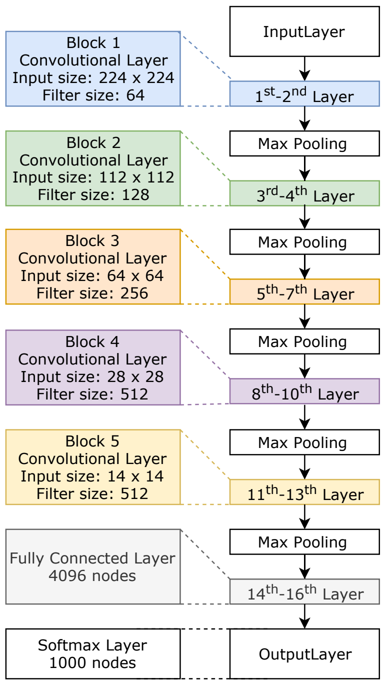

Fig. 5: VGG-16 model architecture

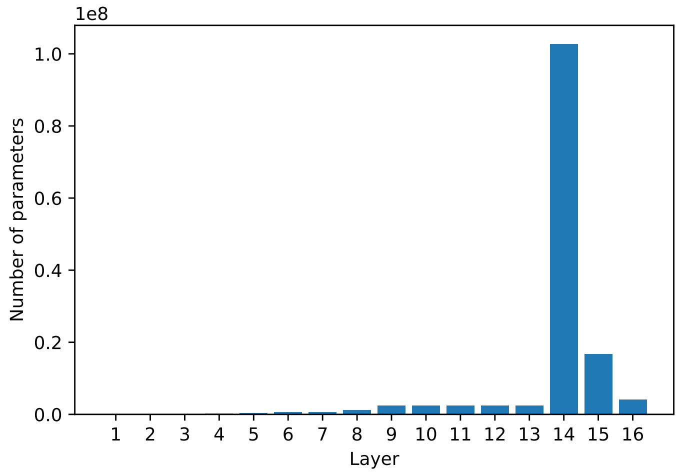

Fig. 6: Number of weights in VGG-16

occupies the majority of the entire model. The number of weights has the same trend as the biases, but the number of biases is less than the number of weights by 10^5 times and thus insignificant in terms of storage space.

2) Model quantization: Quantization was applied to both biases and weights with varying number of centroids.

Figure 7 presents the model size of the quantized VGG-16 models when quantizing the weights. We found that quantizing the weights in the 1^{st}–13^{th} layers and 16^{th} layer did not decrease the model size more than 15%. Quantizing the weights in the 14^{th} and 15^{th} layer reduced the model size by 20% and 65%, respectively. Therefore, we decided to quantize the weights in the 14^{th} and 15^{th} layer.

