Margin-Maximization in Binarized Neural Networks for Optimizing Bit Error Tolerance

Sebastian Buschjäger*, Jian-Jia Chen, Kuan-Hsun Chen, Mario Günzel,
Christian Hakert, Katharina Morik, Rodion Novkin,
Lukas Pfahler*, Mikail Yayla*
**These authors contributed equally*
Technical University of Dortmund
{sebastian.buschjaeger, jian-jia.chen, kuan-hsun.chen, mario.guenzel,
christian.hakert, katharina.morik, rodion.novkin, lukas.pfahler, mikail.yayla}@tu-dortmund.de

***Abstract*—To overcome the memory wall in neural network (NN) inference systems, recent studies have proposed to use approximate memory, in which the supply voltage and access latency parameters are tuned, for lower energy consumption and faster access at the cost of reliability. To tolerate the occuring bit errors, the state-of-the-art approaches apply bit flip injections to the NNs during training, which require high overheads and do not scale well for large NNs and high bit error rates. In this work, we focus on binarized NNs (BNNs), whose simpler structure allows better exploration of bit error tolerance metrics based on margins. We provide formal proofs to quantify the maximum number of bit flips that can be tolerated. With the proposed margin-based metrics and the well-known hinge loss for maximum margin classification in support vector machines (SVMs), we are able to construct a modified hinge loss (MHL) to train BNNs for bit error tolerance without any bit flip injections. Our experimental results indicate that the MHL enables the possibility for BNNs to tolerate higher bit error rates than with bit flip training and, therefore, allows to further lower the requirements on approximate memories used for BNNs.**

***Index Terms*—Neural networks, error tolerance, approximate memory**

I. Introduction

In the last years, numerous fields have benefited from the application of neural networks. One of the biggest challenges in neural networks (NNs) up to date is, however, the resource demand. NNs rely on deep architectures and a large amount of parameters to achieve high accuracy.

Recent studies on efficient NN-based inference systems have explored the use of approximate memory, which has been realized by reducing the memory supply voltage and tuning latency parameters with the goal of lower power consumption and faster access. If these methods are pushed to the limit, high bit error rates (BERs) can occur. For modern memory technologies, such as volatile memories (SRAM [10], [18], DRAM [8]) and emerging non-volatile memories (e.g. STT-RAM [6], [14], RRAM [5]) the BER increases steeply when reducing the voltage and tightening the timing. Without countermeasures, these bit errors can degrade the accuracy of NNs to unacceptable levels.

This paper has been supported by Deutsche Forschungsgemeinschaft (DFG) as part of the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Analysis" (project number 124020371) and project OneMemory (project number 405422836).

The survey in [16] by Huitzil et al. provides a comprehensive overview of the recent and further back work about fault and error tolerant NNs, from which we summarize some representative studies. For example, the study in [3] proposes a penalty term which aims at distributing the computation to neurons optimally to achieve error tolerance. Another study [2] distributes the absolute values of weights evenly to neurons, while the work in [13] aims at low weight importance.

Currently, the only known method to achieve bit error tolerance on the part of NNs is training with bit flip injections according to the error model. Bit flip injection during training, however, has disadvantages. First, recent studies have reported that injecting bit flips during training can significantly degrade accuracy. The higher the BER during training, the more significant the accuracy degradation [1], [5], [8]. Another disadvantage is the additional overhead [9]. During the training with bit flip injection, for every bit of the error-prone data a decision has to be made whether to inject a bit flip, which adds numerous additional steps in the NN training.

Achieving bit error tolerance in NNs without bit flip injection, and thus, conquering the above disadvantages, would be a breakthrough for the research area of NNs using approximate memory. To achieve this, the principles of bit error tolerance in NNs need to be well understood. However, to the best of our knowledge, the theoretical foundations of NN bit error tolerance have not been reported yet.

In binarized NNs, however, a resource-efficient variant of NNs [7], weights are represented by one bit. The binarization of the parameters in BNN allows the exploration of bit error tolerance to examine the effect of each bit flip precisely.

