TensorDynamic: Bridging Application- and Instruction-Level Fault Injection for DNN Tensor Core Execution

Yuxiao Jia∗, **Euijun Chung∗,** Huanzhi Pu∗, Ben Feinberg†, Hyesoon Kim∗
∗Georgia Institute of Technology
†Sandia National Laboratories
2026 IEEE International Symposium on Performance Analysis of System and Software

Sandia National Laboratories

Georgia Tech

Background

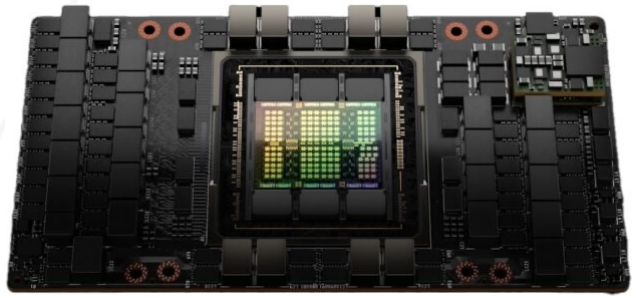
- **GPU Tensor Cores** are vulnerable to **transient faults** – no ECC protection.
  - E.g., Particle strikes, electromagnetic perturbations, etc.

Source: NVIDIA Autonomous Vehicles Safety Report (2025)

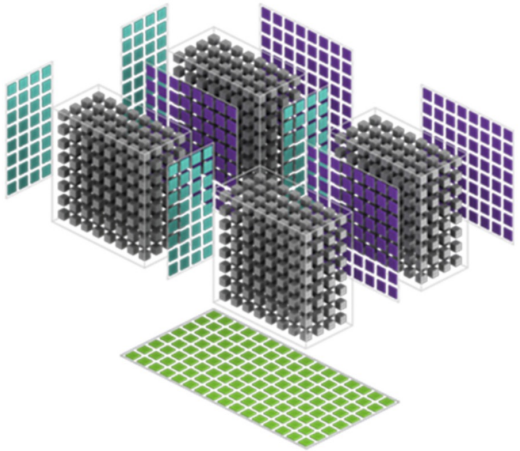
Background

- **GPU Tensor Cores** are vulnerable to **transient faults** – no ECC protection.
  - E.g., Particle strikes, electromagnetic perturbations, etc.

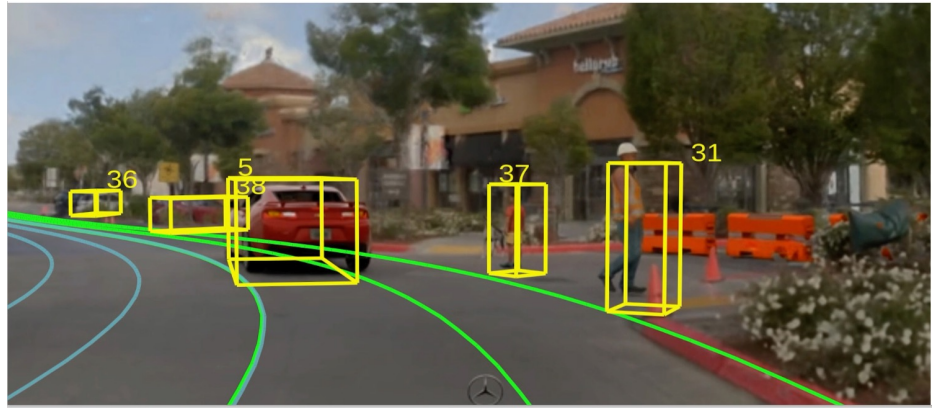
GPU



DNN with tensor core



Object detection in safety-critical environment



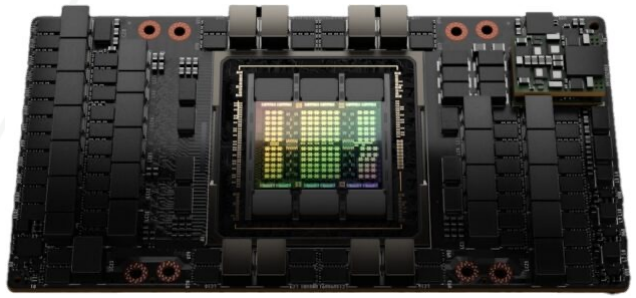
Source: NVIDIA Autonomous Vehicles Safety Report (2025)