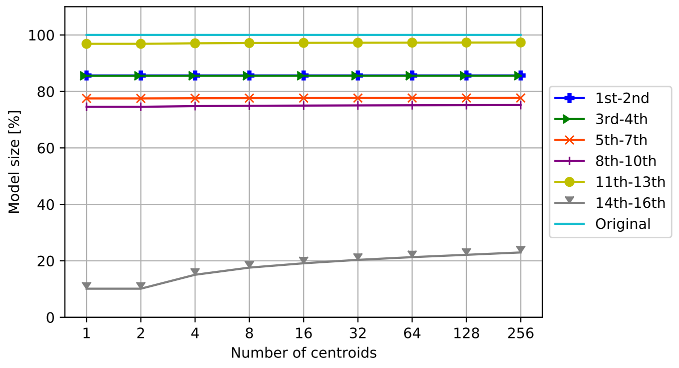

Fig. 7: Size of quantized VGG-16 models

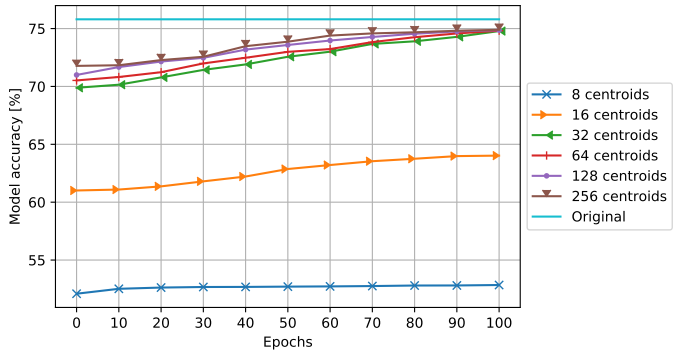

Fig. 9: Retraining quantized VGG-16 models

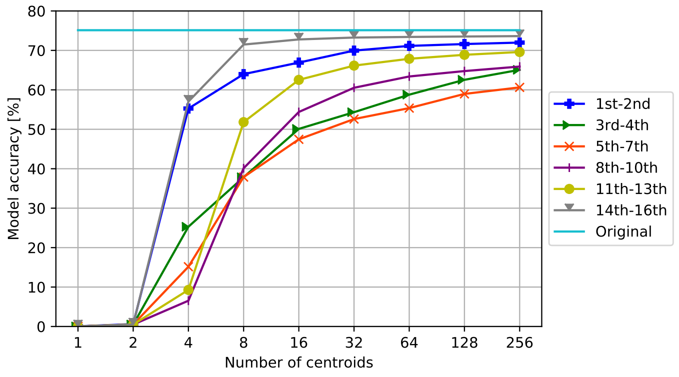

Fig. 8: Accuracy of quantized VGG-16 models

Figure 8 presents the model accuracy of the quantized VGG-16 models when quantizing the weights. We found that quantizing the weights in 14^{th} and 15^{th} layer at the same time decreases the accuracy more than 10% when the number of centroids is less than 8.

We found that quantizing the biases did decrease the model size for more than 15% of the original model. Also, quantizing the biases in any of the layers did not degrade the accuracy more than 5%.

Based on these observations, we decided to quantize the biases in all layers using one centroid and the weights in the 14^{th} and 15^{th}layers using 8–256 centroids.

3) Model retraining: Our retraining method was applied to VGG-16. Figure 9 presents the model accuracy of the quantized VGG-16 models our proposed retraining method for 100 epochs.

We found that using 16 or less centroids clearly degrades the model accuracy. The model accuracy is able to recovered retraining method due to the suitable number of centroids in model quantization remains the necessary parameter value in the model, which makes the model still has learning ability to improve the accuracy. The accuracy of quantized model converged to the accuracy of original model when using 32 or more centroids due to the number of centroids is sufficient to keep the necessary parameter value of learning ability in the model. Therefore, the suitable number of centroids is required for the efficient quantized model.

Quantizing the 14^{th} and 15^{th} layers using 32–256 centroids achieved nearly the accuracy of the original model (75.12%). Therefore, the best configuration for quantizing the VGG-16 model is to quantize the biases in all layer using one centroid and the weights in 14^{th} and 15^{th} layer using 32 centroids because it compressed to possible smallest model size without significant accuracy loss.

B. Case Study of ResNet-50 model

ResNet-50 is a deep neural network model which won the first place in the ILSVRC 2015 classification task. ResNet is a short name for Residual Network. As the name of the network indicates, the new terminology that this network introduces is residual learning [22]. Residual layer is an important component layer in the ResNet model. It enables residual learning, which is skipping over layers to solve gradient vanishing problem to reuse the parameter from the previous layer in the model.

1) Model analysis: There are three types of layers in ResNet-50 when classified by the number of parameters in a layer. The first type is the max pooling layer which does not have any parameter. The second type is the convolutional layer which has two parameters: (1) bias and (2) weight. The third type is the batch normalization layer which has four parameters: (1) alpha (2) beta (3) mean and (4) standard deviation. Figure 10 shows the ResNet-50 model architecture. The 50 layers in ResNet-50 can be divided into six blocks by the size of input and the size of filter. Eeach block has a max pooling layer followed by a batch normalization layer.

