TensorFI: A Flexible Fault Injection Framework for TensorFlow Applications

Zitao Chen & **Niranjhana Narayanan**, Bo Fang, Guanpeng (Justin) Li, Karthik Pattabiraman, Nathan DeBardeleben

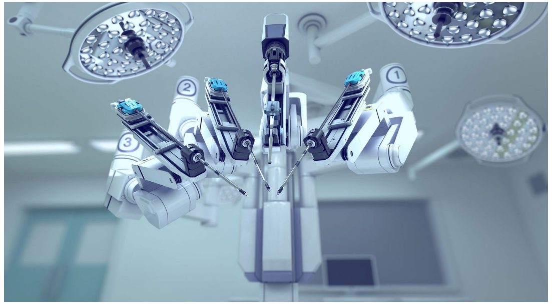
THE UNIVERSITY OF BRITISH COLUMBIA

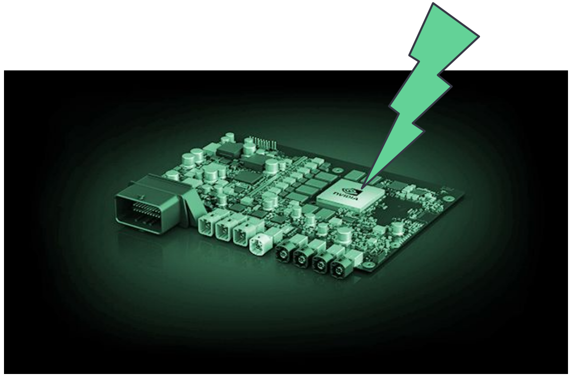
THE UNIVERSITY OF IOWA

Los Alamos NATIONAL LABORATORY EST. 1943

Motivation

- ML is increasingly deployed in safety critical systems
- ML reliability becomes important
  - Soft errors or transient hardware faults: Occur every 53m in 1m nodes*
  - ISO26262 safety standard requires low FIT (10 for ASIL-D level 4) for road vehicles

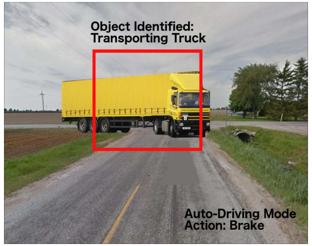


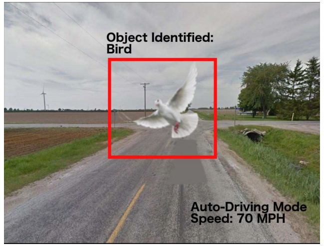


* J. Dongarra, T. Herault, and Y. Robert, “Fault tolerance techniques for high-performance computing,” in Fault Tolerance Techniques for High Performance Computing, 2015, pp. 3–85

Error Consequences: Autonomous Vehicles

- Single-bit fault ➔ Misclassification of image
    - Guanpeng Li et al., “Understanding Error Propagation in Deep Learning Neural Network (DNN) Accelerators and Applications”, SC’17





    - Zitao Chen et al., “BinFI: An Efficient Fault Injector for Safety-Critical ML Systems”, SC’19

Current Solutions: Fault Injection

- Generic Fault Injection (FI)
  - Non-optimal for DNN applications, costly performance
  - Low level, not readily accessible to users
- ML specific FI
  - ML models coupled with hardware faults or the platforms used
  - Limited number of ML models supported

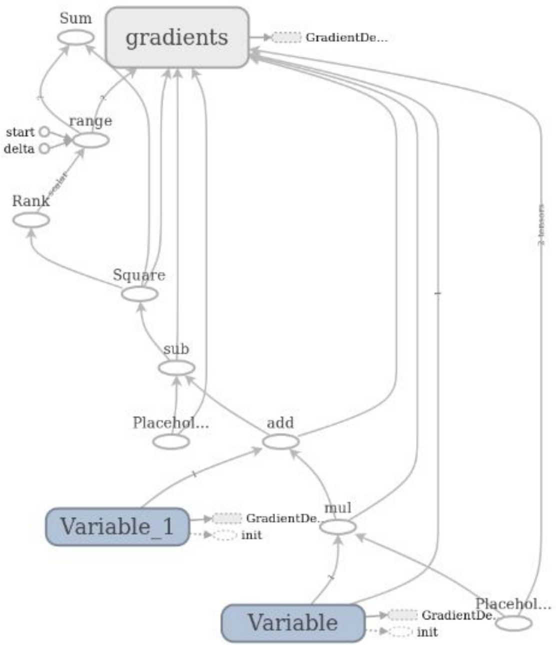
Our Goal

- Inject faults into any generic ML program
- Emulate both hardware and software faults
- High level, easy to use and understand
- Highly portable across platforms

