Compressing a convolution neural network based on quantization

Dmitry Pertsau
Electronic Computing Machines department
Belarusian State University of Informatics and Radioelectronics
Minsk, Belarus
ORCID: 0000-0003-3830-378X

Marina Lukashevich
Post Doctoral Reseacher, Electronic Computing Machines department
Belarusian State University of Informatics and Radioelectronics
Minsk, Belarus
ORCID: 0000-0003-1255-5612

Dziana Kupryianava
PhD student, Electronic Computing Machines department
Belarusian State University of Informatics and Radioelectronics
Minsk, Belarus
ORCID: 0009-0007-8259-0655

***Abstract*—Modern deep neural network models contain a large number of parameters and have a significant size. In this paper we experimentally investigate approaches to compression of convolutional neural network. The results showing the efficiency of quantization of the model while maintaining high accuracy are obtained.**

Keywords—convolution neural network, quantization, Quantization-Aware Training, Post-Training Static Quantization

I. Introduction

Overparameterized neural networks show significant performance in computer vision (CV), natural language processing (NLP), robotics and others domains (Fig. 1, 2). It is very important to find the balance between model size and inference time from one side and accuracy and generalization from the other side. In our research we concentrated on Convolution Neural Networks (CNN).

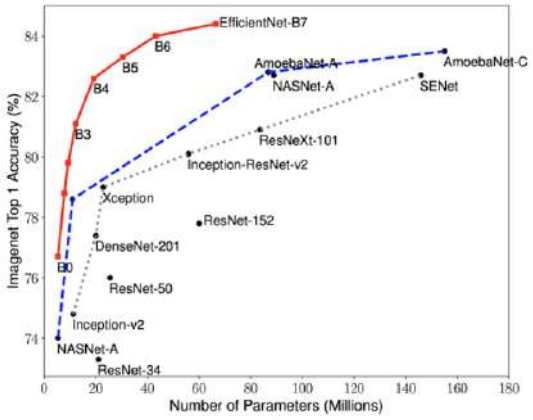

Fig. 1. Accuracy vs Number of Parameters for CNN architectures [1]

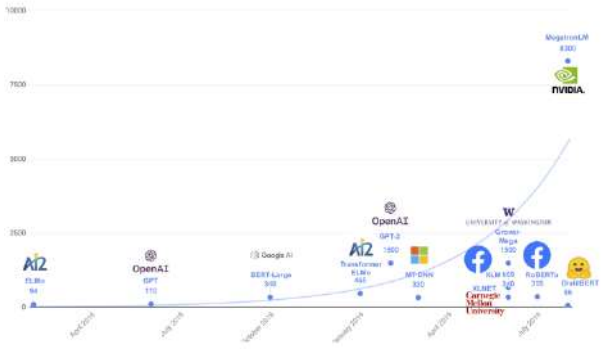

Fig. 2. Parameters vs Years for Transformers [2]

Compressing a deep neural network is an effective way to improve the efficiency of logical inference. Compression methods include the following approaches: parameter pruning [3, 4], low-rank factorization [5-7], weight quantization [8, 9] and knowledge distillation [10, 11]. Recently, quantization has become an important and very active research area in the efficient realization of computations related to neural networks [12, 13]. Network quantization compresses the network by reducing the number of bits per weight needed to represent the deep network. After quantization, the network can also demonstrate higher inference speed. The 32-bit floating-point format is dominant for deep learning applications, there is a gradual bias towards less accurate formats (e.g., INT8, FLOAT16 and others). This is due to many factors: storing a neural network model in a reduced-fidelity format requires less storage space; the use of processor blocks operating in integer arithmetic ensures higher instruction throughput theoretically; higher memory bandwidth and cache requirements are reached.

In the paper investigates the applicability of quantization algorithms to integer form for a convolutional neural network implementing handwritten digit recognition based on the MNIST dataset. The following scenarios are analyzed: quantization of the whole model to 8-bit integer format; quantization of convolution layers to 8-bit integer format; quantization of Dense layers to 8-bit integer format.

II. Approaches of Quantization

The quantization process consists of converting the trained neural network weights from 32-bit floating-point format to an alternative format, usually in reduced precision. More detailed description of the quantization process and its influence on neural network training is given in [12, 13].