Our contributions: The key focus of this work is to explore methods to *achieve bit error tolerance without bit flip injection* in BNNs. We propose margin-based metrics that measure the bit error tolerance of structural elements in BNNs and use them for bit error tolerance optimization by transforming to a margin maximization problem. This allows us to adopt existing methods from support vector machines (SVMs) to solve the bit error tolerance problem. If the margins in the NN are not distorted, the insights and methods we gain by researching BNNs in this study may also be applicable to quantized or floating point precision NNs. Specifically, our contributions are as follows:

- In Section III, we provide a margin-based bit error tolerance metric for single neurons in a BNN, which formally characterizes when a neuron flips its output value. This metric is further propagated to the output layer of the BNN to quantify the bit error tolerance of the output layer, which is used to quantify the ultimate impact of bit flips.
- Based on the margin-based output layer metric and the well-known hinge loss for maximum margin classification in SVMs, we propose a modified hinge loss (MHL) in Section IV for bit error tolerance optimization of BNNs, which works without bit flip injections during training.
- We performed extensive experiments to compare with the-state-of-the-art approaches, which train BNNs with cross entropy loss (CEL) and bit flip injections [5]. The results show that applying the MHL alone (without any bit flip injections) outperforms CEL in terms of accuracy. We further evaluated the combination of MHL and bit flip injections, which significantly improves the accuracy of BNNs at high BERs.

II. System Model and Problem Definition

Fully connected and convolutional NNs consist of layers that perform operations, e.g. a convolution of the input data with the weights. Outputs are fed into an activation function, and the result is supplied to the next layer as a new input. The layer-wise process is repeated until the final output is obtained. We call this way of computing the forward pass. The convolution and the activation is computed using $\sigma(\sum_i W_i^l X^{l-1})$, where W_i^l is the weight of the i-th filter in layer l, X^{l-1} is its input and σ the activation function.

In BNNs, which binarized weights and activations, the convolution and activation can be simplified to

$$2 * popcount(XNOR(W_i^l, X^{l-1})) - \#bits > s,$$

where $popcount$ counts the number of 1s, $\#bits$ is the number of bits in the XNOR operands, and s is a learnable threshold parameter, the comparison against which yields a binarized output [7], [12].

A. Training

Stochastic gradient descent (SGD) with mini-batches is typically employed to train NNs. We denote the training data as $\mathcal{D} = \{(x_1, y_1), \dots, (x_I, y_I)\}$ with $x_i \in \mathcal{X}$ as the inputs, $y_i \in \mathcal{Y}$ as the labels, and $\ell\colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ as the loss function. With $f_W(x)$ as the NN output, the goal is to solve

$$\arg\min_W \frac{1}{I} \sum_{(x,y)\in\mathcal{D}} \ell(f_W(x), y) \tag{1}$$

by a mini-batch SGD strategy using backpropagation. A method to train BNNs has been proposed by Hubara et al. [7], in which weights are stored as floating point numbers for updating the model parameters in the backpropagation. During the forward pass, the weights and activations are deterministically binarized.

BNNs can be trained for bit error tolerance by bit flip injections during training [5]. This enables the possibility to use approximate memory, trading reliability for energy efficiency and performance.

B. BNN Building Blocks

In this work we consider fully connected BNNs (FCBNNs) and convolutional BNNs (CBNNs) performing classification. The layer types we use are: 1) convolutional, 2) maxpool, 3) batch normalization, and 4) fully connected layer. Our theoretical exploration does not need any specific properties of these layers. We present more detailed information on the used BNN architectures in the experiments section.

C. Memory Model

In this study, we assume that bit error rates (BERs) are transient and symmetric, i.e., the probability for 0 to flip to 1 is the same as the probability for 1 to flip to 0. This assumption characterizes the probability of bit flips every time when a bit is read from the approximate memory. This matches the assumptions in recent studies about approximate memories (SRAM [4], [15], [17], DRAM [8]), and non-volatile memories (RRAM [5], MRAM or STT-RAM [6]).

Distinct from the existing results in the literature based on bit flip injections, our training approach based on the modified hinge loss (detailed in Section IV) does not have to consider the BERs in the training phase. Therefore, the trained bit error tolerance of the BNN is not bound to a specific error rate. However, if the BER is specified, the BNN can be trained to target this rate by combining the modified hinge loss in Section IV and bit flip injections.

D. Problem Definition

Given a set of labelled input data, the objective is to train a BNN for high accuracy and high bit error tolerance, provided that the BERs are transient. In this work, we focus on the problem of *how to train BNNs for bit error tolerance without bit flip injections*.

To solve this problem, we explore bit error tolerance metrics in Section III, which allow us to describe how the bit error tolerance of BNNs can be measured with margins. Then, we modify the hinge loss for maximum margin classification (in SVMs) in Section IV to incorporate the metric and make it applicable to BNNs.

