BinFI: An Efficient Fault Injector for Safety-Critical Machine Learning Systems

Zitao Chen, Guanpeng Li, Karthik Pattabiraman, Nathan DeBardeleben

Motivation

- Machine learning taking computing by storm
  - HPC-ML: precision medicine; earthquake simulation;
  - Growing safety-critical applications.
- Reliability of ML becomes important.

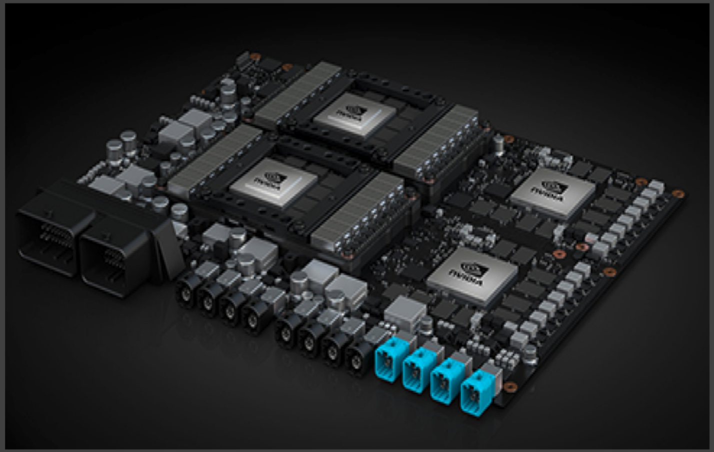
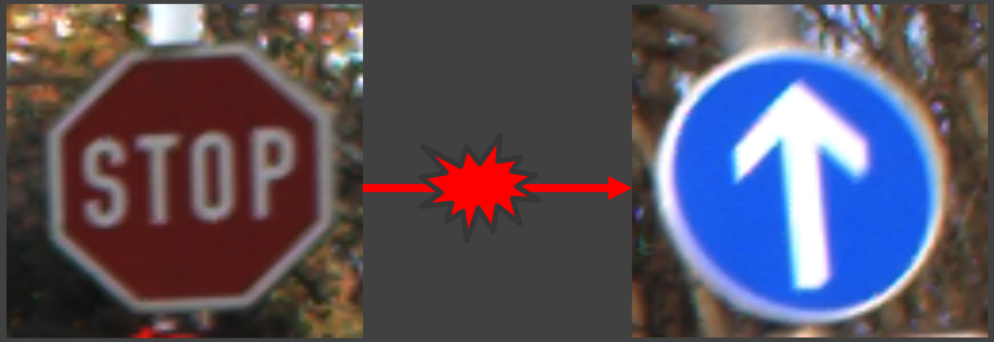


Image source: https://www.nvidia.com/en-us/self-driving-cars/drive-platform/hardware/

Soft Error

- Transient hardware fault
  - Growing in frequency:
    - Occur every 53 mins in 1m nodes [1]
  - Manifested as a single bit-flip
- Silent data corruptions (SDCs)
  - Erroneous ML output.
- Safety standard for road vehicles:
  - ISO26262: 10 FIT

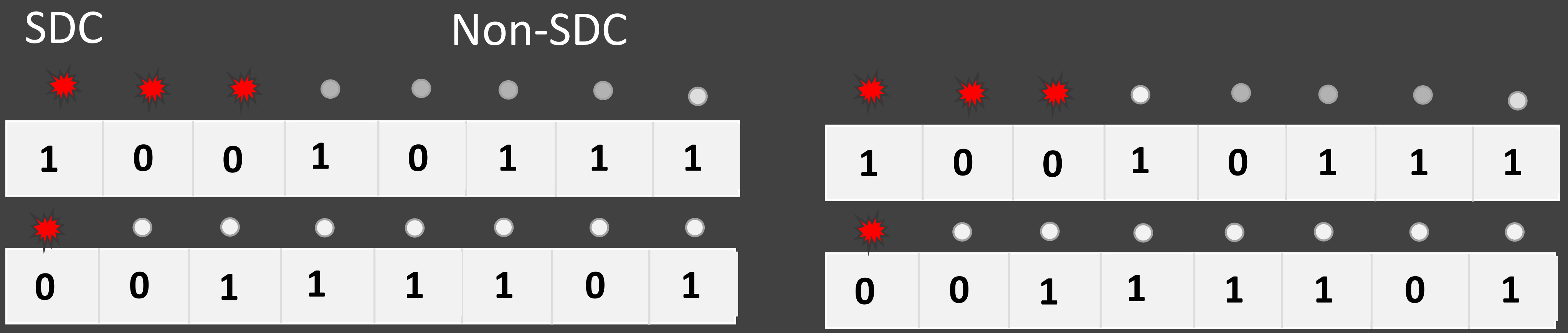


[1] J. Dongarra, T. Herault, and Y. Robert, “Fault tolerance techniques for high-performance computing,” in Fault-Tolerance Techniques for HighPerformance Computing, 2015, pp. 3–85.

Existing solutions

- Application-agnostic:
  - Triple modular duplication (TMR) for execution units.
  - Expensive: Hardware cost, performance (e.g., delay-based accidents).
- Application-specific:
  - *Random fault injection* to guide protection (e.g., instruction duplication).
  - Coarse-grained

Is Random FI Good Enough?

