2019 IEEE/IFIP International Conference on Dependable Systems and Networks Supplemental

Towards a Bayesian Approach for Assessing Fault Tolerance of Deep Neural Networks

Subho S. Banerjee[§], James Cyriac[†], Saurabh Jha[§], Zbigniew T. Kalbarczyk[†] and Ravishankar K. Iyer[†§]
[§]Department of Computer Science, [†]Department of Electrical and Computer Engineering,
University of Illinois at Urbana-Champaign, Urbana IL 61801, USA.

***Abstract*—This paper presents *Bayesian Deep Learning based Fault Injection* (BDLFI), a novel methodology for fault injection in neural networks (NNs) and more generally differentiable programs. BDLFI uses (1) Bayesian Deep Learning to model the propagation of faults, and (2) Markov Chain Monte Carlo inference to quantify the effect of faults on the outputs of a NN. We demonstrate BDLFI on two representative networks and present our results that challenge pre-existing results in the field.**

I. Introduction

For the past decade, machine learning (ML) and particularly deep neural network (DNN) models have had tremendous success in solving problems in image classification, speech recognition, text translation, and game playing. This ubiquitous success has led to the application of a large number of these models in safety critical systems, e.g., autonomous vehicles, and industrial control systems, where reliability and safety guarantees are paramount. Traditional approaches for validation of these guarantees have relied on random *Fault Injection* (FI) [1] to test the resilience of a software/hardware platform in the presence of faults. However, these traditional approaches are fundamentally challenged due to (1) the enormous space of fault locations and program states that must be injected/investigated; (2) the need for significant system support (i.e., debug capabilities like `ptrace` support) to build system-specific injectors; and (3) the inability to provide statistical guarantees about "completeness" beyond performed injections. This presents a significant challenge for today's popular DNN models that scale to millions of parameters and employ special purpose (often proprietary) hardware accelerators that do not support a wide range of debugging capabilities (e.g., GPUs, TPUs).

In this paper, we address the above challenges by presenting *Bayesian Deep Learning based Fault Injection* (BDLFI), a methodology for fault injection in neural networks (NNs). BDLFI uses (1) Bayesian Deep Learning (BDL) to model the propagation of faults, and (2) Markov Chain Monte Carlo (MCMC) inference to quantify the effect of faults on the outputs of a NN [2]. BDL and MCMC provide a formalism and inference method to quantify uncertainty in the inputs, parameters (referred to as weights), activations, and outputs of DNN models. BDLFI leverages this aspect of BDL and models uncertainty as being caused by transient hardware faults.

Advantages of BDLFI over traditional fault injection techniques include: (1) the ability to quantify "completeness" of an injection campaign (i.e., when further injections do not change measured hypothesis) using MCMC-mixing; and (2) the ability to use algorithmic acceleration techniques; (3) the ability to use specialized hardware (without needing system support to do injections) to directly accelerate inference and hence the fault injection campaigns. As a result, we believe that BDLFI can subsume current source-level (e.g., [3]) and debugger-level (e.g., [4]) FIs as a method for rapidly validating the resilience of ML/DNN models executing on embedded accelerator platforms. BFI can be used to inject faults into programs other than neural networks, with the only assumption being that of end-to-end differentiability (i.e., one can calculate the gradient of the output of a program over its input).

II. Bayesian Fault Injector

We now describe the operation of the BDLFI on a multi-layer perceptron (MLP) network shown as ❶ in Fig. 1.