III. Bit Error Tolerance Metrics

In this section, we first introduce a margin-based neuron-level bit error tolerance metric for BNNs, which is extended to formulate a bit error tolerance metric for the output layer.

In the following, we use a notation describing properties of neurons in convolutional layers, but our considerations also apply to neurons in fully connected layers. Let n be the index of one neuron in a NN, and $x \in \mathcal{X}$ an input to the NN. The output of a neuron in a convolutional layer is a feature map with height U and width V. Let $h_{x,n,u,v} \in \mathbb{Z}$ be the pre-activation value of neuron n at place $(u, v) \in \{0, \ldots, U\} \times \{0, \ldots, V\}$, *before* applying the activation function. For BNNs, the pre-activation values of a neuron are computed by a weighted sum of inputs and weights that are ± 1. Therefore, one bit flip in one weight changes the pre-activation value by 2.

Theorem 1. *Let $n \in \{0, \ldots, N\}$ be the index of one neuron. Furthermore, let q be the number of bit flips induced into the weights of neuron n. The pre-activation of neuron n at place (u, v) after induction of these bit flips is in the interval $[h_{x,n,u,v} - 2q, h_{x,n,u,v} + 2q]$.*

Proof. For better readability, we use h for $h_{x,n,u,v}$ for abbreviation in the proof. Each bit flip of one weight of neuron n changes h by 2. Inductively, this shows that q bit flips change h by up to $2q$. Hence, the pre-activation of neuron n after up to q bit flips is in $[h - 2q, h + 2q]$. $\square$

We use this proof to first formulate a neuron-based bit error tolerance metric for hidden-layer neurons. On this basis, we formulate a metric for the entire output layer. We define the set of indices of neurons of the hidden layer layer by N_I and of the output layer by N_O.

For hidden layer neurons, i.e., those with index $n \in N_I$, the pre-activation value is compared with a threshold $s_n \in \mathbb{Z}$, which yields a binary output. Here, we use the following activation function:

$$\sigma: \quad h_{x,n,u,v} \mapsto \begin{cases} 1 & h_{x,n,u,v} > s_n \\ -1 & \text{else} \end{cases} \tag{2}$$

As long as the weights or input flips do not cause the pre-activation value to pass the threshold, a neuron is bit error tolerant. Therefore, the bit error tolerance of an hidden layer neuron depends on the margin

$$T_{x,n,u,v} = |h_{x,n,u,v} - s_n| \,. \tag{3}$$

between the pre-activation value and the threshold. With each bit flip $h_{x,n,u,v}$ can get closer to s_n and can finally flip the output of the activation, if s_n is passed. s_n is taken into account for activation shifts due to the batch normalization layer and without batch normalization $s_n = 0$.

Corollary 1. *Let $n \in N_I$ be the index of one hidden layer neuron. If $h_{x,n,u,v} > s_n$, then $\max\left(0, \left\lfloor \frac{T_{x,n,u,v}}{2} \right\rfloor - 1\right)$ many bit flips can be tolerated. Else, $\left\lfloor \frac{T_{x,n,u,v}}{2} \right\rfloor$ can be tolerated.*

Proof. We denote h for $h_{x,n,u,v}$, s for s_n, and T for $T_{x,n,u,v}$. We analyze the two cases individually.

If $h - s > 0$, then the output of neuron n is $+1$. We denote by $\tilde{h}$ the value of h after up to $q := \max\left(0, \left\lfloor \frac{T_{x,n,u,v}}{2} \right\rfloor - 1\right)$ bit flips. By Theorem 1, we have $\tilde{h} \in [h - 2q, h + 2q]$. We conclude that $\tilde{h} - s \geq h - 2q - s = T - 2q$ which is > 0 for q defined as above. More specifically, the output of neuron n is still $+1$.

On the other hand, if $h - s \leq 0$, then the output of neuron n is -1. Let $\tilde{h}$ be the pre-activation value after up to $q := \left\lfloor \frac{T_{x,n,u,v}}{2} \right\rfloor$ bit flips. Again, using Theorem 1 yields $\tilde{h} \in [h - 2q, h + 2q]$. We obtain $\tilde{h} - s \leq h + 2q - s = -T + 2q \leq 0$ and the output of neuron n is still -1. $\square$

We note that bit flips may propagate through the layers of BNNs. If the margin from Eq. (3) is small, weight flips might cause the neuron n to flip its output. This affects subsequent neurons for which neuron n provides inputs. A flip of the input value for a neuron affects the pre-activation value exactly as the bit error of a weight, i.e., it modifies the pre-activation value by 2. Therefore, the subsequent neurons may have to tolerate a higher number of bit errors.