Broadly, there are two kinds of quantization: weight quantization; weight and activation quantization.

The main difference is the following: whether the inverse conversion to floating point format is performed when applying the neural network model. In case of weight quantization, training process is in floating point format, quantization into integer format is performed when saving the model. When loading the model, weights are restored to floating point format. Further calculations are performed in floating point format. In the case of weight and activation quantization, training is also doing in floating point format, then weights and activation are quantized and stored. There is no backward conversion to floating point format when working with such a model.

The most well-known frameworks working with convolutional neural networks support 3 quantization models:

- *Dynamic Quantization*, where not only weights convert to int8, but also converting the activations to the same format before doing the computation. Theoretically, the computations will used special integer hardware blocks (like tensor blocks) for matrix multiplication and convolution, resulting in faster compute;
- *Post-Training Static Quantization* quantizes the weights and activations of the model. It fuses activations into preceding layers where possible and requires calibration with a representative dataset to determine optimal quantization parameters for activations;
- *Quantization-aware training (QAT)*, where all weights and activations are "fake quantized" during both the forward and backward passes of training: that is, float values are rounded to mimic int8 values, but all computations are still done with floating point numbers.

III. Experiments

A. Datasets

To set up the experiment, we used the MNIST dataset, consisting of 70 000 images, each of which has a resolution of 28x28 pixels in grayscale. The whole set is divided into 3 groups:

- training set (70% of the total number of images, 48 999 frames in total);
- validation set (20%, 14 000 images);
- test set (10%, 7 001 images).

For results validation we also used CIFAR-10 dataset (60 000 image, each of which has a resolution of 32x32 pixels in color).

B. Neural network architecture and parameters for training

Default neural network architecture (model name is 'default') is shown on Fig.3. There are 7 layers:

- convolution layer, where 32 kernels of 3x3 elements are used, activation function is RELU;
- max pooling with size 2x2;
- convolution layer, where 64 kernels of 3x3 elements are used, activation function is ReLU;
- max pooling with size 2x2;
- flatten;
- dropout layer with drop probability 0.5;
- dense layer with SoftMax activation function.

Training of the neural network was performed on a server with the following configuration:

- Intel(R) Xeon(R) CPU @ 2.20GHz;
- NVIDIA Tesla T4 / 15Gb.

Parameters for training:

- epoch count – 15;
- batch size – 128;
- loss function – categorical cross entropy.

Training and validation loss and accuracy for default model are shown on Fig.4. Accuracy is 98,9%. Confusion matrix is shown on Fig.5.

In our experiments we used the next technological stack: Python, TensorFlow, Keras, TensorFlow Lite, TensorFlow Model Optimization Toolkit.

C. Experiment 1. Neural network architecture with whole model quantization

For 'default' model we performed quantization and after that realized training process. The final model names 'quant'. In this case quantization performed using QAT approach.

For trained 'default' model we performed quantization and after that realized retrain process using the same data. The final model names 'quant_weights'. In this case quantization performed using Post-Training Static Quantization.

The obtained training results are shown in Table 1:

- without preloading the weight coefficients, a significant drop in recognition accuracy is observed (up to 83.43%);
- in the model with weight coefficients loading there appears an additional first layer - quantization layer;
- total volume of parameters increases for 'quant_weights' version.