Georgia Tech

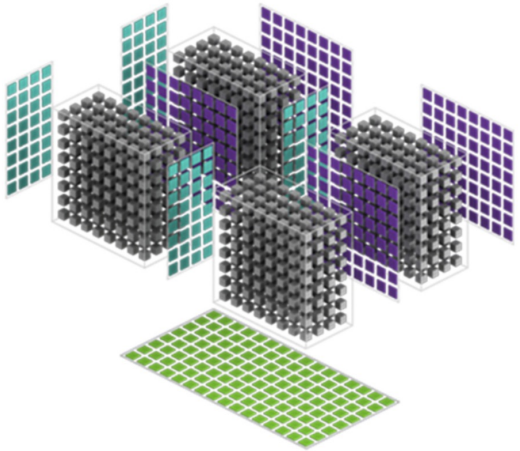
Background

- **GPU Tensor Cores** are vulnerable to **transient faults** – no ECC protection.
  - E.g., Particle strikes, electromagnetic perturbations, etc.

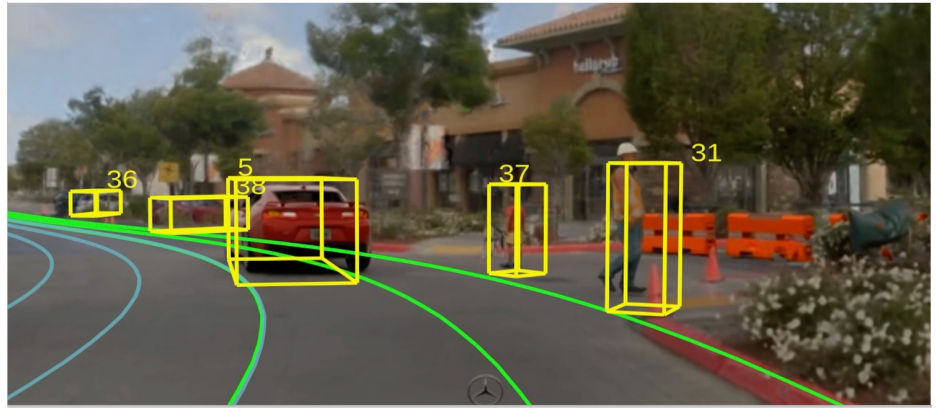
GPU



DNN with tensor core



Object detection in safety-critical environment



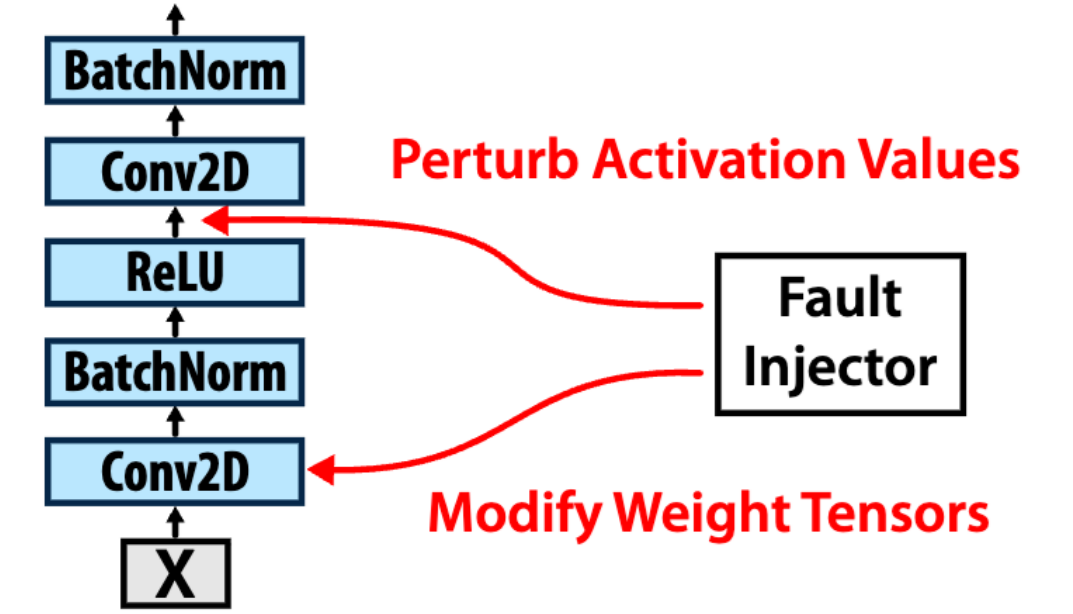
Vulnerable to **transient faults** → System's **safety threat**