A detailed analysis of the neuron-based bit error tolerance metric has been conducted in [1], showing the relation of this metric to the bit error tolerance of BNN. Using this metric for optimizing bit error tolerance has been reported to be unsuccessful. The reason may be that bit flips of neuron outputs can only affect the BNN prediction if the effect of bit flips reach the output layer and lead to a change of the predicted class. Therefore, we now shift our focus on applying the notion of margin presented above to the output layer, i.e., to neurons with index in N_O.

Each neuron in the output layer has only one output value $h_{x,n,1,1}$ which is one entry in the vector of predictions $\hat{y}$. No activation function is applied to the output value of these neurons. There are as many values in $\hat{y}$ as there are neurons in the last layer. The index of the entry with maximum value in $\hat{y}$ determines the class prediction, where we assume that ties are broken arbitrarily.

If bit errors modify the output values in the output layer such that another neuron provides the highest output value, then the class prediction changes. Let $h_{x,n',1,1}$ and $h_{x,n'',1,1}$ with $n', n'' \in N_O$ be the highest and the second highest output value of neurons in the output layer. The following corollary shows that the margin

$$m := h_{x,n',1,1} - h_{x,n'',1,1} \tag{4}$$

serves as bit error tolerance metric for the output layer.

Corollary 2. *If $m > 0$, then the output layer of the BNN tolerates $\max(0, \lfloor \frac{m}{2} \rfloor - 1)$ bit flips.*

Proof. Let $q \in \{0, \ldots, \max(0, \lfloor \frac{m}{2} \rfloor - 1)\}$ be a number of bit flips. We consider any distribution of the q bit flips to weights or inputs of the output layer, i.e., $\sum_{n \in N_O} q_n = q$ where q_n is the number of bit flips in weights or inputs of the neuron n.

Let n' be the index of the neuron with the highest output value. Furthermore, let $n \neq n' \in N_O$. For better readability, we denote $h_{n'}$ for $h_{x,n',1,1}$ and h_n for $h_{x,n,1,1}$. Furthermore, we denote by $\tilde{h}_{n'}$ and $\tilde{h}_n$ the values of $h_{n'}$ and h_n after $q_{n'}$ and q_n bit flips. By Theorem 1, we know that $\tilde{h}_{n'} \in [h_{n'} - 2q_{n'}, h_{n'} + 2q_{n'}]$ and $\tilde{h}_n \in [h_n - 2q_n, h_n + 2q_n]$. We conclude

$$\begin{aligned} \tilde{h}_{x,n',1,1} - \tilde{h}_{x,n,1,1} &\geq h_{x,n',1,1} - 2q_{n'} - h_{x,n,1,1} - 2q_n \\ &\geq h_{x,n',1,1} - h_{x,n'',1,1} - 2q = m - 2q > 0, \end{aligned}$$

as $q \leq \max(0, \lfloor \frac{m}{2} \rfloor - 1) < \frac{m}{2}$. Since $n \neq n' \in N_O$ is chosen arbitrarily and we have shown that $\tilde{h}_{x,n',1,1} > \tilde{h}_{x,n,1,1}$, the output value of neuron n' is still maximal even after q bit flips in the output layer. □

IV. Margin-Maximization for Bit Error Tolerance Optimization

In this section, we describe how we use the the margin-based bit error tolerance metric of the output layer and the well-known hinge loss for maximum margin classification to construct the modified hinge loss for optimizing the bit error tolerance of BNNs.

For bit error tolerance of the last layer, the margin m as introduced in Eq. (4) needs to be large, so that the maximum number of bit flips the output layer can tolerate is high. The margin can be directly computed by subtracting the second highest entry $\hat{y}_{c''}$ of the output vector $\hat{y}$ from the highest entry $\hat{y}_{c'}$, i.e., $m = \hat{y}_{c'} - \hat{y}_{c''}$. However, optimizing with respect to m without considering the other entries $\hat{y}_c$ of $\hat{y}$ may not exhaust the full potential of the margin between $\hat{y}_{c'}$ and the output of the other classes $\hat{y}_c$. The larger the margin between $\hat{y}_{c'}$ and $\hat{y}_c$ of other classes c, i.e. $m_c = \hat{y}_{c'} - \hat{y}_c$, the more bit errors can be tolerated in the neuron that calculates $\hat{y}_c$ without a change of the prediction. To put it concisely, for a bit error tolerant output layer, $\hat{y}_{c'}$ needs to be as large as possible, while the other $\hat{y}_c$ need to be as small as possible. To achieve this, we build upon the hinge loss for maximum margin classification.