TensorFlow is an open source, popular framework

TensorFlow

TensorFlow v1 Workflow



- ML algorithms are expressed as dataflow graphs

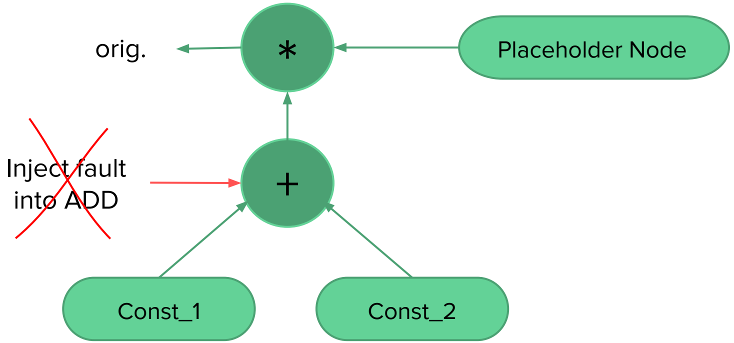
```
#!usr/bin/python

import tensorflow as tf

W = b = tf.Variable([0.3], dtype=tf.float32)

X = Y = tf.placeholder(tf.float32)

linear_model = W*X + b

...
train = tf.train.GradientDescentOptimizer(0.01).minimize(error)

...
with tf.Session() as sess:
        sess.run(tf.global_variables_initializer())

        for i in range(1000):
                sess.run(train, {X: x, Y: y})
```

Fault Model

- Faults injected only during the inference phase
- Faults injected at TF operator output level (interface-level injection)
- Emulates faults occurring at processor’s datapath
    - Arithmetic operators (add, matmul, concat, conv2d, ..)
    - Operators dealing with shape of tensor excluded
- Faults can be
    - Single bit-flips (now multi-bitflips too)
    - Random value changes
    - Zeroed out tensor values

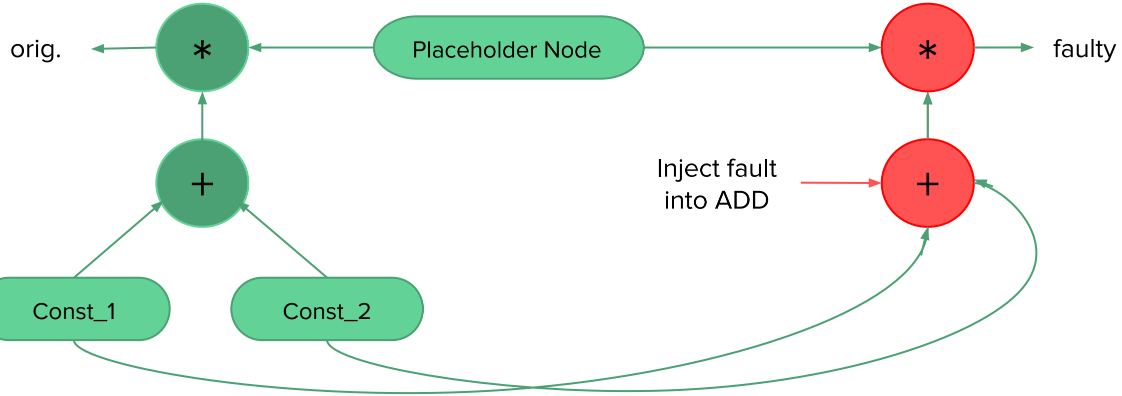
Fault Injection Challenges

- Problem: Tapping into particular node during runtime is not possible



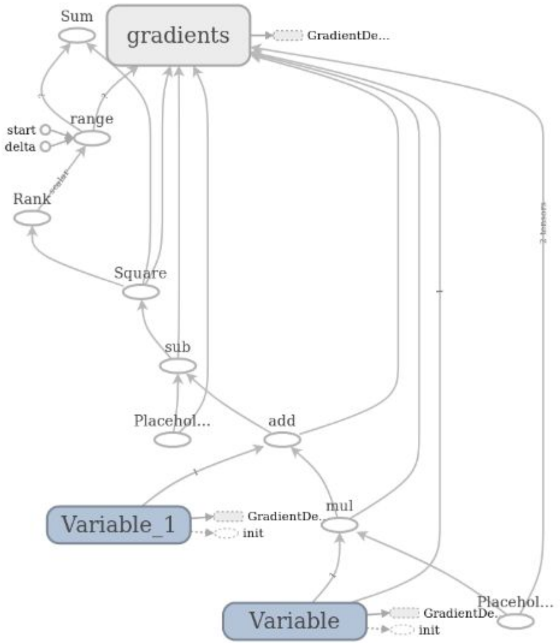
TensorFI Approach

- Solution: Duplicate graph and choose the node at runtime!

