Quantifying the impact of data encoding on DNN fault tolerance

Edward Pyne
Harvard University
Cambridge, USA

Lillian Pentecost
Harvard University
Cambridge, USA

Udit Gupta
Harvard University
Cambridge, USA

Gu-Yeon Wei
Harvard University
Cambridge, USA

David Brooks
Harvard University
Cambridge, USA

***Abstract*—DNN fault tolerance is known to vary depending on per-model and per-layer characteristics, and system architects can exploit these variations to improve energy and space efficiency [3], [5], [7], [9], [10]. In this work, we demonstrate that the data encoding format of weight values impacts DNN fault tolerance by up to 10x, and we quantify the substantial variation in fault tolerance across DNNs trained to convergence with identical training hyperparameters. This variation may be reduced with alternative weight encodings. Both results have impacts on making robust design decisions to maintain model accuracy. These studies are enabled by an updated open-source fault injection framework for DNNs designed to quantify the impact of faults on both weight and activation values in both training and inference.**

***Index Terms*—machine learning, deep learning, approximate computing, codesign**

I. Introduction

Deep Neural Networks (DNNs) are deployed on a variety of different hardware platforms, such as datacenters, mobile, and edge clients [1], [2]. In addition to varying performance and energy capabilities, frequency and type of hardware errors are important characteristic of these platforms. In particular, hardware systems and applications are provisioned to run within strict fault tolerance bounds in order to ensure reliable execution and efficiency. However, relaxing these constraints may enable increased performance, energy, and area efficiency, and previous work has demonstrated that DNN applications may maintain accuracy under relaxed bounds [3], [5], [8]. Thus, characterizing and optimizing the fault tolerance capabilities of DNN applications can enable significant improvements in performance, energy, and area efficiency.

DNNs can tolerate orders of magnitude higher bit error rates (BERs) than traditional CPU workloads [3]–[5], and tools have been developed to simulate DNN faults in software [3], [4], [6]. This fault tolerance can be exploited to trade off individual errors during DNN inferences for performance, area, and energy efficiency without sacrificing overall DNN accuracy. For example, recent work has proposed exploiting DNN fault characteristics for lossy algorithmic compression [7]–[10], embedded non-volatile memory storage [11], and hardware acceleration [5], [12], [13]. Unfortunately, the fault tolerance of both weights and activations is model and layer specific [3], [16], and architects must design for worst-case performance, implying tight bounds on model variability can enable better tuned designs.

This work improves DNN reliability by co-designing datatype encodings for model parameters with hardware fault tolerance. We build upon related work [3], [14] to evaluate fixed-point representations for model parameters. We find that changing data encoding from two's complement to sign-magnitude effectively mitigates the impact of sign bit errors, allowing DNNs to maintain baseline accuracy at higher bit error rates. Additionally, we find that instances of the same model topology with identical hyperparameters and training, differing only in random seed, may have significant variation in fault tolerance.

In this paper, we make the following observations:

1) Compared to traditional two's complement representation, sign-magnitude encoding improves fault tolerance by up to 10x, nearly matching the impact of oracle protection of the sign bit.
2) There is significant fault tolerance variation between identical training runs. For example, instances of CNN-based models (i.e., VGG) vary in fault tolerance by up to 5x. If a model is retrained and redeployed, such as in a self-driving car, assuming consistent fault tolerance may be an inaccurate and even dangerous assumption.
3) Compared to traditional two's complement, sign-magnitude encodings suffer from up to 4x less variability in fault tolerance across identical training runs of the same model.

We develop and use an updated version of Ares [3] built with PyTorch [15], which enables native CPU and GPU injections of bitwise weight faults across multiple weight encodings. The updated framework is released as an open-source tool with additional models, support for activation and training injections, and increased customizability for fault model and datatype, as detailed in Table I.

II. Ares Enables Studying DNN Fault Injection

DNN inference fault tolerance is determined by a variety of factors. First, recent work demonstrates that layer type (i.e., FC verus CNN versus RNN) impacts fault tolerance. Second, weights and activations may have varying degrees of fault tolerance [3], [16]. This work demonstrates other design choices, such as bit representation, also impact fault tolerance.

