Research on Mixed-Precision Quantization and Fault-Tolerant of Deep Neural Networks

Zhaoxin Wang
Capital Normal University
Information Engineering College,
Beijing 100089 China
wangzhaoxin@cnu.edu.cn

Jing Wang*
Capital Normal University
Information Engineering College,
Beijing 100089 China
jwang@cnu.edu.cn

Kun Qian
Capital Normal University
Information Engineering College,
Beijing 100089 China
kunQian@cnu.edu.cn

ABSTRACT

As deep neural networks become more and more common in mission-critical applications, such as smart medical care, drones, and autonomous driving, ensuring their reliable operation becomes critical. The data in the hardware memory is susceptible to bit-flip due to external factors, which leads to a decrease in the inference accuracy of the deep neural network deployed on the hardware. We solve this problem from the perspective of the deep neural network itself, We use a reinforcement learning algorithm to search for the optimal bit width for the weights of each layer of the deep neural network. According to this bit width strategy, the deep neural network is quantified, which maximizes the limitation of data fluctuations caused by bit-flip and improves the fault-tolerance of the neural network. The fault-tolerance of the network model compared with the original model, the solution proposed in this paper improves the fault-tolerance of LeNet5 model by 8.5x , the fault tolerance of MobileNetV2 model by 15.6x , the fault tolerance of VGG16 model by 14.5x , and the accuracy decreases negligibly.

CCS CONCEPTS

• **Computing methodologies**; • **Machine learning**; • **Machine learning approaches**; • **Neural networks**;

KEYWORDS

Deep neural network, Mixed-precision quantization, Fault-tolerant, Reinforcement learning

ACM Reference Format:
Zhaoxin Wang, Jing Wang*, and Kun Qian. 2021. Research on Mixed-Precision Quantization and Fault-Tolerant of Deep Neural Networks. In *The 5th International Conference on Computer Science and Application Engineering (CSAE 2021), October 19–21, 2021, Sanya, China.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3487075.3487135

1 INTRODUCTION

In recent years, the classification accuracy of convolutional neural networks on the ImageNet dataset has surpassed that of humans. Due to the superior performance of deep neural networks, deep neural networks have been applied to various key fields, such as national defense, health, finance and control fields.

When deep neural networks are deployed on various hardware devices, there are hardware-related reliability problems, and this problem is currently not paid enough attention. The most common is the occurrence of soft errors in integrated circuits. Soft errors are transient errors in the circuit caused by various noises and interferences, such as temperature, voltage fluctuations and radioactive particle impact. With the reduction in the size of integrated circuit logic gates, the reduction of power supply voltage and the increase of circuit frequency, integrated circuits have become more and more sensitive to various disturbances, and the problem of soft errors in nano-integrated circuits has become more and more prominent. Soft errors may cause data errors, and may even cause the entire system to crash. Although the algorithms of deep neural networks have certain fault tolerance, such as the Relu function, BN algorithm and pooling layer that most deep neural networks have, although these algorithms have certain fault concealment capabilities, the facts have proved that these are still far not enough.

Many applications use the data type of single-precision floating-point numbers to represent these parameters, and the range that single-precision floating-point numbers can represent far exceeds the range required by the data [1]. Using fewer bits to represent data, that is, quantization, is a commonly used neural network compression technique, which can effectively reduce energy consumption and improve performance.

Recent studies have shown that the quantified neural network model has higher fault tolerance than the unquantified model [2]. The reason is that the quantized data has a small fluctuation range when subjected to bit inversion. The deep neural network model after binary quantization has the highest fault tolerance [3]. However, the accuracy of the quantized binary deep neural network model is more obvious than that of the full-precision deep neural network model [4].