We counted the number of parameters in each layer separately. Figure 11 shows the number of weights in each layer. The number of weights is more than the number of biases, alpha, beta, mean and std by 10^4 times.

2) Model quantization: Quantization was applied to the biases, weights, alpha, beta, mean and std values in the ResNet-50 model.

Figure 12 presents the model size when quantizing the weights. We found that quantizing the weights in the 1^{st}–22^{nd} layers is not decreasing more than 20% of the original model, but quantizing the weights in block3 (23^{nd}–40^{th} layers) and block4 (41^{st}–49^{th} layers) makes the size of quantized model 20% smaller. Quantizing the weights in block3 and block4

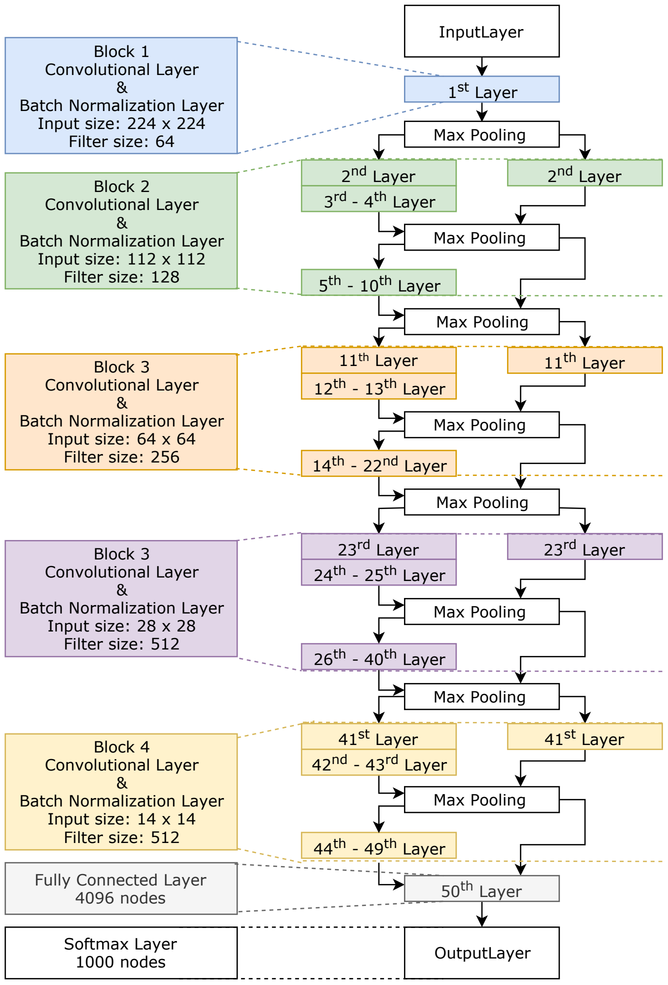

Fig. 10: ResNet-50 model architecture

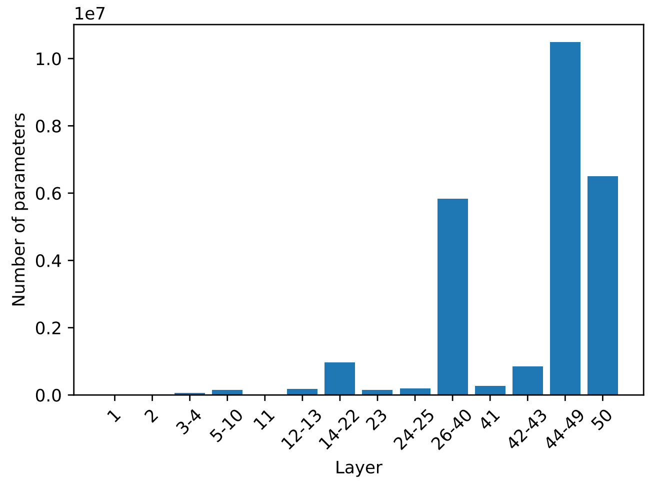

Fig. 11: Number of weights in ResNet-50

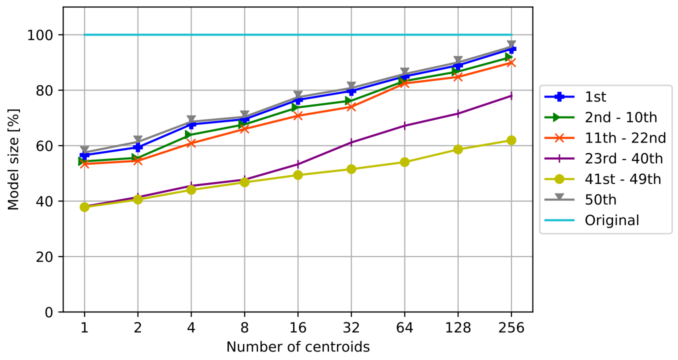

Fig. 12: Size of quantized ResNet-50 models

reducing the model size by 20% and 40%, respectively. Therefore, we focus on quantizing weights in block3 and block4. Quantized model is validated to calculate the accuracy of each quantized model.

Figure 13 presents the model accuracy when quantizing the weights. We found that quantizing the weight in block3 and block4 layer makes the accuracy of the quantized model is decreasing by more than 40% when the number of centroids is less than 16 centroids.

We found that quantizing the biases does not decrease the model size for more 10% of the original model. Also, quantizing the biases in the different layer makes the accuracy of quantized model is not decreasing more than 5% from the original model.