To study the impact of datatype representation on fault tolerance, we build on top of Ares, a high-level fault-injection

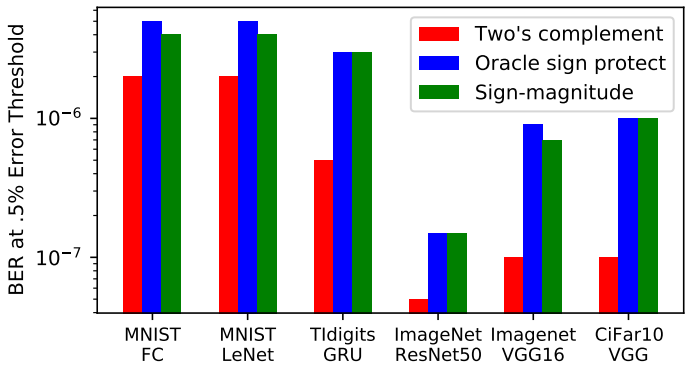

Fig. 1. Sign-magnitude encoding results in models tolerating up to 10x higher bit error rates (BERs) than two's complement, nearly matching the effect of oracle protection of the sign bit.

framework [3]. Fault models are user-configurable, and provided examples include Gaussian noise, uniformly random bit flips, zeroing of values, and more. All types of faults, including bit flips, can be performed on-GPU using tensor operations.

In order to study the implication of faults on DNNs, we model the presence of faults by injecting them into representations of quantized weights and activations. We model weight faults as permanent flips in bits in the weight tensors. In addition to weights, open source support for injections in activations and hidden states (applicable for Recurrent NNs) during model evaluation are added. All transformations may be applied to any subset of model weights or activations, down to particular bits of specified layers. In addition, injections on weights, activations and gradients during training are now supported. An interface to sweep hyperparameter values and fault characteristics over many orders of magnitude is also provided. A user can easily study a new model architecture by extending the base model class, and existing PyTorch models can be added in fewer than 20 lines of code.

III. Data Encoding Impacts Fault Tolerance

How weights are represented in memory changes the impact of certain bit flips, potentially affecting the model's vulnerability. In the standard two's complement representation, flipping the sign bit is equivalent to subtracting the largest representable value for positive weights. DNN weights are clustered near zero, with fewer than 1×10^{-6} fraction of weights in CiFar10-VGG12 larger than 1.5, so sign bit flips are likely to have a large impact on the magnitude of the weight.

TABLE I
New Ares fault modes and expanded customizability

Feature	Original Ares	New Version
Framework	Keras	+ PyTorch
Open-Source Injection Modes	Weights	+ Activations, Gradients
Provided ImageNet models	VGG16, ResNet50	+ MobileNet, InceptionV3
Data Encodings	Fixed per-mode	2c, SM, Float
GPU Accelerated Injection	Activations	+ Weights, Gradients

To determine the extent of this impact, Ares is used to simulate oracle protection to the sign bit of weights, where flips that would occur in the sign bit are silently suppressed. Fault injections are performed on the weights of the six baseline models used in [3] (MNIST FC, MNIST LeNet5, ImageNet VGG16, ImageNet ResNet50, CiFar10 VGG12 and TIDIGITS GRU), with the per-model quantization as detailed.

For each model, an instance is trained, then trials are performed across 50 bit error rates in the range $[10^{-9}, 10^{-3}]$. For each trial, the test set error after fault injection is recorded. At each BER, 20 trials are averaged to estimate the mean model degradation at that fault rate. Taking the model test error with no fault injections as the baseline, the maximum bit error rate with less than a relative .5% increase in error is calculated. Compared to standard two's complement, oracle protection of only the sign bit gives up to an order of magnitude improvement in fault tolerance (Figure 1).

Unfortunately, completely protecting sign bit errors may require architectural tradeoffs, such as storing the bits on a different memory technology or with error correction, which introduces overhead. We achieve the benefits of suppressing sign errors without special protection by changing the bit representation of the weights to sign-magnitude (SM).

In sign-magnitude encoding, flipping the sign bit transforms x to $-x$ while leaving the absolute value unchanged. If M is the largest representable value, for x such that $|x| \leq M/2$, a sign bit error changes x by less in sign-magnitude representation than in two's complement. For values close to 0, the effect on weight magnitude of a sign bit error is much larger in two's complement than sign-magnitude.

For CiFar-VGG12, which has an approximately normal distribution of weights with mean zero and standard deviation less than .02, for over 99.9% of weights a sign bit flip in 2c will have more than $10\times$ greater impact on weight magnitude than the equivalent flip in SM.