The hinge loss, c.f. [11], for maximum margin classification is defined as

$$\ell(y, f) = \max(0, 1 - y \cdot f), \tag{5}$$

with the ground truth prediction $y = \pm 1$ and the prediction $f \in \mathbb{R}$. This loss becomes small if the predictions have the same sign as the predicted class and are close to 1 in magnitude. For predicted values larger than 1, the loss becomes 0. The "1" in the loss forces the classifier to maximize the margin between two class predictions. To solve optimizations problems that use the hinge loss, many of the common optimizers or algorithms can be used, such as the stochastic gradient descent (SGD) strategy [19].

In the case of BNNs for multi-class problems, the version of the hinge loss in Eq. (5) cannot be directly used. To extend the hinge loss to multiple classes, we define y_{enc} as a one-hot vector with elements in $\{-1, 1\}$, which has a $+1$ at the index with the ground truth, else -1. y_{enc} has the same number of elements as $\hat{y}$. Then the element-wise product $y_{enc} \cdot \hat{y}$ is computed. In this product, in case of correct predictions, positive predictions in the correct class will stay positive, negative predictions that should be as negative as possible become positive. In case of wrong predictions, i.e. high negative value for the correct class and high positive value for the wrong class, the values become negative. For a high penalty in the wrong case and a small penalty for the correct case, we subtract the product $y_{enc} \cdot \hat{y}$ from a parameter b, and get $(b - y_{enc} \cdot \hat{y})$. Since we do not demand higher prediction values than b, we set negative values to zero with the max function, and denote the modified hinge loss (MHL):

$$\ell_{MHL}(\hat{y}, y_{enc}) = max\{0, (b - y_{enc} \cdot \hat{y})\}. \tag{6}$$

Eq. (6) is still a convex function like Eq. (5), so it can be used with the same optimizers. In this work, for optimizing BNNs, the ℓ in Eq. (1) is replaced with ℓ_{MHL}, which optimizes the BNN via the mini-batch stochastic gradient descent (SGD) strategy to minimize the difference $(b - y_{enc} \cdot \hat{y})$, as described in Section II-A. The lower this difference, the larger the margin between $\hat{y}_{c'}$ and all the other $\hat{y}_c$. Above, we demanded exactly this property for a bit error tolerant output layer.

Name	# Train	# Test	# Dim	# classes
FashionMNIST	60000	10000	(1,28,28)	10
CIFAR10	50000	10000	(3,32,32)	10

TABLE I: Datasets used for experiments.

Parameter	Range
Fashion FCNN	In → FC 2048 → FC 2048 → FC 10
Fashion CNN	In → C64 → MP2 → BN → C64 → MP2 → BN → FC2048 → BN → FC10
CIFAR10 CNN	In → C128 → BN → C128 → MP2 → BN → C256 → BN → C256 → MP2 → BN → C512 → BN → C512 → MP2 → BN → FC1024 → BN → FC10

TABLE II: BNN architectures used in this work.

V. Experiments

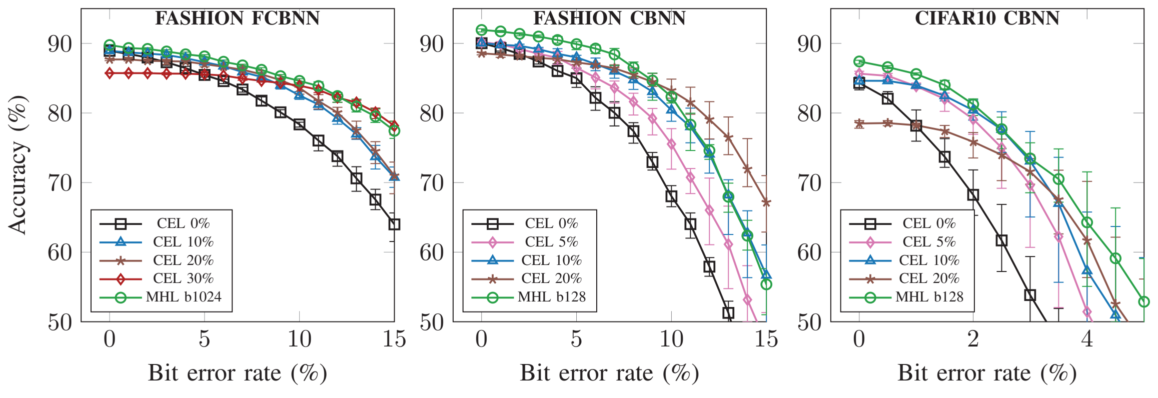
This section presents the results demonstrating the performance of BNNs optimized using MHL in comparison to the state-of-the-art cross entropy loss (CEL) [5]. The experiment setup is presented in Section V-A. We report the performance of the MHL without and with bit flip injection in Section V-B and V-C, respectively.