Quantizing bias, alpha, beta, mean and std value in each layer provides the same trend result of both model size and model accuracy. We found that quantizing alpha, beta, mean and std value does not decrease the model size more than 20%, but the model accuracy decreases more than 45%.

Based on these observations, we decided to quantize the biases in all layers using one centroid and the weights in block3 and block4 using 16–256 centroids except the batch normalization layers. The model accuracy when quantizing the block3 and block4 layers at the same time is reduced less than 30% and the model size is reduced more than 50%.

3) Model retraining: Our retraining method was applied to ResNet-50. Figure 14 presents the quantized ResNet-50 model accuracy when applying our proposed retraining method for 100 epochs.

We found that using 128 or less centroids clearly degrades the model accuracy. In addition, quantizing the 13^{rd}–49^{th} layers using 256 centroids, achieves the model accuracy close to the original model. Therefore, the best configuration to quantize ResNet-50 is to quantize the biases in all layer using one centroid and the weights in the 13^{rd}–49^{th} layers using 256 centroids.

V. Evaluation

We compared the proposed retraining method against the conventional retraining using VGG-16 and ResNet-50.

Figure 15 shows the improvement of the model accuracy when applying the proposed and conventional retraining methods to the quantized VGG-16 and ResNet-50 models for 100 epochs. The accuracy of the quantized VGG-16 model was 69.88% before retraining. The accuracy of the quantized VGG-16 model was improved to 74.78% and 74.89% after