We use Ares to analyze the same set of model instances using sign-magnitude representation. For CiFar10-VGG, ImageNet ResNet50 and TiDigits GRU, sign-magnitude provides equal resilience as oracle protection in terms of tolerable bit error rate (Figure 1), achieving *all* of the benefit without special protection of sign bits. For the MNIST FC and LeNet and ImageNet VGG16 DNNs, the maximum sign-magnitude BER is within 20% of that achieved with oracle protection.

Thus, SM representation could allow sign bits to be stored on the same memory as other data, while obtaining the benefit of "protecting" them from error. Despite the design and implementation costs, moving to a sign-magnitude encoding for weights could result in 10x higher acceptable bit error rate while using the same number of bits per weight value.

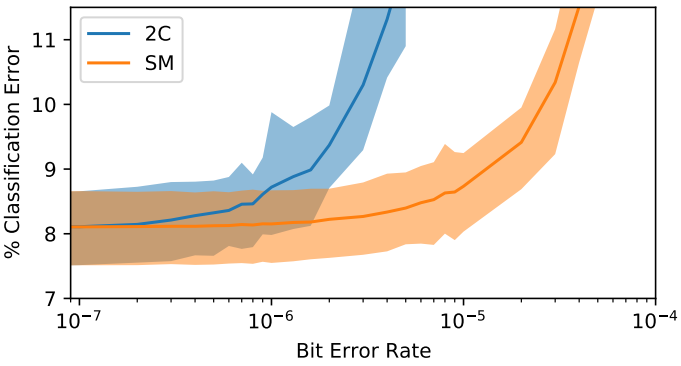

Fig. 2. Mean and min/max (indicated by shaded region) classification error across 20 instances of VGG12 trained on CiFar10 for varying BERs and different data encodings (two's complement, 2c, and sign-magnitude, SM). SM has slower degradation and lower variance, visible as a smoother curve.

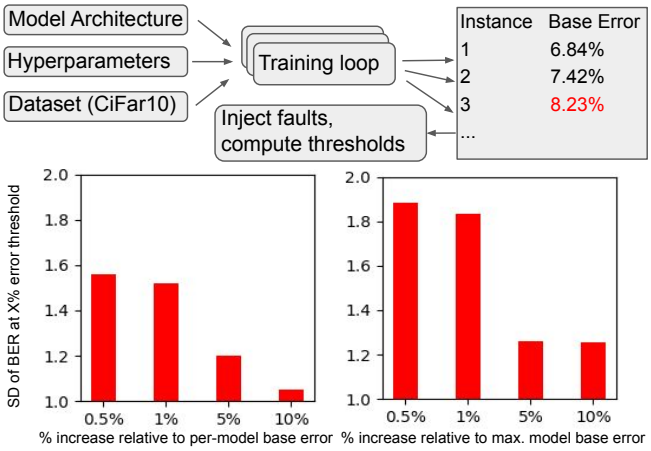

Fig. 3. The process of computing variance in fault tolerance for models with sign-magnitude (SM) weight encoding. With both per-instance and global baselines, BERs required to achieve a certain accuracy exhibit less variance at higher error tolerances (expressed as a percentage of base error).

IV. Fault Tolerance Varies Across Identically Trained Models

Implicit in experiments that examine the fault tolerance of a single instance of a model [3], [13], [16] is the assumption that identically trained instances have equivalent fault tolerance. However, this assumption does not necessarily hold. Even after isolating a specific dataset, model architecture, and training hyperparameters, we find a wide distribution of fault tolerance across training runs. Furthermore, changes to bit representation can affect the *variability* of fault tolerance, in addition to the absolute level.

As with variance of hardware characteristics between devices, variance in the fault tolerance of models on a fixed device may result in a requirement to design for the worst case. Thus, studying the variability in fault tolerance has implications for provisioning hardware if a specific model architecture is continually re-trained and re-deployed [17].

We examine the variation of fault tolerance across training runs of a VGG12 model on the CiFar10 dataset. Each model instance is trained to convergence with identical hyperparameters (e.g., learning rate $= 0.1$, L2 $= 5 * 10^{-4}$). We study the fault rates at different thresholds of accuracy degradation. For each, uniformly random bit flips are injected across all weight values, with 50 trials per BER per model instance.