Source: NVIDIA Autonomous Vehicles Safety Report (2025)

Georgia Tech

Evaluating DNN resilience

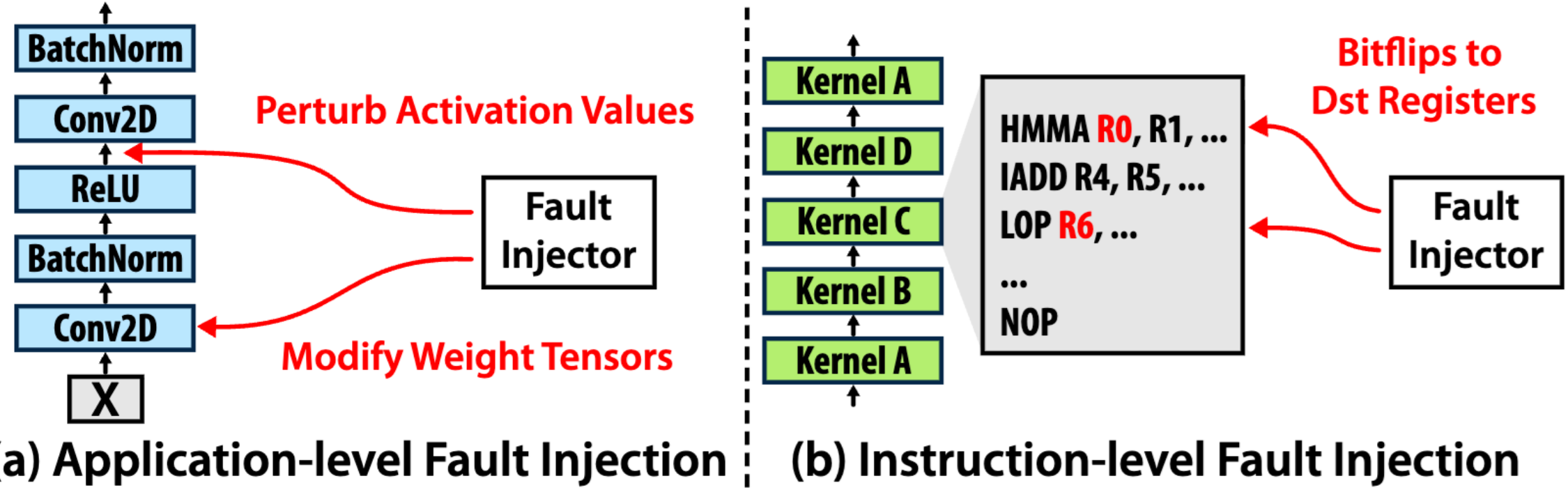
- **Fault injection (FI)** tools can simulate transient faults:



(a) Application-level Fault Injection

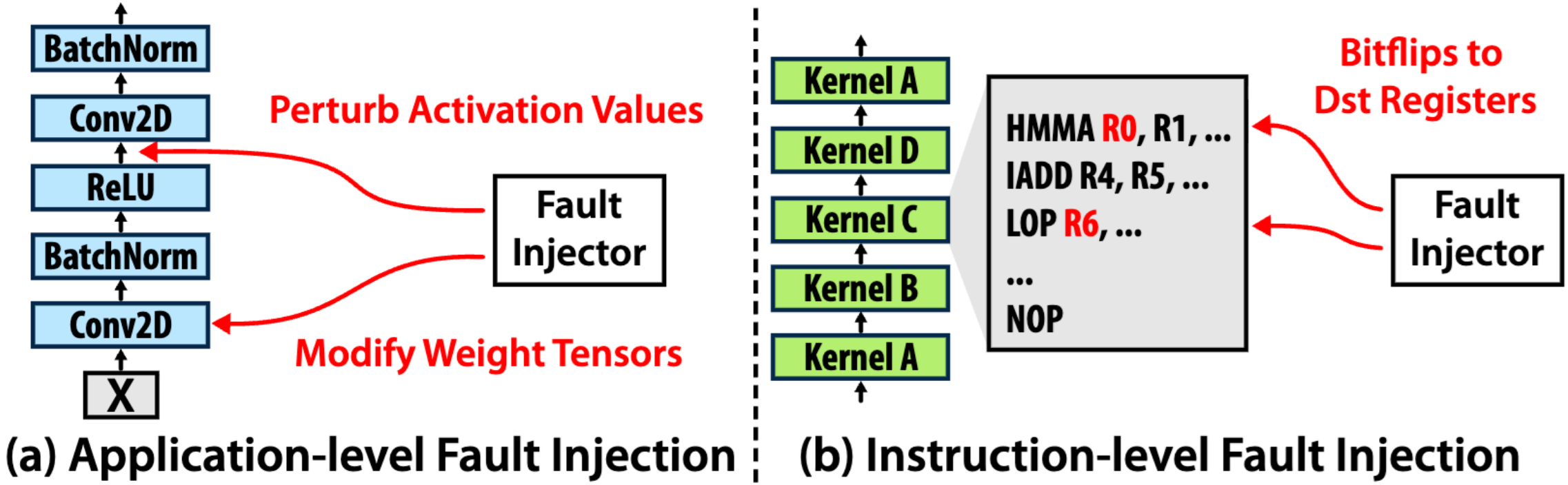
Evaluating DNN resilience

- **Fault injection (FI)** tools can simulate transient faults:



Evaluating DNN resilience

- **Fault injection (FI)** tools can simulate transient faults:

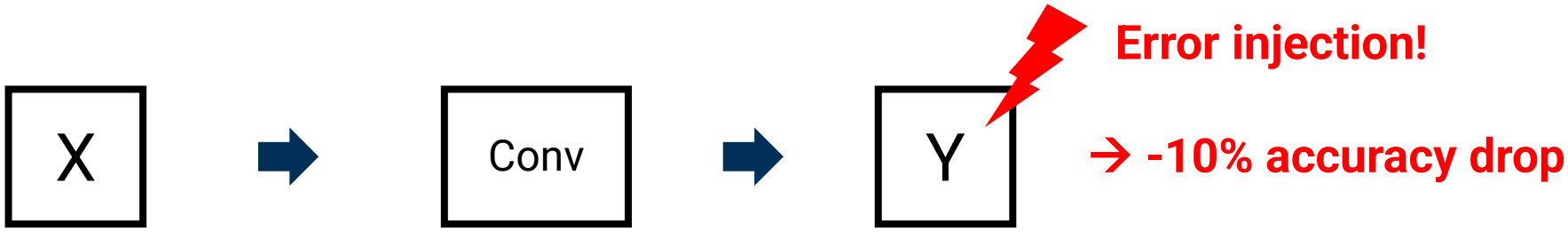


While existing tools expose either **high-level behavior** or **low-level instruction streams, *a gap* exists** between the two.

Application-level FI misses HW Behavior

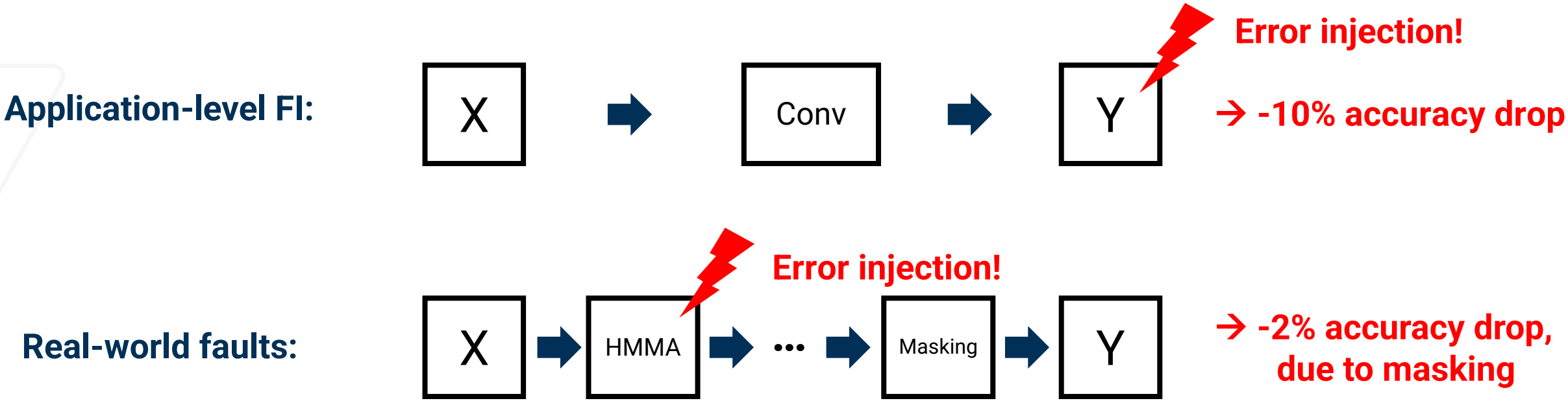
- Application-level FI *directly* perturbs the **final output of operations**.