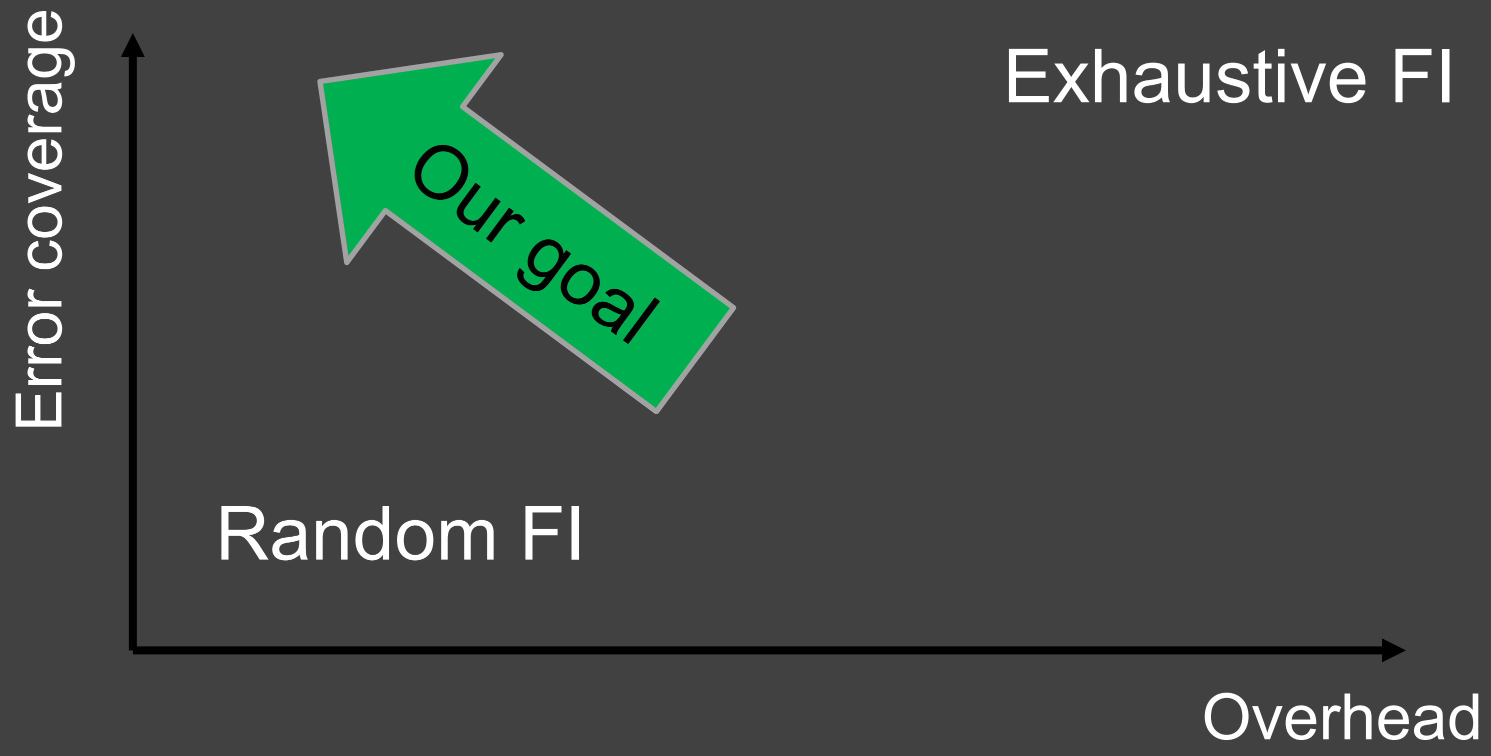


Randomly simulate bit-flip, and then obtain statistical error resilience

25% SDC

1. Where are the *critical faults* in the entire system?
2. Are the critical faults uniformly distributed (or not)?

Our goal



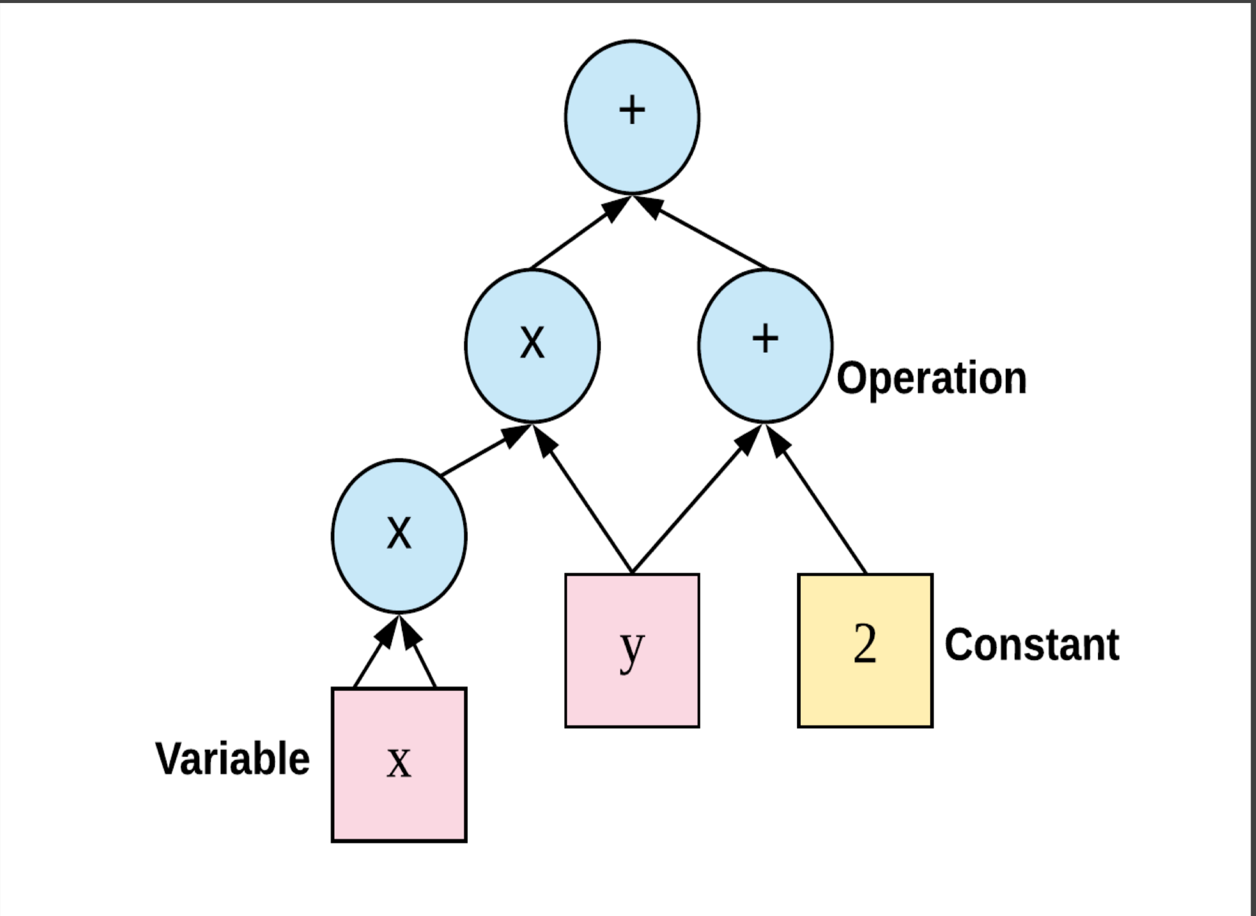
An efficient approach to obtain fine-grained understanding of the error resilience of ML systems