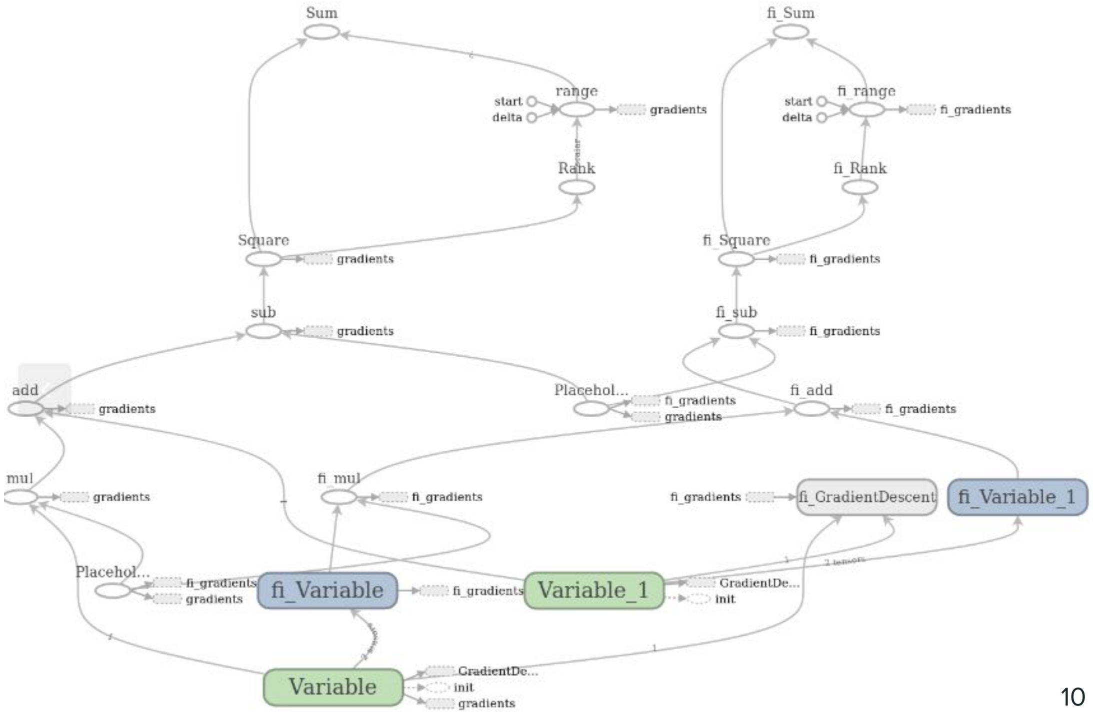


TensorBoard Visualization: Linear Regression

Before calling TensorFI: Original graph



After calling TensorFI: Duplicate FI nodes inserted



TensorFI Usage Example

```
#!usr/bin/python

import tensorflow as tf
import TensorFI as ti

W = b = tf.Variable([0.3], dtype=tf.float32)

X = Y = tf.placeholder(tf.float32)

linear_model = W*X + b

...
train = tf.train.GradientDescentOptimizer(0.01).minimize(error)

...
with tf.Session() as sess:
        sess.run(tf.global_variables_initializer())

        for i in range(1000):
                sess.run(train, {X: x, Y: y})

        fi = ti.TensorFI(sess, name="linear_reg", logLevel=50,
                                disableInjections=True)

        W_, b_, e_ = sess.run([W, b, error], {X: x, Y: y})

        fi.turnOnInjections()

        W_i, b_i, e_i = sess.run([W, b, error], {X: x, Y: y})
```