Application-level FI:



Application-level FI misses HW Behavior

- Application-level FI *directly* perturbs the **final output of operations**.
- However, **not all transient errors** have equal impact.

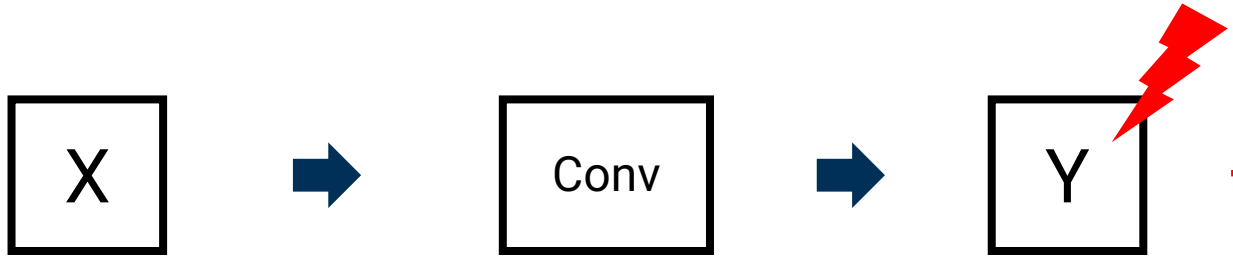


Georgia Tech

Application-level FI misses HW Behavior

- Application-level FI *directly* perturbs the **final output of operations**.
- However, **not all transient errors** have equal impact.

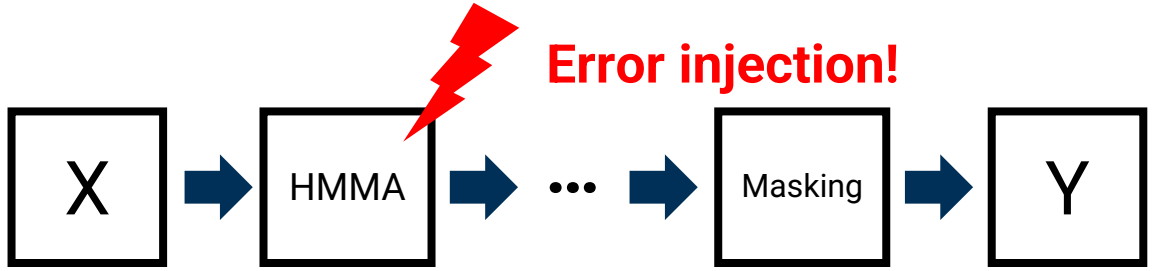
Application-level FI:



Error injection!

→ -10% accuracy drop

Real-world faults:



Error injection!