Contributions:

- Identify the property of ML computations which constrain the fault propagation behaviors.
- Characterize the pattern of critical faults.
- Propose a Binary-FI approach to identify critical bits.

ML framework - TensorFlow

- TensorFlow: **framework for executing dataflow graphs**
- ML algorithms expressed as dataflow graphs

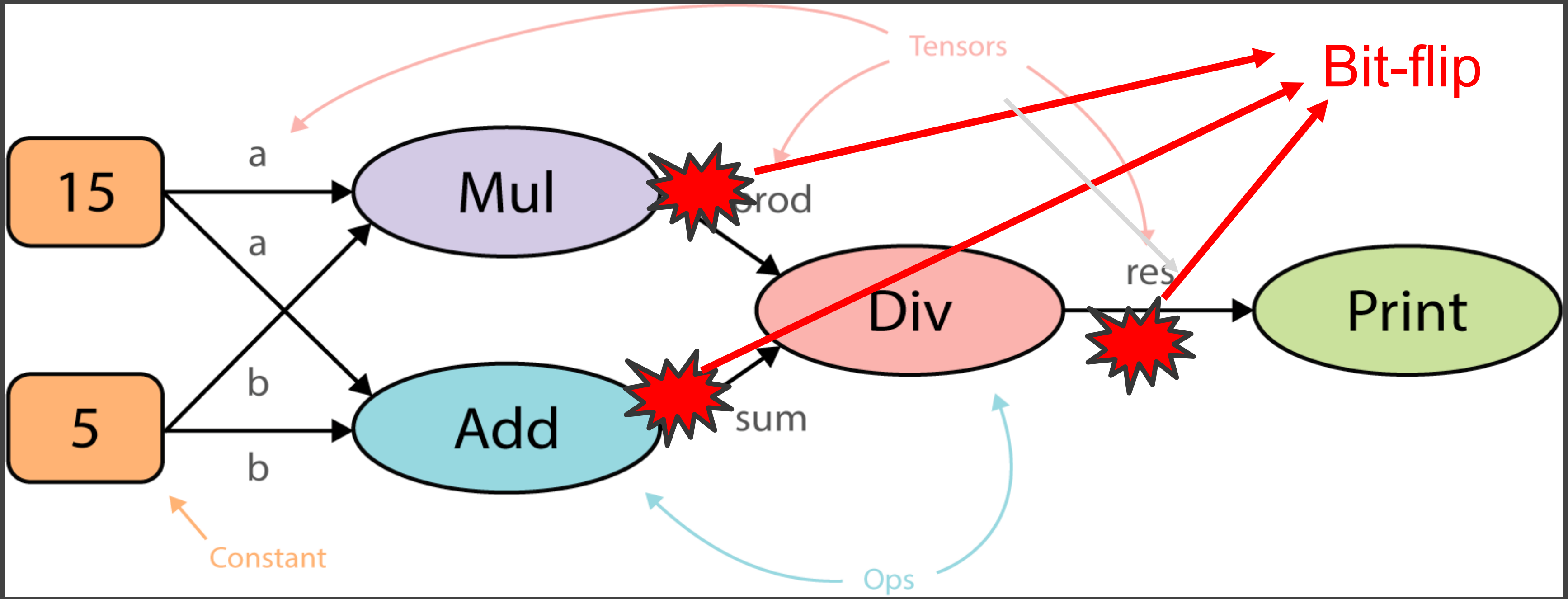


- Others: PyTorch Caffe

Image source: https://www.easy-tensorflow.com/tf-tutorials/basics/graph-and-session

Fault model

- Focus on inference phase
- Faults at processor's datapath (e.g. ALUs)
- Interface-level fault injection (i.e. TensorFlow Operators) [2]