We propose a deep neural network mixed-precision quantization method based on reinforcement learning, which improves the fault tolerance of the deep neural network model under the premise of ensuring the accuracy of the deep neural network model. When hardware equipment fails, the quantified deep neural network model can reduce the harm of these faults to the data, and then can ensure the actual accuracy of the deep neural network model. Improving the reliability of the applications that deploy these deep neural network models can reduce hardware costs and protect the safety of people's property.

CSAE 2021, October 19–21, 2021, Sanya, China
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8985-3/21/10…$15.00
https://doi.org/10.1145/3487075.3487135

2 RELATED WORKS

2.1 Fault Tolerance of Deep Neural Networks

At present, work has been conducted on the fault tolerance of deep neural networks. For example: FTT-NAS [5]. This work proposes a fault-tolerant neural network architecture search to automatically find a neural network architecture that is reliable for various faults in today's equipment. This scheme first uses the controller to sample different deep neural network structures from the search space, and then performs fault tolerance training on the sampled neural network structures to obtain candidate deep neural network models. Then verify the accuracy and fault tolerance on the hardware accelerator. Combine fault tolerance and accuracy to as a reward R, and finally use the reward R to update the controller to sample a deep neural network architecture with better fault tolerance and accuracy. Some work has proposed a technique based on the clipping activation function (FT-ClipAct) [6], which systematically defines the clipping threshold of the activation function to improve the ability of deep neural networks to recover from data failures, that is, the fault tolerance of deep neural networks. The specific steps are to first count the maximum activation of each layer of the deep neural network, then dynamically set the unbounded activation function to be bounded, and set the upper bound to ACT_{max}, finally fine-tune the ACT_{max}. When the data in the memory is bit inverted, the data has obvious deviation. After the error data passes through this activation function, the error data is adjusted. The larger propagation of errors is avoided, so the fault tolerance of the deep neural network model is improved.

2.2 Mixed-Precision Quantization Method of Deep Neural Network

With the development of variable bit-width accelerators [7], mixed-precision quantization is considered to be a more promising quantization method. In a deep neural network model, the importance or redundancy of different layers is different, and the sensitivity to quantization between similar different layers is also different. Using a uniform and aggressive quantization bit width for all layers of a deep neural network will seriously damage the accuracy of the deep neural network. Therefore, using different quantization bit widths for different layers of the deep neural network, that is, using mixed-precision quantization can solve the above-mentioned layer sensitivity. The problem of mixed-precision quantization can simultaneously take into account the fault tolerance and accuracy after quantization. In recent years, many works have been studying the mixed-precision quantification of deep neural networks. The most difficult point of this type of work is to find the optimal bit width allocation scheme for each layer. The search space grows exponentially with the depth of deep neural networks. According to how to find the optimal bit width for each layer, these tasks can be roughly divided into two categories. One is to extract the features of each layer of the deep neural network, and use the features of the layer to allocate the bit width [8] [9] [10]. The other is based on the automatic mixed-precision quantification of reinforcement learning [11] [12], which uses reinforcement learning agents to find the optimal bit-width allocation scheme. Both types of schemes can achieve the effect of mixed-precision quantization. The former relies on expert experience to allocate the depth of each layer of the deep neural network to easily reach the sub-optimal level of the model. The latter is based on reinforcement learning and the mixed-precision quantization scheme is fully automated and better. But they did not consider the impact of mixed-precision quantization on the fault tolerance of deep neural networks.

3 FRAMEWORK DESIGN

We propose an automatic mixed-precision quantization framework with fault tolerance as the main goal. The framework uses reinforcement learning algorithms to automatically determine the mixed-precision quantization strategy for various deep neural network models, that is, determine the optimal position for each layer of the deep neural network model. The quantization strategy is customized according to different deep neural networks to achieve the purpose of improving the fault tolerance of the deep neural network model, and the accuracy loss is negligible.