A. Experiment Setup

We evaluate three types of BNNs: Fully connected BNNs (FCBNNs) and small convolutional BNNs (CBNNs) for Fashion, and a larger CBNN for CIFAR10. The BNN architectures are presented in Table II and the dataset info is in Table I. The BNNs use convolutional (C) layers with size 3×3, fully connected (FC) layers, maxpool (MP) with size 2×2, and batch normalization (BN) layers followed by activation.

For training, we run the Adam optimizer [7] for 200 epochs for Fashion and 500 epochs for CIFAR10, with either cross entropy loss (CEL) or modified hinge loss (MHL). We use a batch size of 256 and an initial learning rate of 10^{-3}. To stabilize training we exponentially decrease the learning rate every 25 epochs by 50 percent.

To cover a wide spectrum of bit errors, for testing we use bit error rates (BERs) from 0% (no bit errors) up to 35%, with increments of either 1% for Fashion and 0.5% for CIFAR10. For training with bit flips we use different BERs, from 1% up to 30% BER, such that accuracy degradation is below 10% from the original accuracy. Depending on the



(a) Fashion FCBNN (b) Fashion CBNN (c) CIFAR10 CBNN

Fig. 1: Accuracy over bit error rate for BNNs trained with CEL under a given bit flip injection rate (specified in the legend, 0%, 5%, 10%, etc.) and BNNs trained with MHL without bit flip injections for a specified b in Eq. (6).

approximate memory and its properties, accepting BERs of this extent can improve the approximate memory, such as in energy consumption, timing parameters, production cost, etc.

As the base line to the MHL, the CEL with bit flip injections is used (proposed in [5] and in [8] for classification models). The CEL measures the performance of classification NNs returning values that can be interpreted as class probabilities. With the ground truth class i and softmax as input, the CEL is defined as $\ell_{CEL}(\hat{y}, i) = -\log\left(\frac{\exp(\hat{y_i})}{\sum_j \exp(\hat{y_j})}\right)$, which can be written as $\log \sum_j \exp(\hat{y_j}) - \hat{y_i}$. With large differences among $\hat{y_j}$, the term $\sum_j \exp(\hat{y_j})$ becomes approximately the highest value $\hat{y_j}*$ due to the exponential, so the term $\hat{y_j}* - \hat{y_i}$ influences the optimization most. In case of good predictions, this term will be close to zero. In case of bad predictions, i.e. large $\hat{y_j}*$ and small $\hat{y_i}$, the $\hat{y_j}*$ will be decreased and $\hat{y_i}$ increased. In these two cases, the margins between the neuron that returns $\hat{y_i}$ and neurons other than the one returning $\hat{y_j}*$ may not always be considered in the loss, because of the distortion by the exponential. However, the actual margins are considered in the MHL, which we compare to the CEL regarding bit error tolerance next.

B. *MHL without Bit Flips against CEL with Bit Flips*

Fig. 1 presents the experimental results of different BNNs with respect to the accuracy over BER (from 0% to up to 15% in Fig. 1(a) and (b) and from 0% to up to 5% in Fig. 1(c)). For each data set, five BNNs were conducted using MHL without any bit flip injections and CEL with different BERs for bit flip injections. Moreover, for all BNNs trained with MHL, we employed a parameter search for b (in Eq. (6)), testing powers of two, up to two times of the maximum value the neurons in the output layer can compute (maximum output value of a neuron in the output layer is the number of neurons in the layer before the output layer). Among these configurations of b, the best one was chosen.