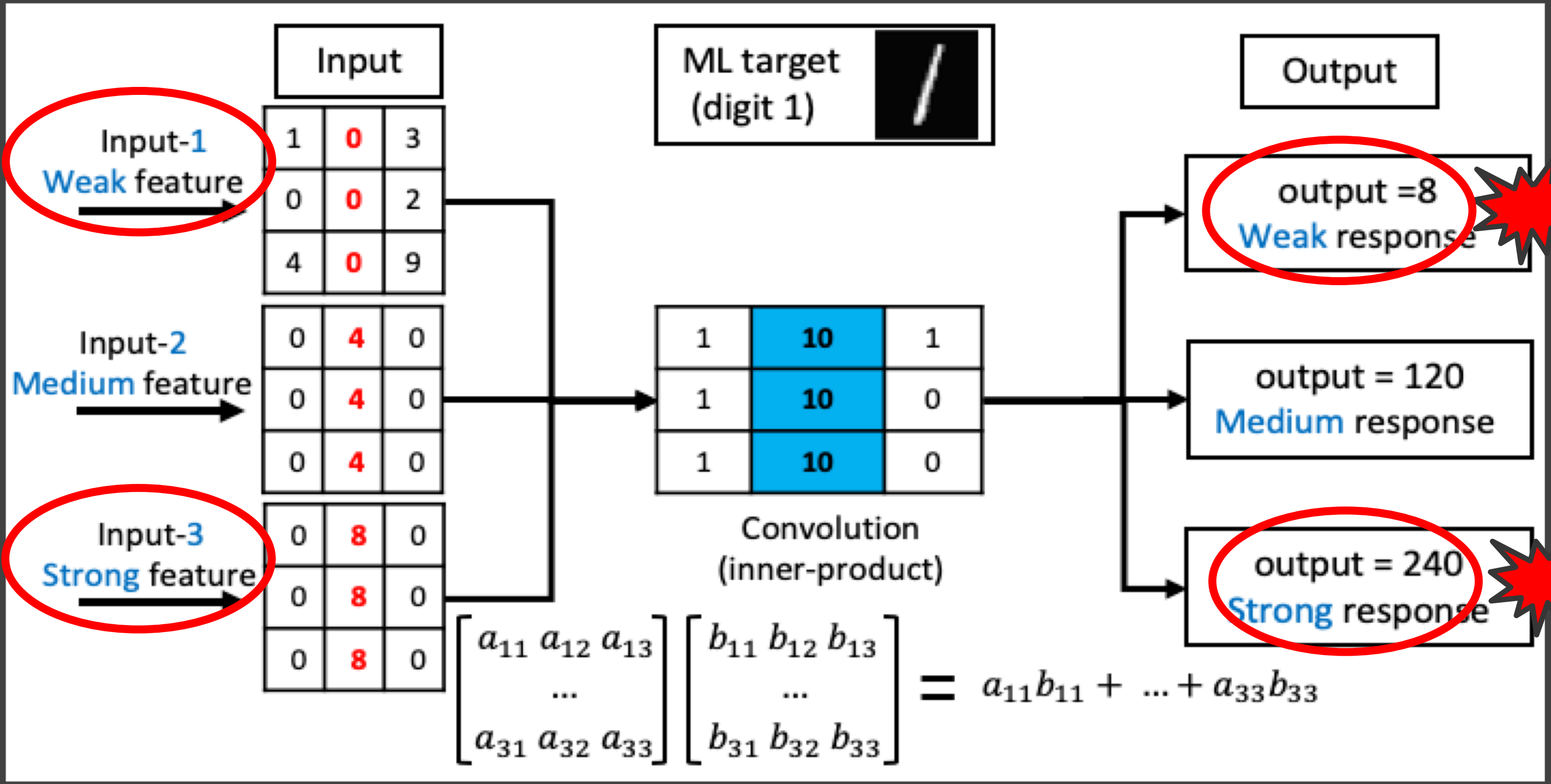


[2] Li et al. TensorFI: A Configurable Fault Injector for TensorFlow Applications. ISSREW'18).

Image source: https://medium.com/@d3lm/understand-tensorflow-by-mimicking-its-api-from-scratch-faa55787170d

How to cause an SDC in ML

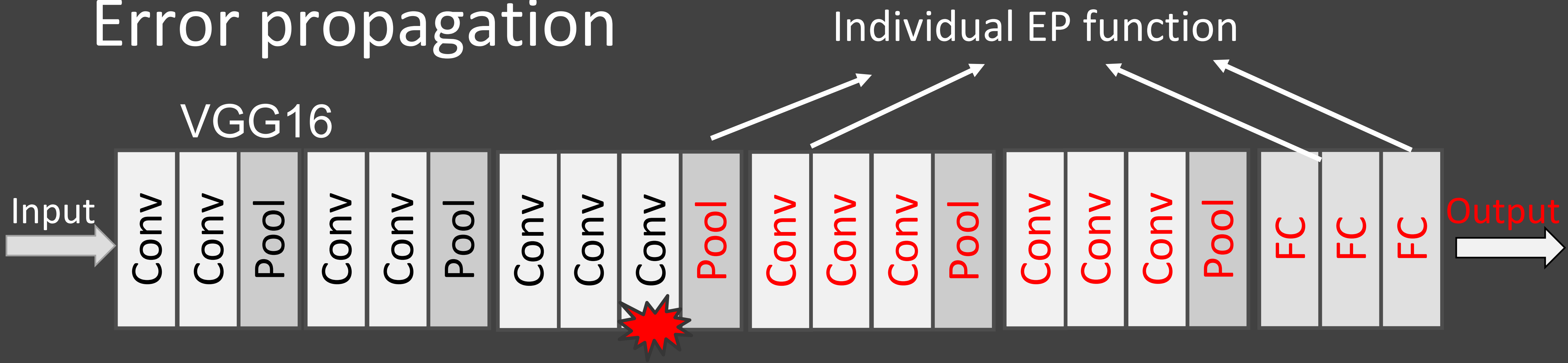
- In ML, fault usually results in numerical change in the data.
- Output by ML is usually determined by numerical magnitude.
  - ***To cause SDC:*** large deviation at the output