→ -2% accuracy drop, due to masking

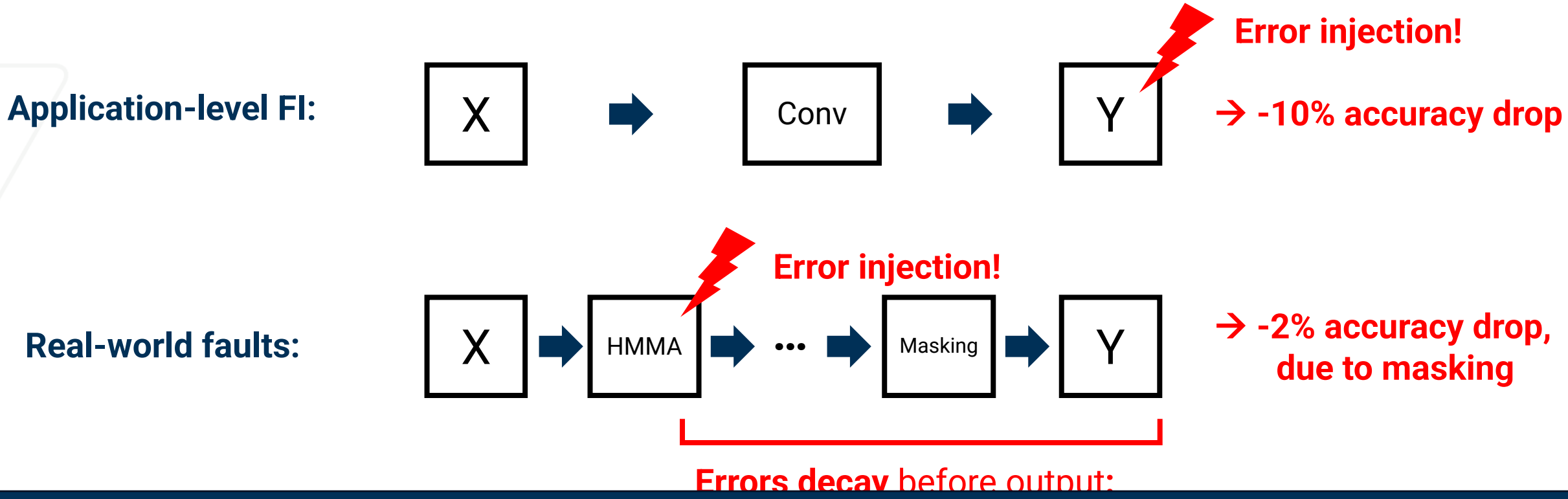
Errors decay before output:

- ⚠ Zero registers
- ⚠ Intermediate values
- ⚠ Masking

Georgia Tech

Application-level FI misses HW Behavior

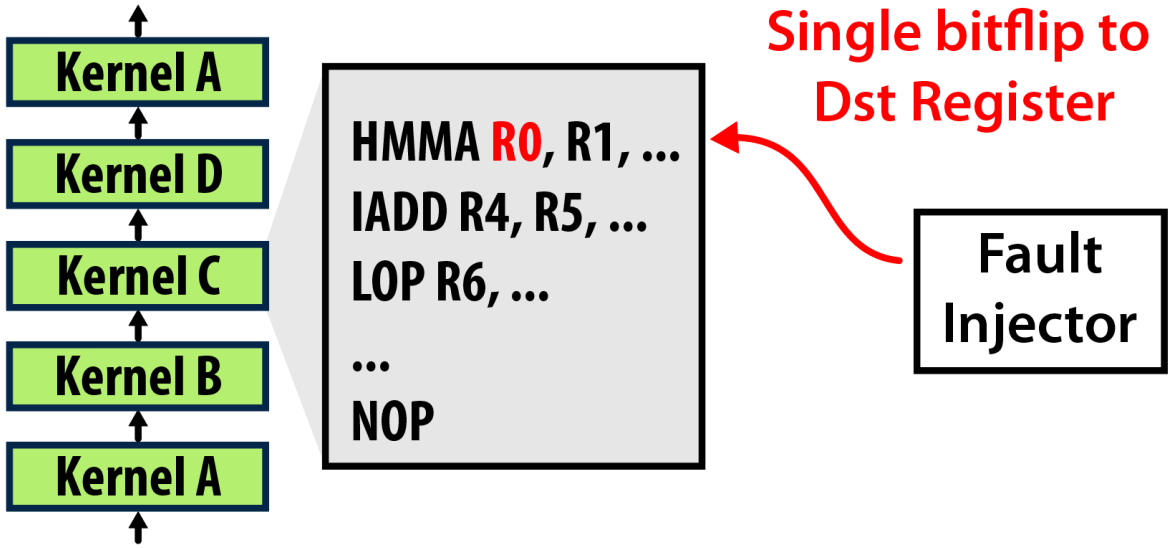
- Application-level FI *directly* perturbs the **final output of operations**.
- However, **not all transient errors** have equal impact.



Errors in **intermediate values** may **attenuate** toward the output.

Instruction-level FI lacks DNN context.