Fault Model. We consider transient faults in the memory

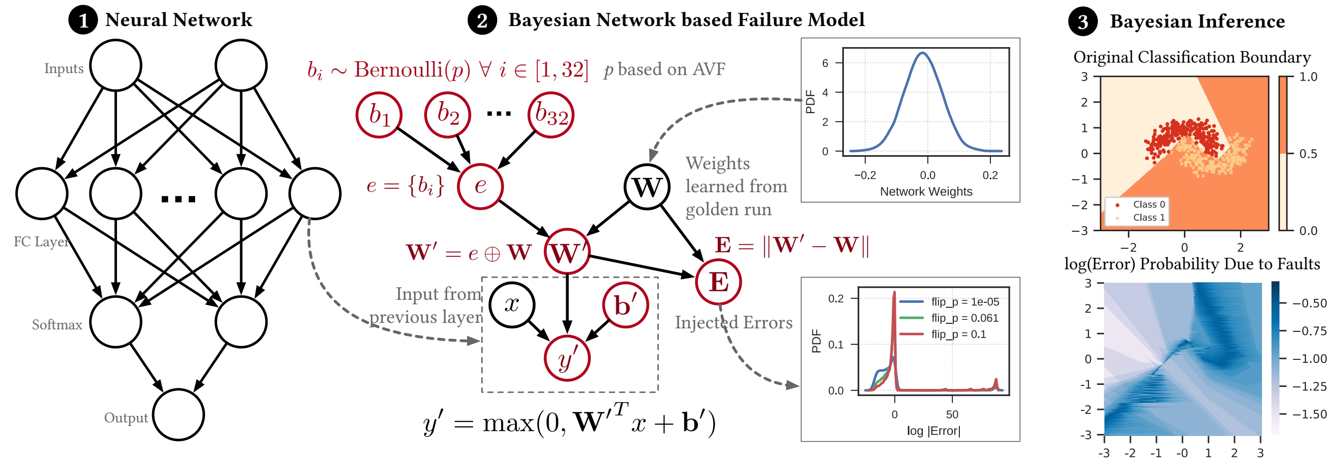
❶ Neural Network
Inputs
FC Layer
Softmax
Output
❷ Bayesian Network based Failure Model
$b_i \sim \text{Bernoulli}(p)\ \forall\ i \in [1, 32]$
p based on AVF
b_1
b_2
b_{32}
$e = \{b_i\}$
e
W
Weights learned from golden run
$\mathbf{W}' = e \oplus \mathbf{W}$
$\mathbf{W}'$
$\mathbf{E} = \|\mathbf{W}' - \mathbf{W}\|$
E
Input from previous layer
x
$\mathbf{b}'$
y'
Injected Errors
$y' = \max(0, \mathbf{W}'^T x + \mathbf{b}')$
PDF
Network Weights
flip_p = 1e-05
flip_p = 0.061
flip_p = 0.1
log |Error|
❸ Bayesian Inference
Original Classification Boundary
Class 0
Class 1
log(Error) Probability Due to Faults

Figure 1. The BDLFI Approach: Faults are modeled (as Bayesian Networks) at every neuron of the network and propagated as probability distributions through the network, and inferred at the output using Markov Chain Monte Carlo.

978-1-7281-3028-6/19/$31.00 ©2019 IEEE
DOI 10.1109/DSN-S.2019.00018

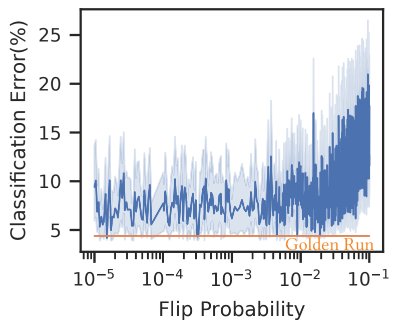

Figure 2. Error injections in all layers of multi-layer perceptron from Fig. 1.

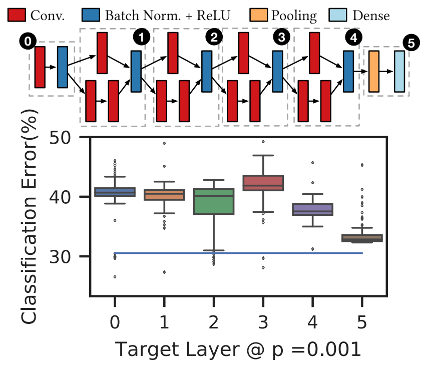

Figure 3. ResNet-18 network and results of injections on a layer-by-layer basis.