3.1 Method Overview

As shown in Figure 1, firstly, input the unquantized deep neural network model and the available set of quantized bit widths into our framework, the models are represented by 32-bit floating point numbers. Then the reinforcement learning agent selects the bit width within our pre-defined range according to the state of each layer of the current deep neural network, and then quantifies the deep neural network model with mixed-precision according to the selected bit width and our quantitative formula, verifies the accuracy and fault tolerance on the verification set. Regarding the combination of accuracy and fault tolerance as a reward for the reinforcement learning agent, the agent is motivated to take actions that maximize the reward, that is, to choose the best bit width for each layer. When the maximum number of iterations is reached, output the optimal bit width of the model. After verification, this model has better fault tolerance than the previously unquantified model, and the accuracy loss is negligible.

3.2 Fault Tolerance Verification Framework

The fault tolerance verification method we proposed in 3.1 Evaluating the fault tolerance of the unquantified deep neural network model and the quantified model respectively. The first is to evaluate the fault tolerance of the unquantified deep neural network, and obtain the fault tolerance value of the model. The fault tolerance verification process is shown in Figure 2. First, the model uses a random method to inject faults into the bits of the each layer's weights. Then test the accuracy of the deep neural network after injecting the fault on the verification set. After we inject the fault, the deep neural network may have a small loss of accuracy, because the deep neural network itself has a certain fault tolerance, but it may also have a large loss of accuracy, such as a sudden drop in accuracy of 50%. When the accuracy loss is small, we continue to increase the BER(bit error rate) for fault injection until the accuracy loss suddenly exceeds 50%. We regard the value of the BER where the accuracy suddenly drops by more than 50% as the fault tolerance of the model. The mixed-precision quantitative model is also used for fault injection in the same way as above, and finally the fault tolerance value of the deep neural network model after quantization

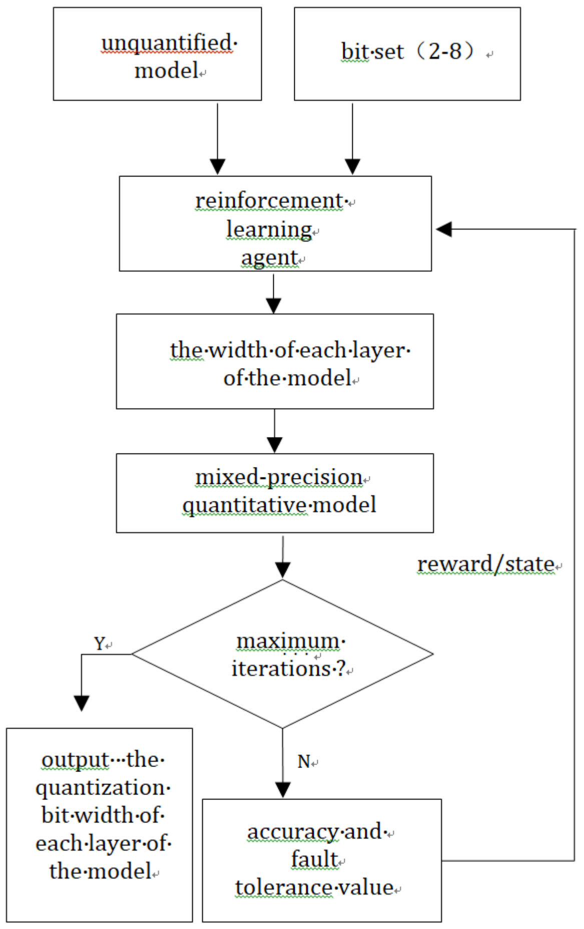
unquantified model
bit set（2-8）
reinforcement learning agent
the width of each layer of the model
mixed-precision quantitative model
reward/state
maximum iterations ?
Y
N
output the quantization bit width of each layer of the model
accuracy and fault tolerance value

Figure 1: An Overview of Our Framework.

of mixed precision is obtained. Experimental comparison found that the fault tolerance of several deep neural network models after using our method has been improved.