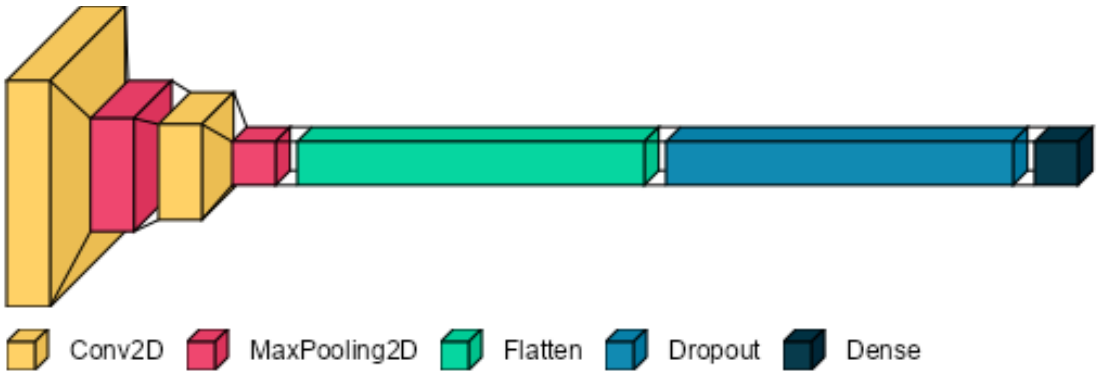

Fig. 3. CNN architecture

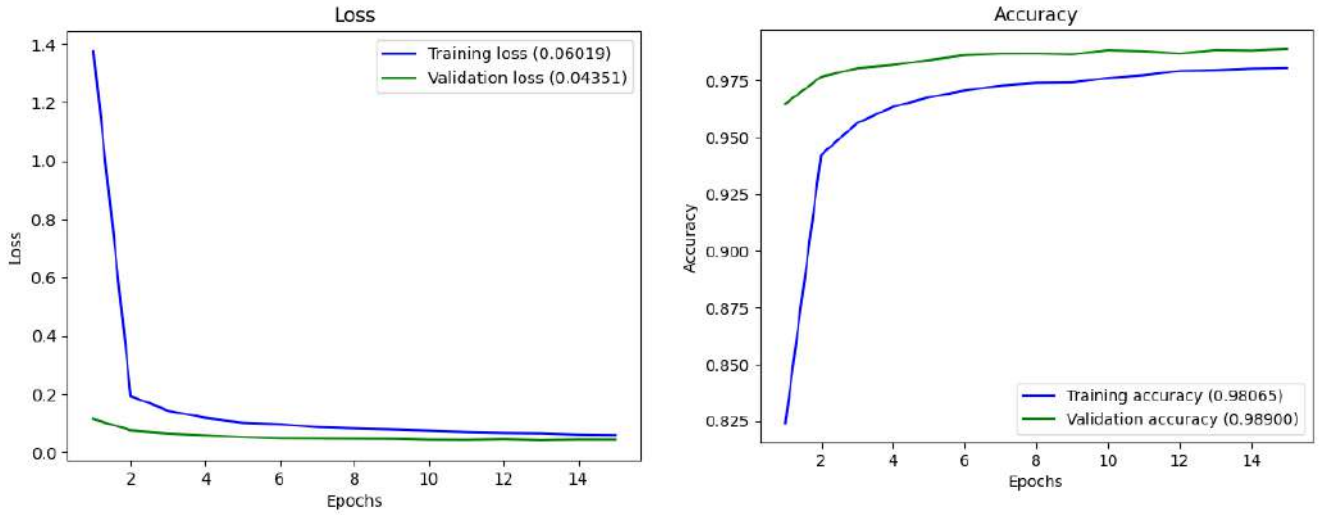

Fig. 4. Training and validation loss and accuracy for ‘default’ model

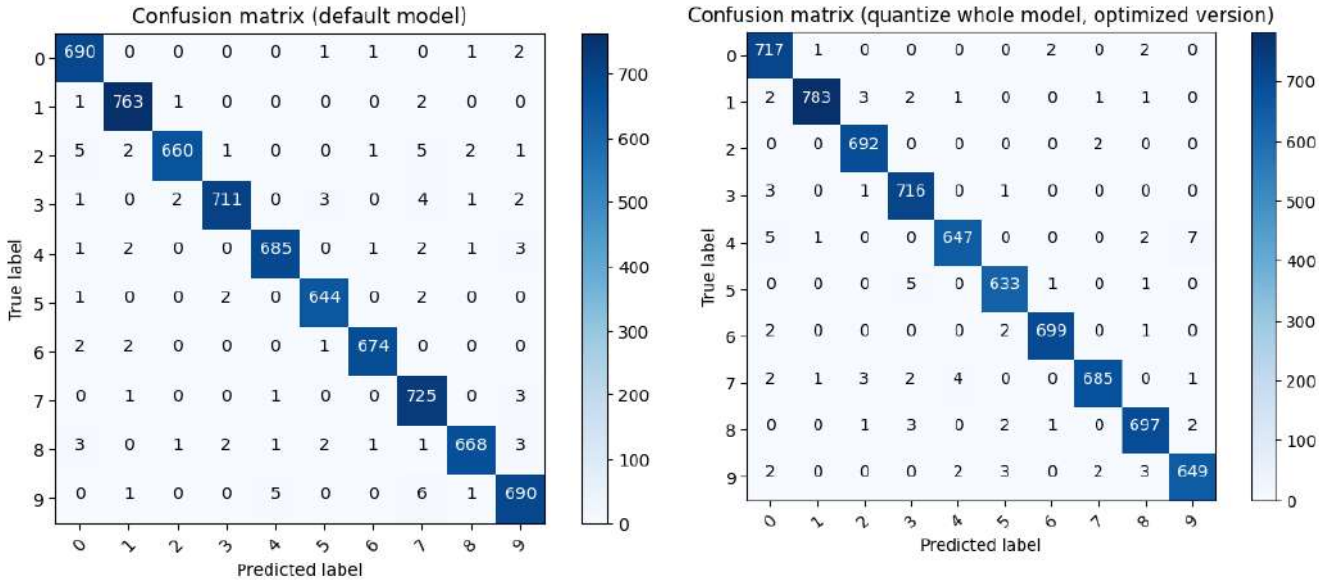

Fig. 5. Confusion matrix for ‘default’ and ‘quant_weights’ models (MNIST dataset)

D. Experiment 2. Neural network architecture with Dense layers quantization

For trained ‘default’ model we performed quantization only for dense layers and after that realized retrain process using the same data. The final model names ‘quant_dense’. Quantization performed using Post-Training Static Quantization approach.

E. Experiment 3. Neural network architecture with Conv2d layers qantization

For trained ‘default’ model we performed quantization only for convolution layers and after that realized retrain process using the same data. The final model names ‘quant_conv2d’. Quantization performed using Post-Training Static Quantization approach too.