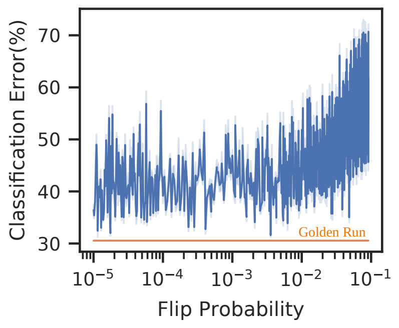

Figure 4. Error injections in all layers of ResNet-18 from Fig. 3.

units for storing NN parameters, inputs, intermediate activations and outputs. We model such faults by using the per-bit architectural vulnerability factor (AVF), i.e., each bit error is treated as a Bernoulli random variable with probability p. We do not make any assumptions about the number of bits in error; this is determined by p. All network parameters, inputs, and outputs are encoded as 32-bit floating point numbers. BDLFI can also be extended to other fault models.

Bayesian Network of Fault Model. We encode the above fault model in a Bayesian Network shown as ❷ in Fig. 1 for each neuron in the NN. The weights and biases ($\mathbf{W}$ and $\mathbf{b}$) are transformed into their error injected version ($\mathbf{W}'$ and $\mathbf{b}'$) by performing bitwise-XOR operations with flipped bits described above. As a result, output of each neuron is a probability distribution over its output space given the original weights and p. *The key insight is that the fault behavior is modeled in the mathematical formulation of the Bayesian network. The fault behavior is then propagated through the NN (using BDL) with every neuron activation capturing the fault behavior of its input, as well as its own fault behavior. We can hence capture end-to-end fault propagation in the NN as a statistical model that can be automatically inferred.*

Inference. We quantify the effect of the injected faults by comparing the classification accuracy of a golden run of the network with the distribution of classification accuracy produced by the BDLFI. This distribution is produced by performing Markov Chain Monte Carlo based inference on the output of the fault injected network. The fault injection is performed using the following steps:

1) The network is trained to obtain the weights of the golden network that provide good classification accuracy.
2) The bit flip fault model and the trained weights of the network are used to create the error distribution over the network weights.
3) A Bayesian fault model is created for each neuron in the network using the error distribution to construct a DBN.
4) Perform inference multiple times on the DBN using MCMC to obtain the classification uncertainty of the network for different flip probabilities.

❸ in Fig. 1 illustrates a golden run as well as the distribution of classification error produced by BDLFI.

III. Preliminary Results

We performed fault injection campaigns over an MLP in Fig. 1 and a ResNet-18 network trained on the CIFAR-10 dataset. Our preliminary results in characterizing DNN resiliency, show several interesting behaviours that challenge existing notions generated from traditional random fault injection methods.

Effect of faults is most significant at the decision boundary. ❸ in Fig. 1 shows the effect of memory faults in an MLP network. The most likely classification errors are produced as a result of faults that happen at the decision boundary. This motivates the need for having reliability features in safety-critical systems, where the points that are close to the decision boundary (i.e., harder to classify) are more egregiously affected by errors. By analyzing the probability of errors near the boundaries, we can set a threshold on the regions of the feature space that need more protection and verification of correctness.

Scope for trading off reliability and performance in DNN architectures. The relationship of NN classification accuracy with p can be seen in Fig. 2 and Fig. 4. In both cases, we can see that there are two clear regimes explaining the effect of bit flip probability on network classification error. In the first regime consisting of smaller flip probability values, we observe that there is no significant increase in average classification error as the bit flip probability increases. In the second regime, we see that average classification error increases significantly with flip probability. Hence operating at the knee of these curves provides the optimal performance-reliability trade-offs in building DNN systems.

Error propagation to the output of the neural network is not related to the depth of the injection layer. Contrary to previous work [1] Fig. 3 shows that there is no direct relationship between the layer in which the fault manifests and the network classification error. We believe that this artifact appears in traditional FI because of incomplete traversal of the entire injection space. We are currently investigating this behavior on other NNs.

References

[1] G. Li *et al.*, "Understanding error propagation in deep learning neural network (dnn) accelerators and applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis.* ACM, 2017, p. 8.
[2] Y. Gal, "Uncertainty in deep learning," *University of Cambridge*, 2016.
[3] B. Reagen *et al.*, "Ares: A framework for quantifying the resilience of deep neural networks," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC).* IEEE, 2018, pp. 1–6.
[4] G. Li, K. Pattabiraman, and N. DeBardeleben, "Tensorfi: A configurable fault injector for tensorflow applications," in *2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW).* IEEE, 2018, pp. 313–320.