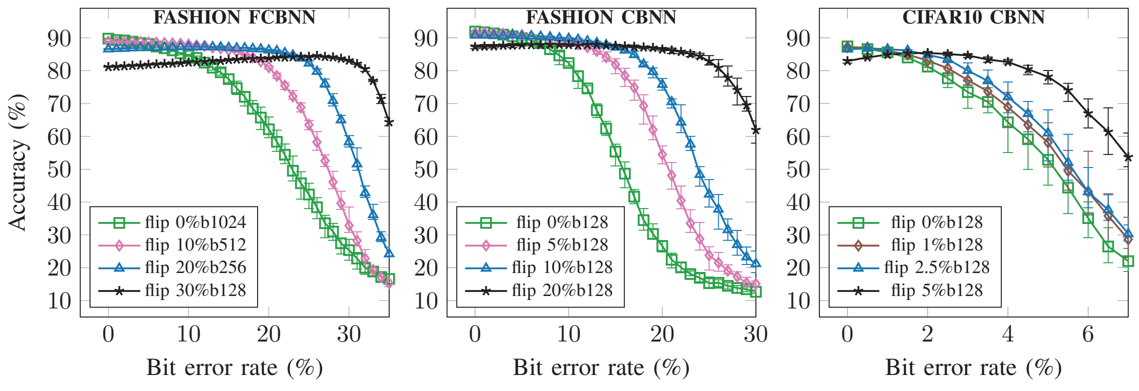
We observe that BNNs trained with the MHL without bit flip injections have better accuracy over BER than the BNNs trained with CEL under bit flip injections, i.e., in Fig. 1(a) and (b) up to 10% and Fig. 1(c) up to 5%. The BNNs trained with CEL suffer from significant accuracy drop for lower BERs, when the BER during training is high, e.g., CEL 20% and/or CEL 30% in Fig. 1 at low BER. The BNNs trained with MHL, however, do not suffer from this accuracy drop. Although the BNNs trained with CEL 20% and bit flip injections have better accuracy for Fashion CBNN in Fig. 1(b) when the error rate is higher than 10%, the accuracy of the BNNs drops by a significant amount, which may be unacceptable. Further investigations should be deployed, to be presented in Section V-C.

C. *Combination of Modified Hinge Loss and Bit Flip Training*

In this section, we evaluate the BNNs trained with the MHL and bit flip injections under different BERs. In addition, the BNNs trained with the MHL without bit flip injections (i.e., those BNNs generated using the MHL in Section V-B under 0% BER) are included here as the baseline in this subsection. For all configurations, we employed the same parameter search for b as in Section V-B.

Fig. 2 presents the experimental results of different BNNs with respect to the accuracy over BER (from 0% to up to 30% in Fig. 2(a) and (b) and from 0% to up to 6% in Fig. 2(c)). In all experiments, we observe that the accuracy over the BER of the BNNs trained under MHL and bit flip injections is significantly higher than that of the baseline trained by only MHL. For example, for Fashion in Fig. 2(a) and 2(b), the BER at which the accuracy degrades significantly is extended from 5% (baseline, green curve) to 20% and 15% respectively, with a small trade-off in the accuracy at 0% BER. If more accuracy at low error bit rates is traded, the BER at which accuracy degrades steeply can be shifted even further. For CIFAR10 in Fig. 2(c), this breaking point can also be increased. However, more accuracy has to be traded compared to the previous cases.

If b is higher than the ones shown, the accuracy for lower BERs suffers similar to using CEL with high BERs. If b is lower, there will be no significant change compared to CEL with 0% BER. We only show the results with the best b.



(a) Fashion FCBNN (b) Fashion CBNN (c) CIFAR10 CBNN

Fig. 2: Accuracy over bit error rate for BNNs trained with MHL and bit flip injections (denoted as flip 0%, 1%, etc). The number after the b is the value to which the parameter b in the MHL is set during training (see Eq. (6)).

VI. Conclusion

In this work, we proposed a concept of margin to formulate a bit error tolerance metric for the entire output layer, for which we formally proved that it measures the maximum number of any bit flips that can be tolerated. Based on this metric and the well-known hinge loss for maximum margin classification in SVMs, we proposed the modified hinge loss (MHL) for optimizing the bit error tolerance of BNNs. Our experimental results show that the BNNs trained with the MHL achieve higher levels of bit error tolerance and accuracy compared to BNNs trained with the cross entropy loss (CEL) and bit flip injections.

