Performance Oriented Error Correction for Robust Neural Networks

Kunping Huang[†], **Paul H. Siegel**[‡], **Anxiao (Andrew) Jiang**[†]
[†] CSE Department, Texas A&M University
[‡] ECE Department, University of California, San Diego

I. Introduction

When deep neural networks (DNNs) are implemented in hardware, their weights usually need to be stored in non-volatile memory devices. As noise accumulates in the stored weights, the DNN's performance will degrade. This work studies how to use error correcting codes (ECCs) to protect the weights. Different from classic error correction in data storage, the optimization objective is to optimize the DNN's performance after error correction, instead of minimizing the Uncorrectable Bit Error Rate (UBER) in the protected bits. That is, the error correction scheme is performance-oriented. A main challenge is that a DNN often has millions to hundreds of millions of weights, causing a large redundancy overhead for ECCs, and the relationship between the weights and the DNN's performance is highly complex. To address the challenge, we propose a *Selective Protection* (SP) scheme, which chooses only a subset of important bits for ECC protection.

The scheme can be evaluated based on the redundancy-performance tradeoff as follows. Let k_{total} denote the total number of bits used to represent the neural network's weights. Let k_{pro} denote the number of bits we protect with ECCs. Let the ECCs be (n, k) linear codes, where n denotes the codeword length and k denotes the number of information bits. Then the number of parity-check bits is $\frac{n-k}{k} \cdot k_{pro}$. We normalize it by k_{total}, and call it *redundancy* r, namely, $r = \frac{k_{pro}(n-k)}{k_{total}k}$. As for the performance of the neural network, for classification tasks (which this work focuses on), it usually refers to the classification accuracy, namely, the probability that the inputs are classified correctly.

To determine which important bits to protect and achieve an optimized tradeoff between ECC's redundancy and DNN's performance, we present an algorithm based on *deep reinforcement learning* (DRL). We compare the performance of our algorithm to a natural baseline scheme, where all layers of the neural network receive the same level of protection from ECCs. Experimental results verify that our proposed algorithm achieves substantially better performance. For example, when the neural network is ResNet-18 [1] and its weights are represented by bits using the IEEE-754 standard (i.e., the single-precision floating-point format), and when BER is 1% and BCH codes are used, the baseline scheme's classification accuracy drops very quickly once its redundancy r is below the threshold 0.04525. In comparison, our algorithm can decrease the corresponding threshold to 0.03879, which represents a reduction of 14.3% in the redundancy requirement. If the ECC approaches the Shannon capacity, this reduction can be further enlarged to 25.7%.

II. Selective Protection Scheme

Before presenting the Selective Protection (SP) scheme, we first introduce two ways to represent weights by bits in DNNs: 1) the *IEEE-754 Standard Floating-Point Representation*: IEEE-754 is an international standard widely used in hardware. We adopt its 32-bit version. Given a weight $w \in \mathbb{R}$, let $B_w^{32} = (b_0, b_1, \cdots, b_{31})$ be its binary representation: $w = (-1)^{(b_0)_2} \times 2^{(b_1 b_2 \cdots b_8)_2 - 127} \times (1.b_9 b_{10} \cdots b_{31})_2$. Here b_0 is the *sign bit*, $b_1 b_2 \cdots b_8$ are the *exponent bits*, and $b_9 b_{10} \cdots b_{31}$ are the *fraction bits*. 2) the *Fixed-Point Representation*: in this representation, the weights in a range $[-c, c]$ are linearly quantized and represented as bits. (Such a representation has been used in neural networks before, including [2].) Consider its m-bit version. Let $s = c/(2^{m-1} - 1)$ be a scaling factor. Given a weight $w \in [-c, c]$, let $D_w^m = (b_0, b_1, \cdots, b_{m-1})$ be its binary representation: $w = (-1)^{(b_0)_2} \times (b_1 b_2 \cdots b_{m-1})_2 \times s$.

We now present the SP scheme, which selects important bits and protects them from errors with ECCs. Consider a neural network with N edge layers. For $i = 1, 2, \cdots, N$, let L_i denote the ith edge layer, and let W_i denote the set of weights in L_i. Assume that every weight is represented by m bits. The SP scheme will select a *bit-mask vector* $M_i = (\mu_{i,0}, \mu_{i,1}, \cdots, \mu_{i,m-1}) \in \{0, 1\}^m$ for each edge layer L_i. For each weight $w = (b_0, b_1, \cdots, b_{m-1}) \in W_i$, its jth bit b_j will be protected by ECC if $\mu_{i,j} = 1$. Let $\bar{r}$ be a target redundancy for the SP scheme, and let $\mathcal{P}$ be the DNN's performance (e.g., classification accuracy). The optimization objective of the SP scheme is to maximize $\mathcal{P}$ given that the actual redundnacy does not exceed $\bar{r}$.

We assume that the bits suffer from errors of a Binary Symmetric Channel (BSC) with Bit Error Rate (BER) p, and a suitable (n, k) linear ECC is used that can correct errors of BER p with a probability that approaches 1. And in our study, we focus on Convolutional Neural Networks (CNNs), which usually have two types of layers: convolutional layers and fully-connected layers. (The latter can be seen as a special case of a convolutional layer, where its convolutional kernel has the same size as its input feature map.)