We perform two sets of comparisons. For the first, we take each model instance's accuracy on the test set with no fault injections as that instance's *baseline error*. Considering sign-magnitude and two's complement encoding, we calculate the highest BER at which each instance stays below .5%, 1%, 5% and 10% relative increase in classification error from its baseline error. We refer to this value as *relative error*, and it is the metric used in [3]. However, comparing instances to their own baseline implicitly sets a higher standard for instances with lower test error. To address this, we fix a common reference point per model by taking the maximum baseline error over all instances, and calculating the corresponding .5%, 1%, 5% and 10% accuracy loss thresholds. For each instance, we then compute the highest BER where the model stays under each threshold and denote this *absolute error*.

A. Results: Relative Variation in Fault Tolerance

Even comparing instances relative to their baseline error, there is nontrivial variation in fault tolerance. For two's complement, identically trained instances vary in 0.5% and 1% thresholds by up to 4x and in 5% by up to 4.5x. For sign-magnitude encoding, BER at the 0.5% threshold varies by up to 2.6x. For a CiFar-VGG instance retrained with identical hyperparameters, assuming equal fault tolerance results in over 5% increase in missclassified images across the test set.

Model instances initially degrade smoothly below a BER of approximately $2 \cdot 10^{-6}$ for SM and $2 \cdot 10^{-7}$ for 2c, then rapidly decay (Figure 2). The 10% threshold corresponds to the knee of the curve at which this rapid decay accelerates, and we observe that the different instances tend to hit this level of accuracy degradation at similar BERs. We quantify this with the geometric standard deviation of the thresholds, as BERs vary multiplicatively, not additively. The geometric standard deviation of relative error falls at higher thresholds, implying versions converge in performance as BER reaches the knee. However, convergence does not occur until possibly unacceptable degradation in accuracy (Figure 3).

B. Results: Absolute Variation in Fault Tolerance

Baseline model classification error varies from 7.5% to 8.6% among CiFar-VGG instances, and we choose the lowest accuracy as an absolute baseline. Examining the accuracy degradation of all model instances compared to the highest observed test error, there is up to 16x variance in 0.5% error threshold. This is because instances with higher baseline accuracy have a larger margin before reaching the threshold.

But despite up to 16x variance in 0.5% threshold, error converges at higher thresholds, with all model instances experiencing 10% degradation thresholds within 1.5x of the same

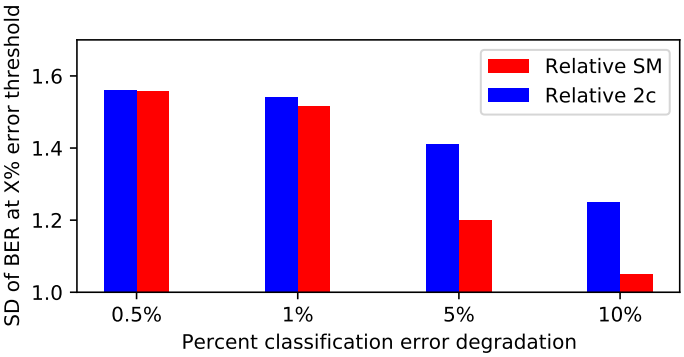

Fig. 4. Geometric standard deviation of relative error thresholds with two's-complement (2c) and sign-magnitude (SM) weight encoding. SM encodings have extremely low variance at higher thresholds.

BER (Figure 3). This strengthens the previous convergence result, as in both the relative and absolute cases the knee of the curve is close across all model instances.

These results suggest that the acceptable BER to minimize impact on classification accuracy differs even among identically trained instances. A given model can be associated with a specific BER above which the model accuracy is unacceptable across all instances, but architects working at lower error tolerances must consider per-instance variation.

C. Results: Data Encoding Affects Variance

Comparing the variance of percent degradation thresholds between sign-magnitude (SM) and two's complement (2c) reveals a substantial difference. The standard deviation of relative 0.5% percent error thresholds is similar between SM and 2c, but falls sharply for SM at higher error degradation thresholds (Figure 4). At the 10% threshold, SM variance is over 6x lower, and the knee location is almost completely invariant among model instances at a BER of 10^{-5}.