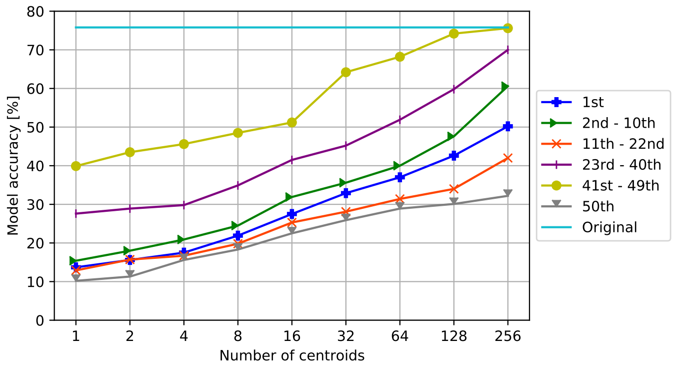

Fig. 13: Accuracy of quantized ResNet-50 models

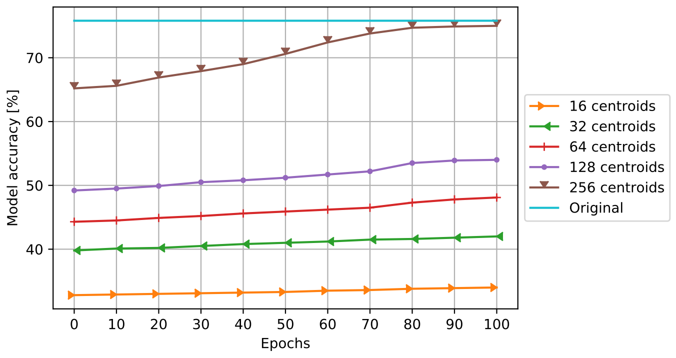

Fig. 14: Retraining quantized ResNet-50 models

applying the proposed and conventional retraining methods, respectively. As a result, the difference of accuracy between the retrained models was only 0.11%.

The accuracy of the quantized ResNet-50 model was 65.22% before retraining. The accuracy of the quantized ResNet-50 model was 75.09% and 75.12% after applying the proposed and conventional retraining methods, respectively. As a result, the difference of accuracy between the retrained models was only 0.03%.

The retraining time for the quantized VGG-16 and ResNet-50 models are shown in Fig. 16. It indicates that the proposed retraining method takes 85.46% less time than the conventional retraining when applied to VGG-16. Also, our proposed retraining method takes 82.49% less time than the conventional retraining when applied to ResNet-50. The proposed retraining method is clearly faster than conventional retraining method.

VI. Conclusion

In this paper, we proposed a novel retraining method for compressed neural network models to reduce the size of neural network models without significant accuracy loss. We designed and implemented our proposed retraining method technique which does not require labeled dataset as a training dataset. Vector quantization is applied to compress neural network model for decreasing model size. We proposed a retraining method that trains both the original model and the compressed model without using a labeled dataset. The output vectors from

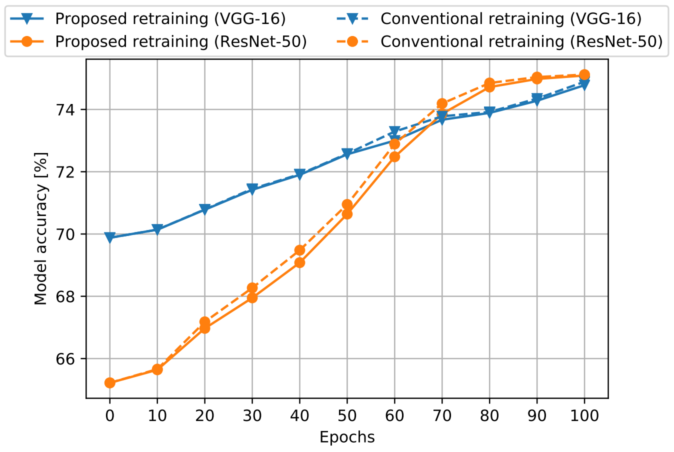

Fig. 15: Accuracy of quantized model through retraining

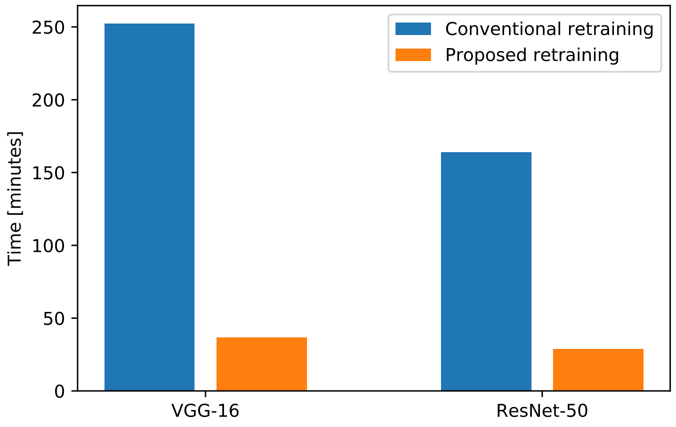

Fig. 16: Retraining time of quantized model

the both models were used as a training dataset to retrain the last layer of the compressed neural network model.