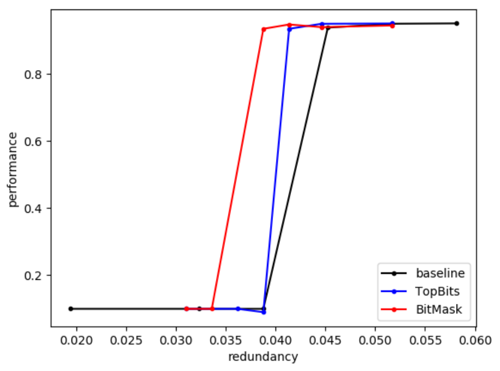
We present a deep reinforcement learning (DRL) algorithm for optimizing the bit-mask vectors. We first introduce the *state space*. For $i = 1, 2, \cdots, N$, let c_{in}^i (resp., c_{out}^i) be the number of input (resp. output) channels for the ith layer L_i, i.e., the number of input (resp., output) feature maps. Let s^i_{kernel} be its kernel size (i.e. the size of its filter for the convolution operation). Let s^i_{stride} be its stride for convolution. Let s^i_{feat} be the size of its input feature map (i.e., each input feature map is a two-dimentional array of size $s^i_{feat} \times s^i_{feat}$). Let a_i be the most recent action taken by the DRL algorithm for L_i (namely, which value was selected for its bit-mask vector). Let $\alpha_i = (c^i_{in}, c^i_{out}, s^i_{kernel}, s^i_{stride}, s^i_{feat}, |W_i|, a_i)$ denote a local state vector associated with L_i. In each iteration of the DRL algorithm, the N layers take their *actions* sequentially, using only their local state for efficiency. The *reward function* not only considers the performance of the neural network, but also the distance between the current redundancy r and the target redundancy $\bar{r}$. The DRL algorithm uses ideas of the DDPG algorithm [3] for robust learning.

Let the above method be called the *BitMask* method. For comparison, we also study its simplified version: the *TopBits* method, where each layer always chooses the first few bits of its weights for ECC protection, although the number of bits chosen by different layers can still be different.

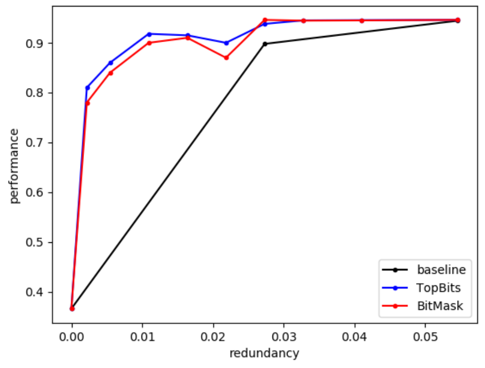
III. Experimental Evaluation and Analysis

We experimentally evaluate the Selective Protection scheme. The *redundancy-performance tradeoff* for ResNet-18, an important DNN for computer vision, is shown in Figure 1. (We also experimented with VGG16, another important DNN, and obtained similar results. So we skip its details here.) We compare the *BitMask* method and the *TopBits* method to an intuitive *baseline method*, which is a further simplification of *TopBits* by protecting the same number of bits in every weight.

It can be seen clearly that overall, both the *BitMask* method and the *TopBits* method significantly outperform the *baseline method*. It can also be seen that when the IEEE-754 representation is used, the *BitMask* method outperforms the *TopBits* method substantially overall. It is a very interesting observation because the *TopBits* method always chooses the first few bits of each weight, which are usually considered more significant than the remaining bits. It implies that the *BitMask* method can find Less Significant Bits (LSBs) that are more important than More Significant Bits (MSBs) for a DNN's overall performance. Some typical examples are shown in Figure 2. Our analysis shows that there are two important factors that determine how errors in a bit position affect the DNN's performance: (1) The 0-to-1 error and the 1-to-0 error have an asymmetric impact on the DNN's performance; for IEEE-754, the 0-to-1 errors change the absolute values of the weights more significantly than the 1-to-0 errors; (2) bits in the same position can have a highly imbalanced probability distribution. Those two factors lead to the conclusions that for the performance of DNNs, LSBs can be more important than MSBs for fault tolerance.



(a)



(b)

Fig. 1: The redundancy-performance tradeoff for the Selective Protection scheme. Here the DNN is ResNet-18, the dataset is CIFAR-10, BER $p = 0.01$, and "performance" is measured as the average classification accuracy of the DNN. (a) The data representation scheme is IEEE-754, and the ECC is the (8191, 6722) BCH code, which can correct 115 errors. (b) The data representation scheme is the fixed-point representation, and the ECC is the (8191, 6787) BCH code, which can correct 110 errors.

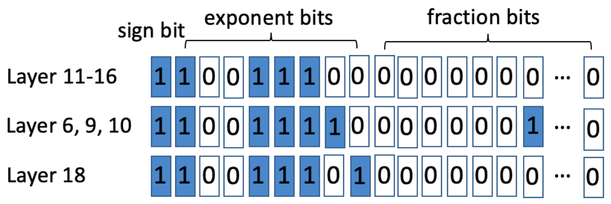


Fig. 2: Typical examples of the bit-mask vector in some edge layers, with the IEEE-754 floating-point representation and the *BitMask* method. The positions of the selected bits for ECC protection correspond to the 1's in the bit-mask vector (of the blue color). Notice that among the exponent bits, some less significant bits are selected instead of more significant bits.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[2] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han. HAQ: hardware-aware automated quantization. *CoRR*, abs/1811.08886, 2018.

[3] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.