3.3 Reinforcement Learning Algorithm

We choose DDPG (Deep Deterministic Policy Gradient) as the reinforcement learning algorithm used in our experiment. It is an algorithm to solve continuous control problems [13], which is composed of Actor network and Critic network. The state space of our reinforcement learning algorithm is based on the observation of the characteristic state of each layer of the deep neural network to obtain a multi-dimensional vector. As shown in Formula 1 and Formula 2, L represents the layer of the network, Cin represents the number of input channels in this layer, $Cout$ represents the number of output channels, $Stride$ represents the convolution step size, and $KernalSize$ represents the size of the convolution kernel . We set the $KernalSize$ of the fully connected layer to 1, and the $Stride$ to 0. $WeightSize$ represents the number of weights of this layer, $FeatSize$ represents the number of feature maps of this layer, and A_L represents the actions of the L layer. We normalize the value range of these vectors to [0, 1] so that they are in the same range.

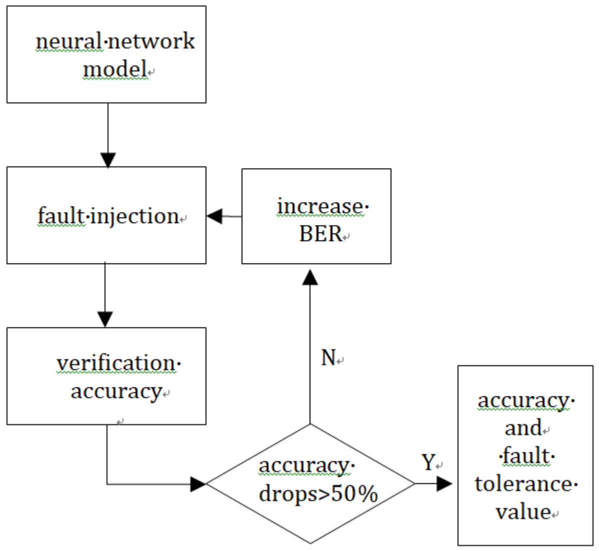

Figure 2: An Overview of Our Accuracy and Fault-Tolerance Verification Framework.

By encoding these feature information, it is then used as the observation state of the reinforcement learning agent. The reinforcement learning agent makes actions by observing this information, that is, the selection of the weight width.

The state space of the convolutional layer:

$$S_L = (L, Cin, Cout, Stride, KernalSize, WeightSize,\ FeatSize, A_{L-1}) \tag{1}$$

The state space of the fully connected layer:

$$S_L = (L, Cin, Cout, 0, 1, WeightSize, FeatSize,\ FeatSize,\ \ A_{L-1}) \tag{2}$$

The range of the mixed-precision quantization bit width we designed is [2, 8], that is, B_{max} is 8 and B_{min} is 2. Because the DDPG algorithm is an algorithm for dealing with continuous action problems, the action space of the reinforcement learning algorithm we designed is [0, 1]. A_L represents the action of the reinforcement learning of the L-th layer, and B_L represents the quantization bit width of the L-th layer, and then formula 3 is used to map the action space to the range of the bit width.

$$B_L = round\,(B_{min} - 0.5\ +\ A_L * (B_{max}\ -\ B_{min} + 1)) \tag{3}$$

As shown in formula 4, ACC_{quant} represents the accuracy of the network model after quantization, and ACC_{origin} represents the initial accuracy of the network. $TOLE_{quant}$ represents the fault tolerance of the quantified network model. $TOLE_{origin}$ represents the unoptimized initial fault tolerance of the network model. γ is the optimization coefficient, and this value is set to 0.5. According to this reward function, the reinforcement learning agent is guided to update the action. After several rounds of iteration, a neural network model with high precision and high fault tolerance can be obtained.

$$R = \gamma\left(ACC_{quant}\ -\ ACC_{origin}\right) + (1-\gamma)\left(TOLE_{quant}\ TOLE_{origin}\right) \tag{4}$$