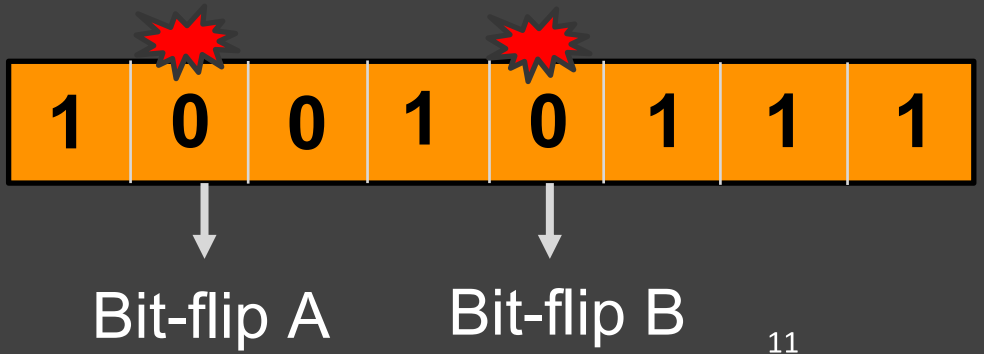
Large positive deviation

Large negative deviation

Error propagation



- Error propagation (EP): from the fault occurrence to the output.
  - **Each EP function**: large response to Large input, i.e., monotone
  - Convolution function: $X * W = \sum x_i w_i$
    - Larger Input deviation: A > B
    - Larger Output deviation: $|Aw_i| \geq |Bw_i|$



Individual EP function

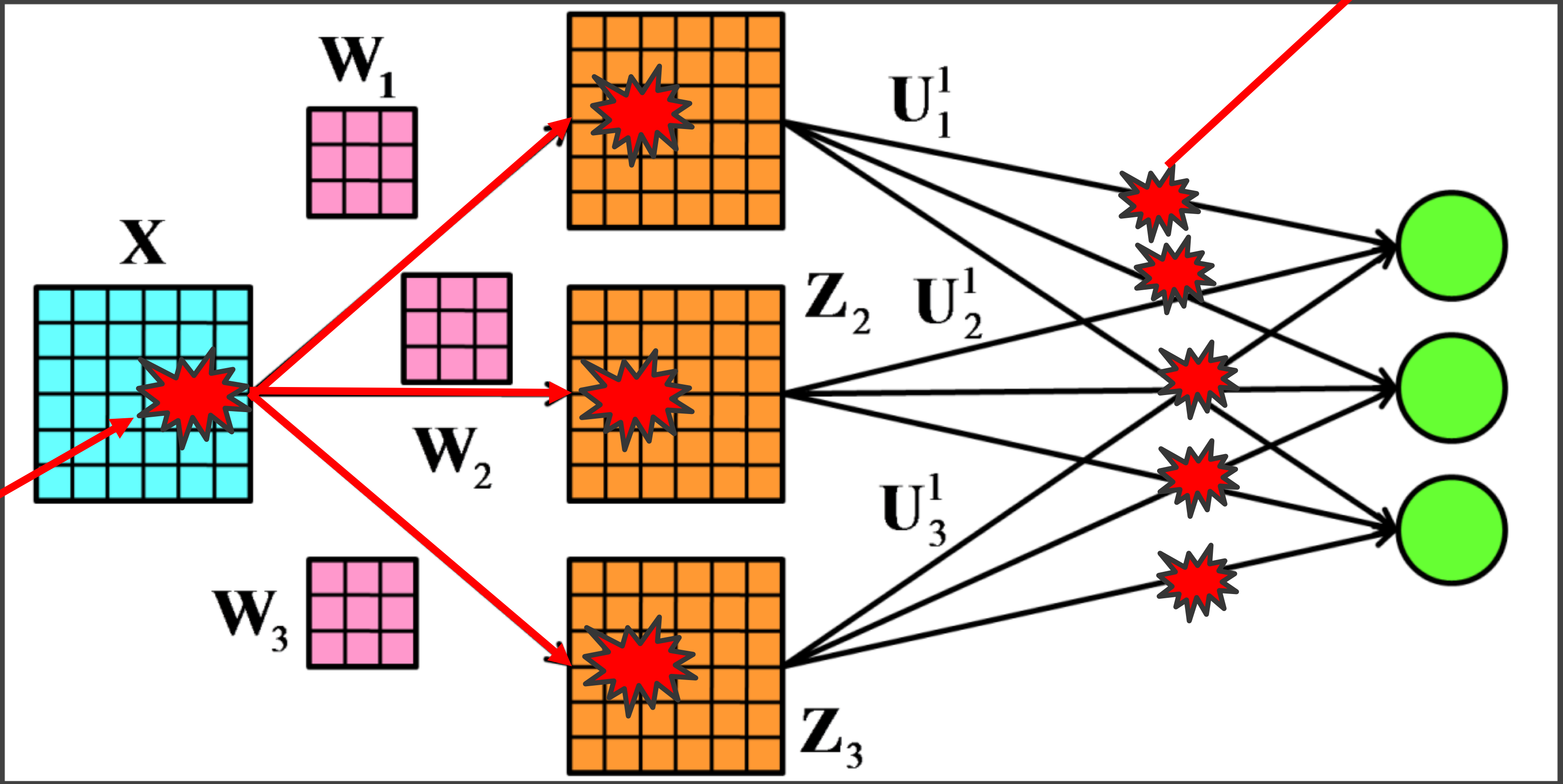
- Common computations in DNNs, satisfy monotone property
- Why: ML tends generate large responses to ``target'' class/feature

Basic	Conv; MatMul; Add (BiasAdd)
Activation	ReLu; ELu;
Pooling	Max-pool; Average-pool
Normalization	Batch normalization (BN); Local Response Normalization (LRN)
Data transformation	Reshape; Concatenate; Dropout
Others	SoftMax; Residual function

Error propagation example

Multiple faults

- One fault propagates into multiple faults.