We performed validation for experimental results using CIFAR-10 dataset (confusion matrix is on Fig.6, accuracy and model parameters are in Table 2). The results correlate with previously obtained results.

TABLE I. MODELS TEST RESULTS (MNIST DATASET)

Model name	**Accuracy**	**Parameters**		
		Total	***Trainable***	***Non-trainable***
default_model	98.69%	34826 (136.0 KB)	34826 (136.0 KB)	0 (0 Byte)
quant	83.43%	34826 (136.0 KB)	34826 (136.0 KB)	0 (0 Byte)
quant_weights	**98.95%**	35036 (136.9 KB)	34826 (136.0 KB)	210 (840 Byte)
quant_dense	98.85%	34832 (136.1 KB)	34826 (136.0 KB)	6 (24 Byte)
quant_conv2d	98.77%	35028 (136.8 KB)	34826 (136.0 KB)	202 (808 Byte)

After quantization, the effect of model compression (size reduction) was observed for ‘quant_weight’ model. The final volume size was 0.245 MB (250.9 KB).

The volume size of ‘default’ was 0.956 MB (978.9KB).

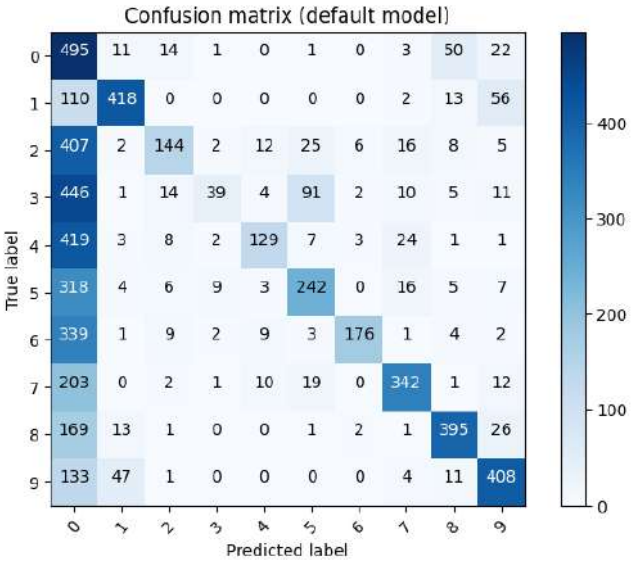

Fig. 6. Confusion matrix for default model (CIFAR-10 dataset)