Table 1: Comparison of Fault Tolerance and Accuracy of LeNet Model after Applying Our Framework

Method	Bits of each layer	Acc	Tole
Unquantified	32 bits float	98.2%	2×10^{-5}
Ours	[2,4,3,4,4]	97.1%	1.7×10^{-4}

3.4 Design of Quantization Function

Experiments show that the weight of the deep neural network is mostly near 0, so the original weight is first scaled to [-1, 1]. After the deep neural network passes through the activation layer, the value is intercepted to a number greater than 0, so the activation value is scaled to [0, 1]. According to the optimal bit width B of each layer obtained in 3.3, quantization formula 4 is used to quantize the weight. Use Equation 5 to quantify the activation value. The weight is signed, so the first bit is reserved as the sign bit, and the rest are digital bits. The integer bit defaults to 0. The activation value is unsigned, and all B bits represent values. W_i represents an unquantized floating-point number, and W_k represents a quantized fixed-point number. A_i represents the activation value of the floating-point number before quantization, and A_k represents the activation value of the fixed-point number after quantization.

$$W_k = \frac{1}{2^{B-1}-1} round\left(\left(2^{B-1}-1\right) * W_i\right) \tag{5}$$

$$A_K = \frac{1}{2^{B}-1} round\left(\left(2^{B}-1\right) * A_i\right) \tag{6}$$

4 EXPERIMENT

In the experiment, the LeNet5 model was tested on the MNIST, and the MobileNetV2 and VGG16 models were tested on the CIFAR10. The unquantized model uses 32-bit floating point numbers to represent weights and activation values. The comparison experiment performed mixed-precision quantization and fault injection on the weights, but the activation values of all layers were uniformly quantized to 8 bits. The experiment codes are all written with Pytorch library.

4.1 Experiment Results

As shown in Table 1, after training, the LeNet5 network model is tested on the MNIST data set and the accuracy is 98.2%. The original model is verified for fault tolerance, and the fault tolerance of the network model is 2×10^{-5}. After the optimization of our framework, the accuracy of the network model is 97.1%, and the fault tolerance of the network model is 1.7×10^{-4}. The fault tolerance of the model optimized by our method is 8.5x higher than that of the original model.

Table 3: Comparison of Fault Tolerance and Accuracy of VGG16 Model After Applying Our Framework

Method	Bits of each layer	Acc	Tole
Unquantified	32 bits float	93.5%	2.4×10^{-6}
Ours	[5,7,4,6,3,5,7,5,3,5, 3,5,3,5,4,5]	91.6%	3.5×10^{-5}

As shown in Table 2, the MobileNetV2 model is trained on the CIFAR10 data set, and inference on the test set shows that the network accuracy is 92.3%, and the fault tolerance of the model after fault injection is 2.5×10^{-6}. After the optimization of our method, the accuracy of the network model is 90.7%, and the fault tolerance of the model is 3.9×10^{-5}. The fault tolerance of the model optimized by our method is 15.6x higher than that of the original model.

As shown in Table 3, the unquantified VGG16 model is verified on the CIFAR10 data set with an accuracy of 93.5% and a fault tolerance of about 2.4×10^{-6}. The fault tolerance of the model after optimization by our method is about 3.5×10^{-5}. The accuracy of the network model is 91.6%, and the fault tolerance of the VGG16 model optimized by our method is 14.5x higher than that of the original model.

Based on the above data, we can conclude that the fault tolerance of the network model using our proposed mixed-precision quantification scheme is improved compared with the original network model. After the network model parameters are quantified as a fixed-point number using the mixed-precision of our proposed scheme, when the parameter is bit flipped, the data fluctuation changes little, so the impact of data failure on the network accuracy is reduced.