For all instances, weight values are concentrated around 0, with a mean of $-.0044$ and a standard deviation of at most $.013$. Fewer than one weight value in ten thousand exceeds the $M/2$ threshold where 2c faults are less damaging. Therefore sign-magnitude sign bit flips are less likely to catastrophically damage a layer, and moving to SM representation effectively mitigates model instance variance due to sign bit vulnerability.

V. Conclusions

We find that moving from standard two's complement to sign-magnitude bit encoding for weights can create up to an order of magnitude increase in fault tolerance. For many models, sign magnitude representation performs as well as oracle protection of the sign bit, potentially enabling storing sign bits with data rather than specially protecting them. Additionally, identically trained instances of a model can have substantial differences in fault tolerance, so fault tolerance may need to be retested after each retraining run, even with all hyperparameters unchanged. However, the bit error rate at which accuracy degradation significantly accelerates is consistent across instances.

The sign-magnitude encoding has lower error variation among instances at all thresholds. For situations where models must adapt on device, or where multiple training runs sweeping fault tolerance are cost-prohibitive, sign-magnitude encoding allows tighter control on the worst case error at a given bit error rate, in addition to lower mean error.

These experiments were enabled by our updates to the Ares[1] framework. Ares enables further characterization of the tradeoffs between faults and DNN performance with a flexible interface for defining fault models and varying design choices. The open source tool has expanded model support, ability to perform all injections on GPU, and support for injecting faults in activations and during training.

References

[1] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." nature 521.7553 (2015): 436-444.
[2] N. P. Jouppi et al., "In-datacenter performance analysis of a tensor processing unit," 2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA), Toronto, ON, 2017, pp. 1-12.
[3] B. Reagen et al., "Ares: A framework for quantifying the resilience of deep neural networks," 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC), San Francisco, CA, 2018, pp. 1-6.
[4] Hari, Siva Kumar Sastry, et al. "SASSIFI: Evaluating resilience of GPU applications." Proceedings of the Workshop on Silicon Errors in Logic-System Effects (SELSE). 2015.
[5] B. Reagen et al., "Minerva: Enabling Low-Power, Highly-Accurate Deep Neural Network Accelerators," 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), Seoul, 2016, pp. 267-278.
[6] "PyTorchFI is a runtime fault injector tool for PyTorch to simulate bit flips within the neural network." https://pypi.org/project/pytorchfi/, 2020.
[7] L. Zonglei and X. Xianhong, "Deep Compression: A Compression Technology for Apron Surveillance Video," in IEEE Access, vol. 7, pp. 129966-129974, 2019.
[8] Reagen, Brandon, et al. "Weightless: Lossy weight encoding for deep neural network compression." arXiv preprint arXiv:1711.04686 (2017).
[9] Y. Zhou, S. Redkar and X. Huang, "Deep learning binary neural network on an FPGA," 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), Boston, MA, 2017, pp. 281-284.
[10] Donato, Marco, et al. "On-chip deep neural network storage with multi-level eNVM." Proceedings of the 55th Annual Design Automation Conference. 2018.
[11] Lillian Pentecost and Marco Donato and Brandon Reagen and Udit Gupta and Siming Ma and Gu-Yeon Wei and David M. Brooks, "MaxNVM: Maximizing DNN Storage Density and Inference Efficiency with Sparse Encoding and Error Mitigation" in Proceedings of IEEE/ACM MICRO '52, pp 769-781, 2019
[12] Gupta, Udit, et al. "MASR: A Modular Accelerator for Sparse RNNs." 2019 28th International Conference on Parallel Architectures and Compilation Techniques (PACT). IEEE, 2019.
[13] M. Lee, K. Hwang and W. Sung, "Fault tolerance analysis of digital feed-forward deep neural networks," 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, 2014, pp. 5031-5035.
[14] Tambe, Thierry, et al. "AdaptivFloat: A Floating-point based Data Type for Resilient Deep Learning Inference." arXiv preprint arXiv:1909.13271 (2019).
[15] "PyTorch: An open source machine learning framework that accelerates the path from research prototyping to production deployment." https://pytorch.org/, 2020
[16] Mahmoud, Abdulrahman, et al. "HarDNN: Feature Map Vulnerability Evaluation in CNNs." arXiv preprint arXiv:2002.09786 (2020).
[17] Hazelwood, Kim, et al. "Applied machine learning at facebook: A datacenter infrastructure perspective." 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA). IEEE, 2018.

[1] https://github.com/alugupta/ares