We believe that the fundamental understanding of the bit error tolerance in this paper for BNNs provides a cornerstone for the exploration of other NN models. Despite the limitation of using only binary values, the concept of margins of individual neurons and the margins of the output layer can be potentially extended to any NN model with higher precision weights. In these evaluations, alternative ways of margin maximization should be investigated, for retaining properties of the CEL (e.g. notion of class probabilities and applicability to imbalanced classification problems), which may be important in some cases. Avoiding paramter search for b is another important step, e.g. with a different hinge loss. We plan to explore such extensions in the future.

References

[1] S. Buschjäger, J. Chen, K. Chen, M. Günzel, C. Hakert, K. Morik, R. Novkin, L. Pfahler, and M. Yayla. Towards explainable bit error tolerance of resistive ram-based binarized neural networks. *CoRR*, abs/2002.00909, 2020.

[2] S. Cavalieri and O. Mirabella. A novel learning algorithm which improves the partial fault tolerance of multilayer neural networks. *Neural Netw.*, 12(1):91106, Jan. 1999.

[3] P. J. Edwards and A. F. Murray. Penalty terms for fault tolerance. In *Proceedings of International Conference on Neural Networks (ICNN'97)*, volume 2, pages 943–947, 1997.

[4] S. Henwood, F. Leduc-Primeau, and Y. Savaria. Layerwise noise maximisation to train low-energy deep neural networks. *arXiv:1912.10764*, 2019.

[5] T. Hirtzlin, M. Bocquet, J. Klein, E. Nowak, E. Vianello, J. M. Portal, and D. Querlioz. Outstanding bit error tolerance of resistive ram-based binarized neural networks. In *International Conference on Artificial Intelligence Circuits and Systems, AICAS*, pages 288–292, 2019.

[6] T. Hirtzlin, B. Penkovsky, J. Klein, N. Locatelli, A. F. Vincent, M. Bocquet, J. M. Portal, and D. Querlioz. Implementing binarized neural networks with magnetoresistive RAM without error correction. *arXiv:1908.04085*, 2019.

[7] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks. In *Advances in neural information processing systems*, pages 4107–4115, 2016.

[8] S. Koppula, L. Orosa, A. G. Yağlıkçi, R. Azizi, T. Shahroodi, K. Kanellopoulos, and O. Mutlu. Eden: Enabling energy-efficient, high-performance deep neural network inference using approximate dram. In *International Symposium on Microarchitecture*, MICRO 52, 2019.

[9] V. Mrazek, Z. Vasicek, L. Sekanina, M. A. Hanif, and M. Shafique. Alwann: Automatic layer-wise approximation of deep neural network accelerators without retraining. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8, 2019.

[10] B. Reagen, P. Whatmough, R. Adolf, S. Rama, H. Lee, S. K. Lee, J. M. Hernndez-Lobato, G. Wei, and D. Brooks. Minerva: Enabling low-power, highly-accurate deep neural network accelerators. In *International Symposium on Computer Architecture (ISCA)*, 2016.

[11] L. Rosasco, E. De, V. A. Caponnetto, M. Piana, and A. Verri. Are loss functions all the same. *Neural Computation*, 15:2004.

[12] E. Sari, M. Belbahri, and V. P. Nia. How does batch normalization help binary training? *arXiv:1909.09139*, 2019.

[13] D. Simon. Distributed fault tolerance in optimal interpolative nets. *IEEE Transactions on Neural Networks*, 12(6):1348–1357, 2001.

[14] L. Song, Y. Wang, Y. Han, H. Li, Y. Cheng, and X. Li. Stt-ram buffer design for precision-tunable general-purpose neural network accelerator. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25:1285–1296, 2017.

[15] X. Sun, R. Liu, Y. Chen, H. Chiu, W. Chen, M. Chang, and S. Yu. Low-vdd operation of sram synaptic array for implementing ternary neural network. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(10):2962–2965, 2017.

[16] C. Torres-Huitzil and B. Girau. Fault and error tolerance in neural networks: A review. *IEEE Access*, 5:17322–17341, 2017.

[17] L. Yang, D. Bankman, B. Moons, M. Verhelst, and B. Murmann. Bit error tolerance of a cifar-10 binarized convolutional neural network processor. In *International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2018.

[18] L. Yang and B. Murmann. Sram voltage scaling for energy-efficient convolutional neural networks. In *2017 18th International Symposium on Quality Electronic Design (ISQED)*, pages 7–12, 2017.

[19] T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *International Conference on Machine Learning*, ICML '04, page 116, New York, NY, USA, 2004.