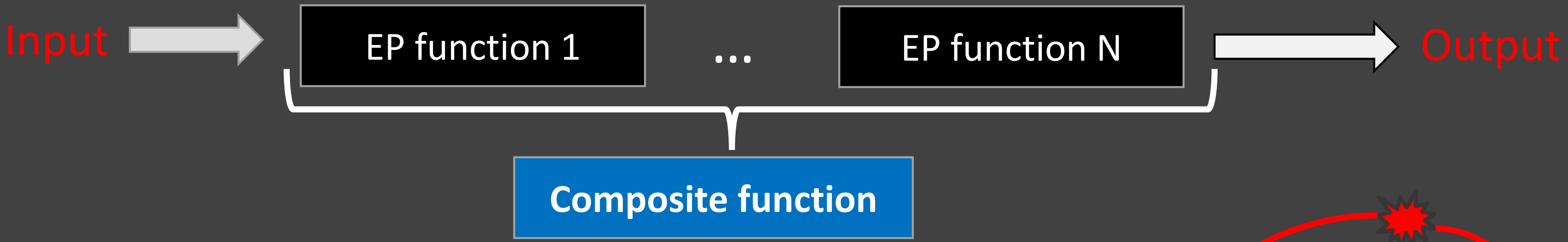


Single fault

Image source: http://worldcomp-proceedings.com/proc/p2014/ABD3492.pdf

All EP functions

- Composite EP function:

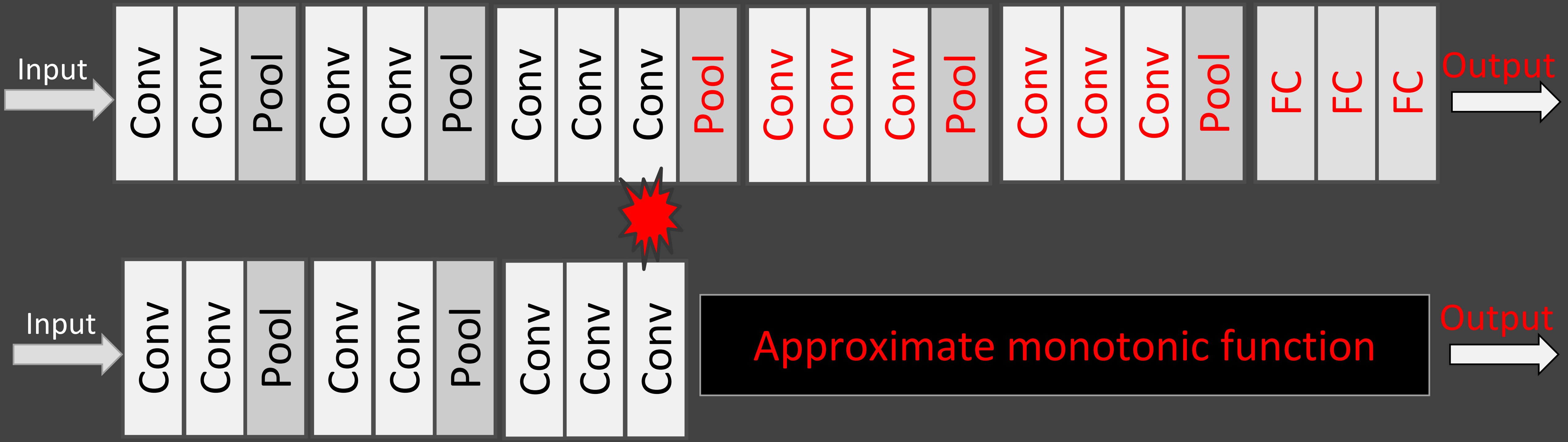


- Break of monotonicity: $EP(x) = 100 * max(x-1, 0) - max(x, 0)$

- We call it *approximate monotone.*

All EP functions (cont.)

- **Approximate** the EP behavior as an *approximate monotonic* function.

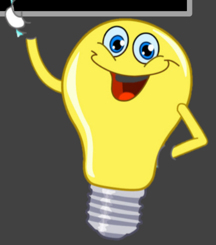


Input (with faults) and output deviation are constrained by the approximate monotonicity