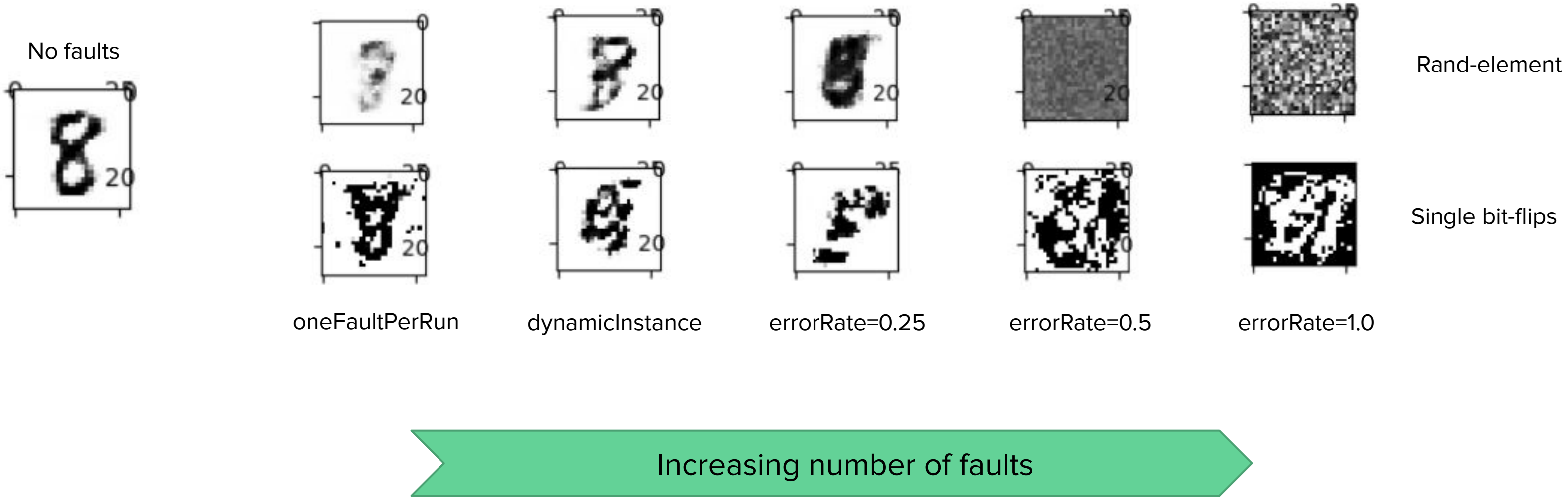
Import module

Instrument code

Begin FI

Example Output for a GAN model

- Generated images of the digit 8 in the MNIST data-set under different fault configurations for Generative Adversarial Networks (GANs)



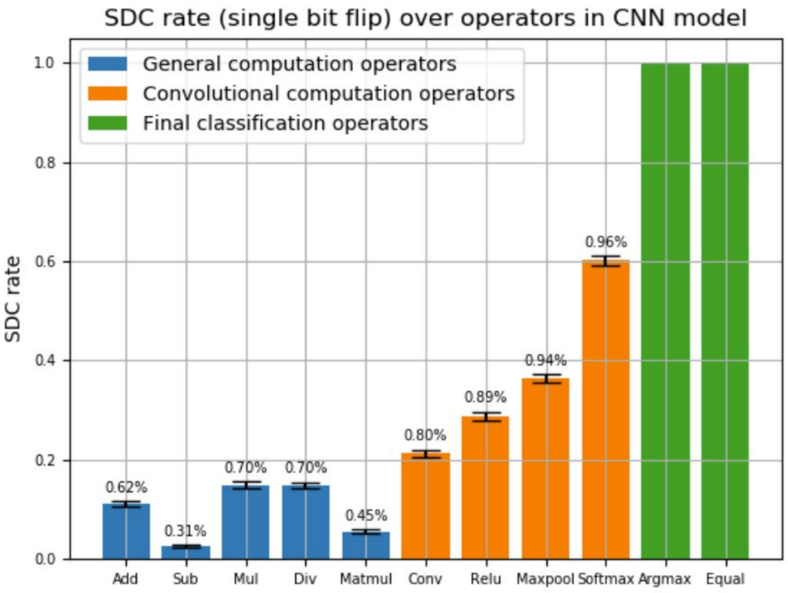
Benchmarks & Experimental Setup

- 12 ML models used
  - Basic, DNNs: NN, CNN, FCN, LeNet, AlexNet, Highway CNN, RNN, VGG-11, ResNet-18, SqueezeNet
  - Driving: Commai.ai driving model
  - Unsupervised: GAN
- 4 open source datasets
  - MNIST, GTSRB (traffic sign), ImageNet, driving frame dataset
- 10000*(15*11 + 3*6*2 + 13) =~ 2 million fault injections
- Silent Data Corruption (SDC) chosen as the standard metric
  - Output mismatch from the fault-free execution