5 CONCLUSION

We propose a quantification framework with fault tolerance as the main goal. This framework is based on reinforcement learning to quantify deep neural networks with mixed- precision. Not only can the model be compressed, but the most important thing is to improve the fault tolerance of deep neural networks by limiting the fluctuation of data bit flips. At present, only some data sets and deep neural networks have been made. In future work, we hope to add the IMAGENET data set and more complex networks to prove our conclusions. This framework can be used in various deep neural networks and has strong universal applicability. Ensure the reliability of the deployment of deep neural network model hardware.

ACKNOWLEDGMENTS

This work is supported by National Natural Science Foundation of China (61772350).

Table 2: Comparison of Fault Tolerance and Accuracy of MobileNetV2 Model after Applying Our Framework

Method	Bits of each layer	Acc	Tole
Unquantified	32 bits float	92.3%	2.5×10^{-6}
Ours	[6,6,5,4,6,6,8,6,4,8,6,6,8,6,5,8,6,6,8,6,3,6,6,6,7,5,8,7,3,6,6,4,6,7,6,6,8,6,6,7,6,6,5,6,6,5,6,6,6,6,5,6,6,6]	90.7%	3.9×10^{-5}

REFERENCES

[1] Zhang J J, Liu K, Khalid F, *et al.* (2019). Building Robust Machine Learning Systems: Current Progress, Research Challenges, and Opportunities[C]// the 56th Annual Design Automation Conference 2019.

[2] Santos F, Lunardi C, Oliveira D, *et al.* (2019). Reliability Evaluation of Mixed-Precision Architectures[C]// 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA). IEEE.

[3] SABBAGH M, GONGYE C, FEI Y, *et al.* (2019). Evaluating Fault Resiliency of Compressed Deep Neural Net-works[C]// 2019 IEEE International Conference on Embedded Software and Systems (ICESS). IEEE.

[4] HUBARA I., COURBARIAUX M., SOUDRY D., *et al.* Binarized Neural Networks. In Proceedings of the 30th International Conference on Neural Information Processing Systems (Vol. 29, pp. 4107-4115).

[5] LI W, NING X G, CHEN X., WANG Y, *et al.* (2020). FTT-NAS: Discovering Fault-Tolerant Neural Architecture. In 2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC) (pp. 211-216).

[6] HOANG L, HANIF M A, SHAFIQUE M (2020) . FT-ClipAct: Resilience Analysis of Deep Neural Networks And Improving Their Fault Tolerance Using Clipped Activation[C]// 2020 Design, Automation & Test in Europe Con-ference & Exhibition (DATE).

[7] Judd P, Albericio J, Hetherington T, *et al.* (2017). Stripes: Bit-Serial Deep Neural Network Computing[C]// International Symposium in Microarchitecture (MICRO). IEEE.

[8] ZHU X, ZHOU W, LI H (2018). Adaptive Layerwise Quantization for Deep Neural Network Compression[C]// 2018 IEEE International Conference on Multimedia and Expo (ICME). IEEE.

[9] DONG Z, YAO Z, GHOLAMI A, *et al.* (2019). HAWQ: Hessian AWare Quantization of Neural Networks with Mixed-Precision[C]// 2019 IEEE/CVF International Conference on Computer Vision (ICCV). IEEE.

[10] CAI Y, YAO Z, DONG Z, *et al.* (2020). ZeroQ: A Novel Zero Shot Quantization Framework[C]// 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE.

[11] ELTHAKEB A T, PILLIGUNDLA P, MIRESHGHALLAH F, *et al.* (2020). ReLeQ: A Reinforcement Learning Ap-proach for Automatic Deep Quantization of Neural Networks[J]. IEEE Micro, PP(99):1-1.

[12] WANG K, LIU Z, LIN Y, *et al.* (2020). HAQ: Hardware-Aware Automated Quantization With Mixed Precision[C]// 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE.

[13] LILLICRAP T P, HUNT J J, Pritzel A, *et al.* (2016). Continuous control with deep reinforcement learning. In ICLR 2016: International Conference on Learning Representations.