Characteristic of critical faults

Approximate monotone

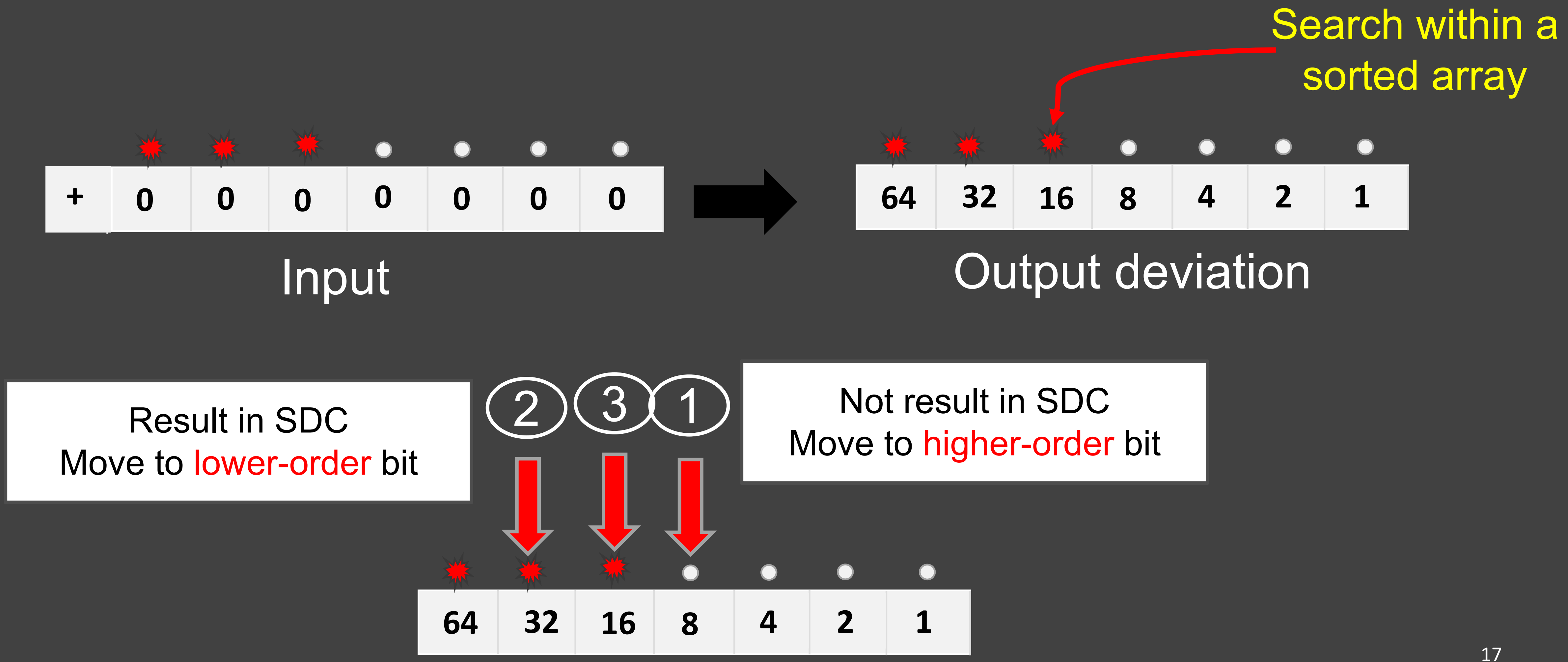
To cause an SDC → Large output dev. → Large Input dev. → Fault at high-order bit



If a fault at high-order bits does not lead to SDC (by FI), faults at lower-order bits would not lead to SDC (without FI), i.e, SDC boundary

Binary fault injection (BinFI)

- Consider the effects of different faults as a sorted array.



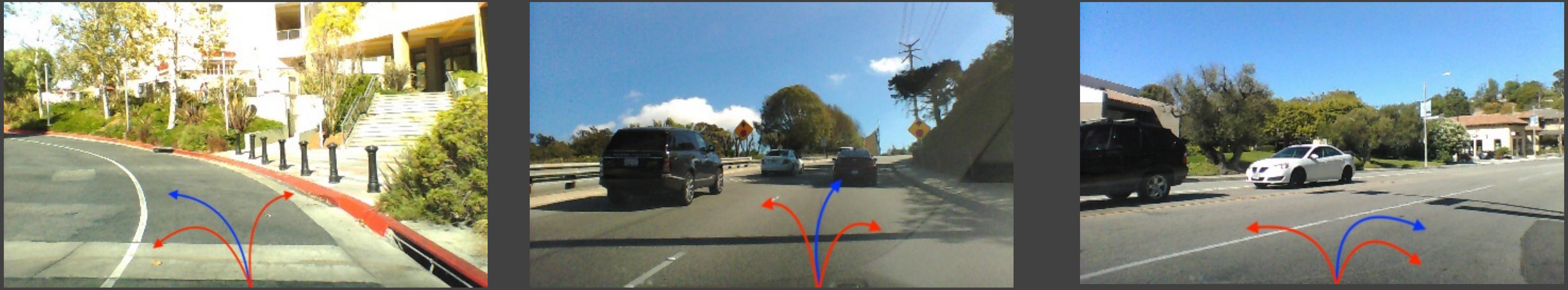
Experimental setup

- *SDC*: Image misclassification; degree of deviation.
- *Fault injection tool*: TensorFI
- *3 FI approaches*: BinFI vs Random FI vs Exhaustive FI

Dataset	Dataset Description	ML model
MNist	Hand-written digits	Neural Network LeNet
Survive	Prediction of patient surivval	kNN
Cifar-10	General images	AlexNet
ImageNet	General images	VGG16
Traffic sign	Real-world traffic signs	VGG11
Driving	Real-world driving frames	Nvidia Dave Comma.ai