Research Questions

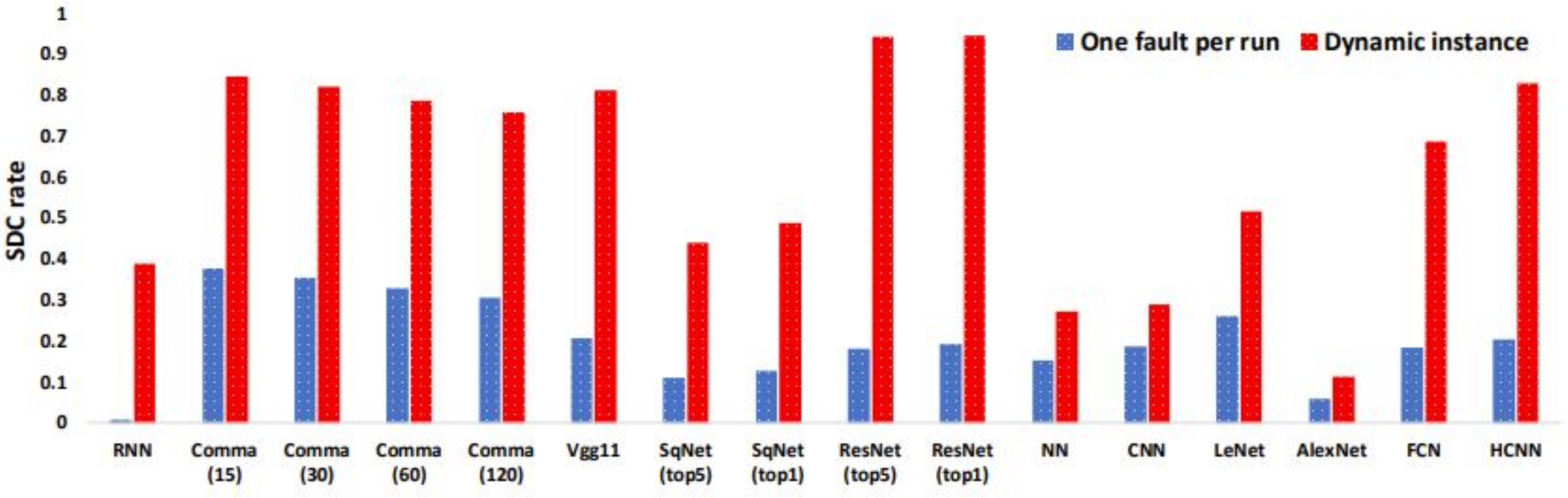
- Fault tolerance of different operators in a single ML model
- Fault tolerance of different ML models under different error modes
- Fault tolerance of different ML models under different error rates
- Instrumentation and injection overheads of TensorFI

Results: Fault Tolerance of Different Operators



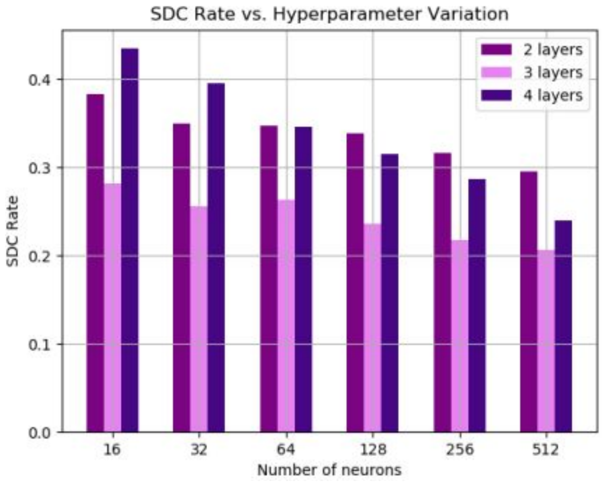
- Faults in the convolution layers are more likely to propagate and amplify
- Faults in output layers (softmax, argmax, equal) almost always result in SDC

Results: Different Models under Different Error Modes



- RNN exhibits the highest resilience in single fault mode, but loses to more faults
- AlexNet has overall high resilience as it has more operators that mask faults (RQ1)
- Comma.ai models have higher SDC rates than most classifier models

Case Study: Effect of Hyperparameter Variations (NN)



- SDC rates decrease with increase in number of neurons
- Layer redundancy has an optimal point (here, 3 layers)

Key: Choose redundancy carefully

Summary: TensorFI

- Built a flexible fault injector for injecting h/w and s/w faults in the TF graph
    - High level representation of faults
    - Portable, configurable, compatible with third-party libraries that use TF
- Used TensorFI to evaluate and study the resilience of 12 ML applications under different fault configurations, including ones used in AVs
- Demonstrated the utility of the tool to improve resilience of selected applications via hyperparameter optimization and selective layer protection
    - Read our paper here: http://blogs.ubc.ca/karthik/files/2020/08/issre20-tensorfi.pdf
    - Try out TensorFI: https://github.com/DependableSystemsLab/TensorFI
    - For information, doubts and clarifications, contact: nniranjhana@ece.ubc.ca