We used VGG-16 and ResNet-50 to evaluate our proposed methodology. The model size of VGG-16 was reduced by 81.10% with only 0.34% loss of accuracy. The difference of accuracy after applying the conventional and proposed retraining was only 0.11%. ResNet-50 was reduced by 52.54% with only 0.71% loss of accuracy. The difference of accuracy after applying the conventional and proposed retraining method with only 0.03%.

A future work is to apply the proposed retraining method to other neural network models. In addition, the structure of other neural network models should be investigated to conduct the efficient retraining method. Moreover, we will try to apply compression techniques other than quantization before the proposed retraining.

Acknowledgement

This work was partly supported by JSPS KAKENHI Grant Number JP18K11326.

References

[1] J. Kim, Y. Park, G. Kim, and S. J. Hwang, "SplitNet: Learning to semantically split deep networks for parameter reduction and model parallelization," in *Proceedings of the International Conference on Machine Learning*, Aug. 2017, pp. 1866–1874.

[2] Y. Wang, C. Xu, C. Xu, and D. Tao, "Beyond Filters: Compact feature map for portable deep model," in *Proceedings of the International Conference on Machine Learning*, Aug. 2017, pp. 3703–3711.

[3] M. A. Carreira-Perpinan and Y. Idelbayev, "Learning-Compression algorithms for neural net pruning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 8532–8541.

[4] H. Chen, K. Chuang, and M. Chen, "On data labeling for clustering categorical data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 11, pp. 1458–1472, Nov. 2008.

[5] Y. Zhu, J. T. Kwok, and Z. Zhou, "Multi-Label learning with global and local label correlation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 6, pp. 1081–1094, Jun. 2018.

[6] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," *CoRR*, vol. abs/1710.09282, Oct. 2017.

[7] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient ConvNets," in *Proceedings of the International Conference on Learning Representations*, Apr. 2017, pp. 1–13.

[8] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," in *Proceedings of the British Machine Vision Conference*, Sep. 2014, pp. 1–13.

[9] Y. Chen, N. Wang, and Z. Zhang, "DarkRank: Accelerating deep metric learning via cross sample similarities transfer," in *Proceedings of the AAAI Conference on Artificial Intelligence*, Oct. 2018, pp. 2852–2859.

[10] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *Proceedings of the NIPS Deep Learning and Representation Learning Workshop*, Mar. 2015.

[11] J. Kim, M. Lee, J. Kim, B. Kim, and J. Lee, "An efficient pruning and weight sharing method for neural network," in *Proceedings of the IEEE International Conference on Consumer Electronics-Asia*, Oct. 2016.

[12] X. Lu, H. Wang, W. Dong, F. Wu, Z. Zheng, and G. Shi, "Learning a deep vector quantization network for image compression," *IEEE Access*, vol. 7, pp. 118 815–118 825, Aug. 2019.

[13] R. Zhao, Y. Hu, J. Dotzel, C. D. Sa, and Z. Zhang, "Improving neural network quantization without retraining using outlier channel splitting," in *Proceedings of the International Conference on Machine Learning*, Jun. 2019.

[14] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," *CoRR*, vol. abs/1412.6115, Dec. 2014.

[15] W. Sung, S. Shin, and K. Hwang, "Resiliency of deep neural networks under quantization," *CoRR*, vol. abs/1511.06488, Nov. 2015.

[16] D. Lee and B. Kim, "Retraining-based iterative weight quantization for deep neural networks," *CoRR*, vol. abs/1805.11233, May 2018.

[17] S. Chen, W. Wang, and S. Pan, "Deep neural network quantization via layer-wise optimization using limited training data," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 3329–3336, Jul. 2019.

[18] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *CoRR*, vol. abs/1605.07678, Apr. 2016.

[19] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[20] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 11, pp. 1958–1970, Nov. 2008.

[21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, Sep. 2014.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2015, pp. 770–778.