FI tool: https://github.com/DependableSystemsLab/TensorFI

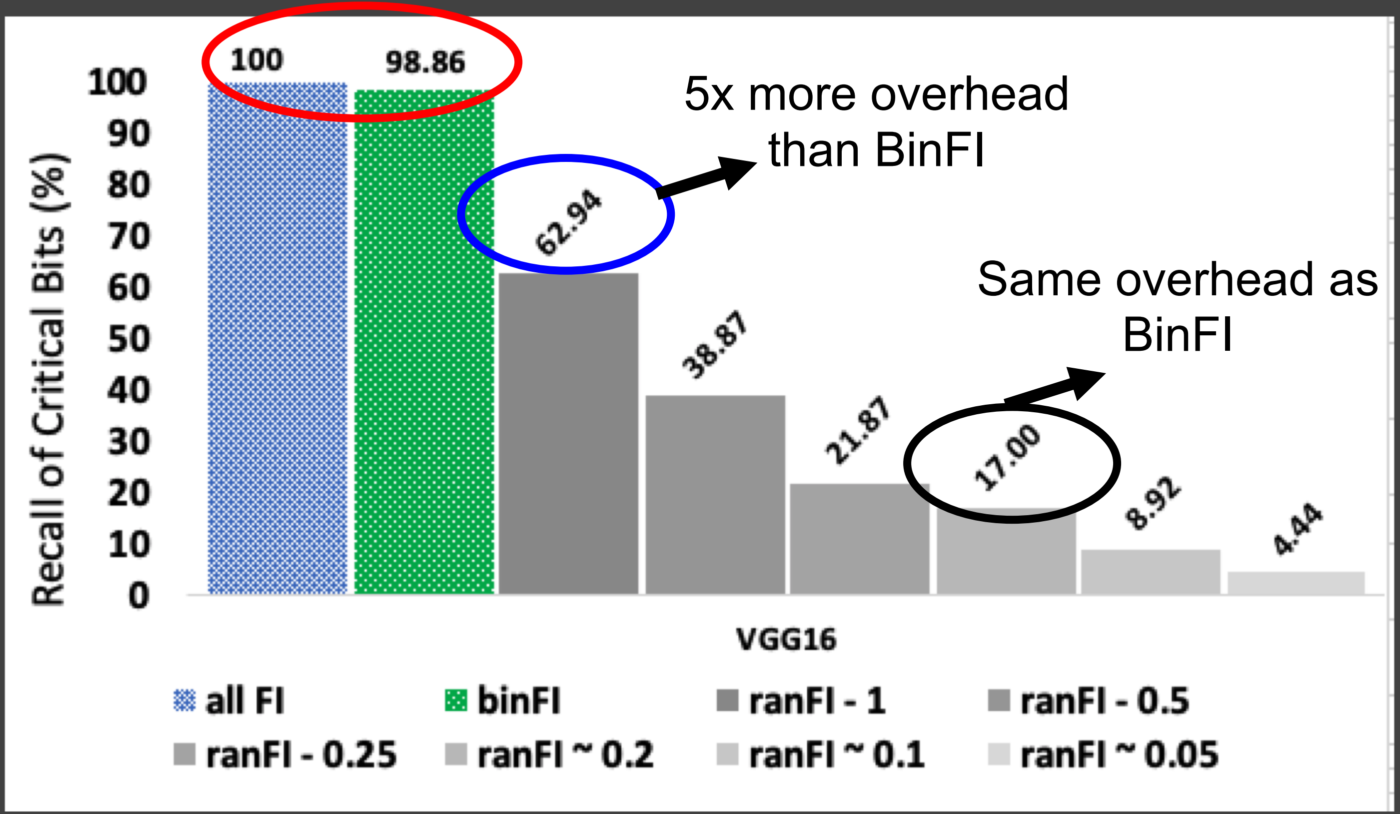
Effects of SDCs



Single bit-flip can cause undesirable output in ML

Identification of critical bits

- **Recall**: 99.56% (average)
- **Precision**: 99.63% (average)

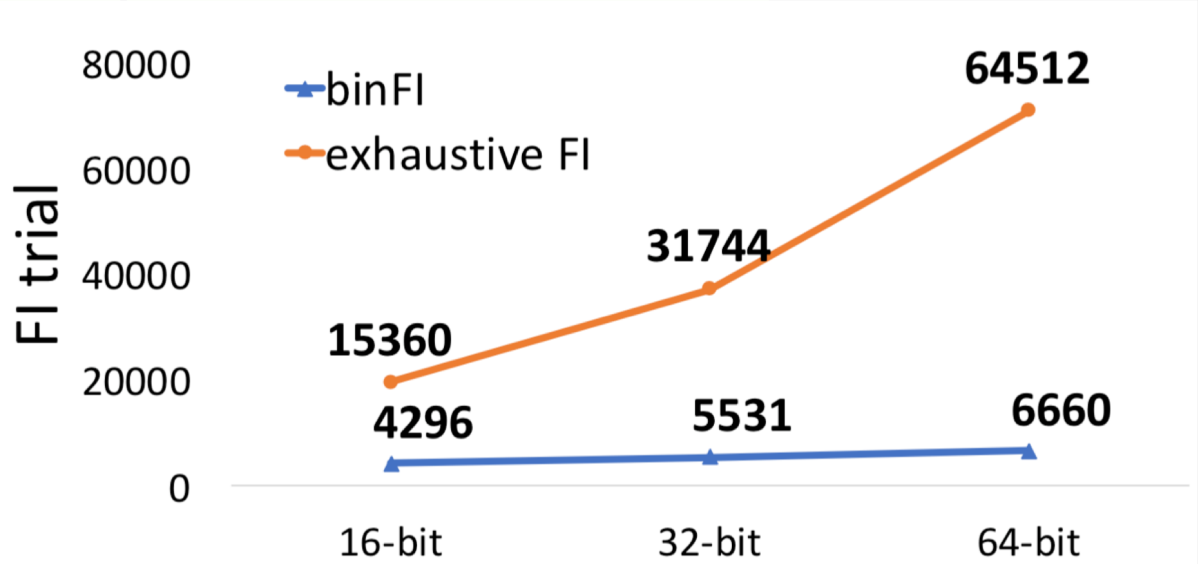


1. BinFI can identify most of the critical bits.
2. Random FI is not desirable for identifying critical bits.

Overhead

Overhead of BinFI is ~20% of that by exhaustive FI

- Data-width: 16, 32, 64 bits.
- 99.5+% recall and precision



1. Overhead of BinFI grows *linearly* with the data width.
2. BinFI is *agnostic* to data width in identifying critical bits.

Summary

- Common ML functions exhibit monotonicity, which constrains the fault propagation behaviors.
- Critical faults in ML programs tend to be clustered: *If a fault at high-order bit does not lead to SDC (by FI), faults at lower-order bits would not lead to SDC (without FI)*

Zitao Chen, Graduate student at University of British Columbia
zitaoc@ece.ubc.ca

Artifact: https://github.com/DependableSystemsLab/TensorFI-BinaryFI