TABLE II. MODELS TEST RESULTS (CIFAR-10 DATASETS)

Model name	Accuracy	Parameters		
		Total	***Trainable***	***Non-trainable***
default_model	61.38%	42442 (165.8 KB)	42442 (165.8 KB)	0 (0 Byte)
quant	10.06%	42652 (166.6 KB)	42442 (165.8 KB)	210 (840 Byte)
quant_weights	64.63%	42652 (166.6 KB)	42442 (165.8 KB)	210 (840 Byte)
quant_dense	65.03%	42448 (165.8 KB)	42442 (165.8 KB)	6 (24 Byte)
quant_conv2d	**65.95%**	42644 (166.6 KB)	42442 (165.8 KB)	202 (808 Byte)

At the same time, Tables 1 and 2 show that compressing the model does not affect its accuracy. And in the case of quantization of only convolutional layer one can get some increase of accuracy. This effect can be explained not by high complexity of the data set and requires additional research.

IV. CONCLUSION

In this paper, a quantization-based approach to convolutional neural network compression was considered. A simple network architecture was examined and experimental research on quantization of both the whole network and separate convolutional and fully connected layers was carried out. The experiments showed the effectiveness of this approach for reducing the model size of the neural network while preserving the required level of accuracy.

REFERENCES

[1] Mingxing Tan, and Quoc V. Le, “Efficientnet: Improving accuracy and efficiency through automl and model scaling”, 2019a. URL https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html

[2] Victor Sanh, “Smaller, faster, cheaper, lighter: Introducing distilbert, a distilled version of bert”, 2019. URL https://medium.com/huggingface/distilbert-8cf3380435b5.

[3] Song Han, Jeff Pool, John Tran, William J. Dally, “Learning both weights and connections for efficient neural network”, NIPS'15: Proceedings of the 28th International Conference on Neural Information Processing Systems, Vol.1, 2015, P. 1135–1143, DOI: 10.5555/2969239.2969366.

[4] Huizi Mao, Song Han, Jeff Pool, Wenshuo Li, Xingyu Liu, Yu Wang, William J. Dally, “Exploring the granularity of sparsity in convolutional neural networks”, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2017, P. 13-20, DOI: 10.1109/CVPRW.2017.241.

[5] Jian Xue, Jinyu Li, and Yifan Gong, “Restructuring of deep neural network acoustic models with singular value decomposition”, Interspeech, 2013, P. 2365-2369.

[6] Emily Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, Rob Fergus, “Exploiting linear structure within convolutional networks for efficient evaluation”, NIPS'14: Proceedings of the 27th International Conference on Neural Information Processing Systems, Vol.1, 2014, P. 1269-1277, DOI: 10.5555/2968826.2968968.

[7] Ross Girshick, “Fast R-CNN”, ICCV '15: Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), 2015, P. 1440-1448, DOI: 10.1109/ICCV.2015.169.

[8] Jiaxiang Wu; Cong Leng; Yuhang Wang; Qinghao Hu; Jian Cheng, “Quantized convolutional neural networks for mobile devices”, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, PP. 4820–4828, DOI: 10.1109/CVPR.2016.521.

[9] Ron Banner, Yury Nahshan, Daniel Soudry, “Post training 4-bit quantization of convolutional networks for rapid-deployment”, NIPS'19: Proceedings of the 33rd International Conference on Neural Information Processing Systems, 2019, Article 714, PP. 7950-7958, DOI: 10.5555/3454287.3455001.

[10] Cristian Buciluǎ, Rich Caruana, Alexandru Niculescu-Mizil, “Model compression”, KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, 2006, P. 535-541, DOI: 10.1145/1150402.1150464.

[11] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, “Distilling the knowledge in a neural network”, 2015, arXiv preprint, arXiv: 1503.02531.

[12] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, Kurt Keutzer, “A Survey of Quantization Methods for Efficient Neural Network Inference”, 2021, arXiv preprint, arXiv: 2103.13630.

[13] Binyi Wu, Bernd Waschneck, Christian Georg Mayr, “Convolutional Neural Networks Quantization with Attention”, 2022, arXiv preprint, arXiv: 2209.15317.