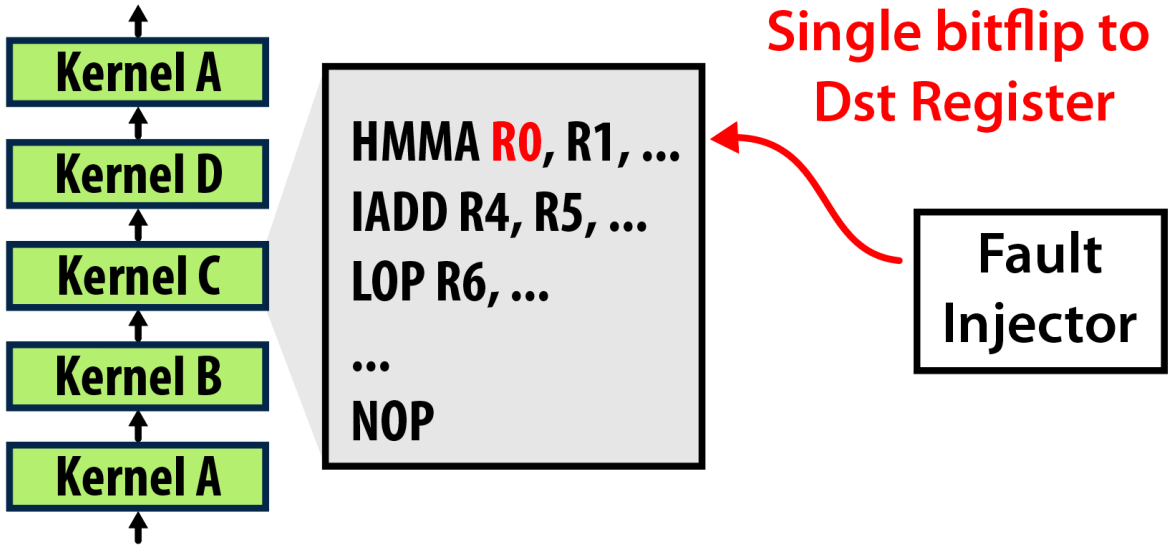
- **Current instruction-level FI tools** cannot distinguish different layer/batch IDs.



Instruction-level Fault Injection

Instruction-level FI lacks DNN context.

- **Current instruction-level FI tools** cannot distinguish different layer/batch IDs.



Instruction-level Fault Injection

✓ **Unaware** about DNN semantics (e.g., Layer ID, Batch ID, etc.)

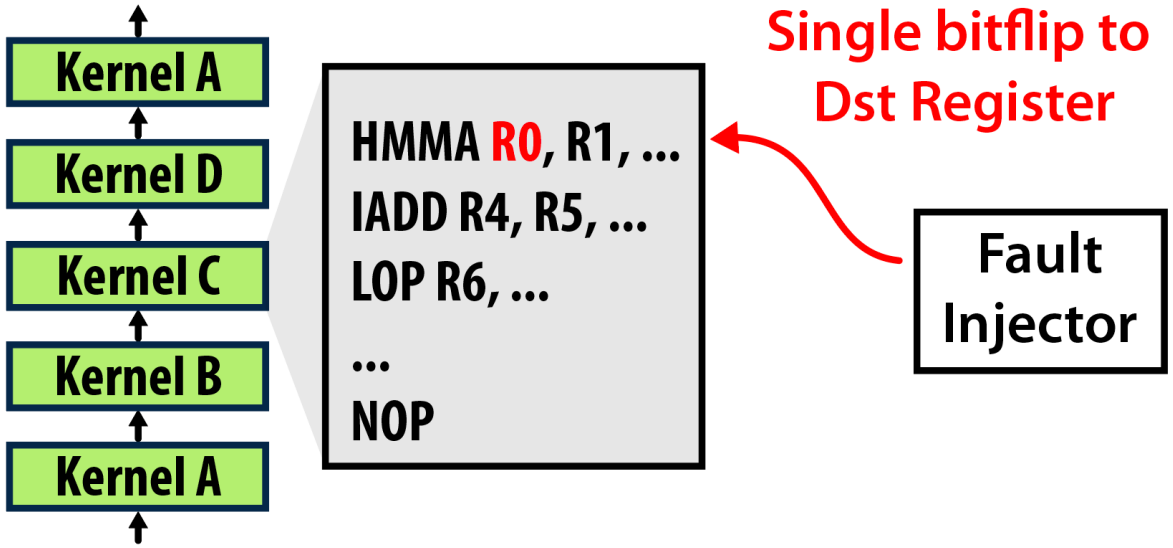
✓ **Multiple runs** required

⚠ Performance **overhead**

Georgia Tech

Instruction-level FI lacks DNN context.

- **Current instruction-level FI tools** cannot distinguish different layer/batch IDs.



Instruction-level Fault Injection

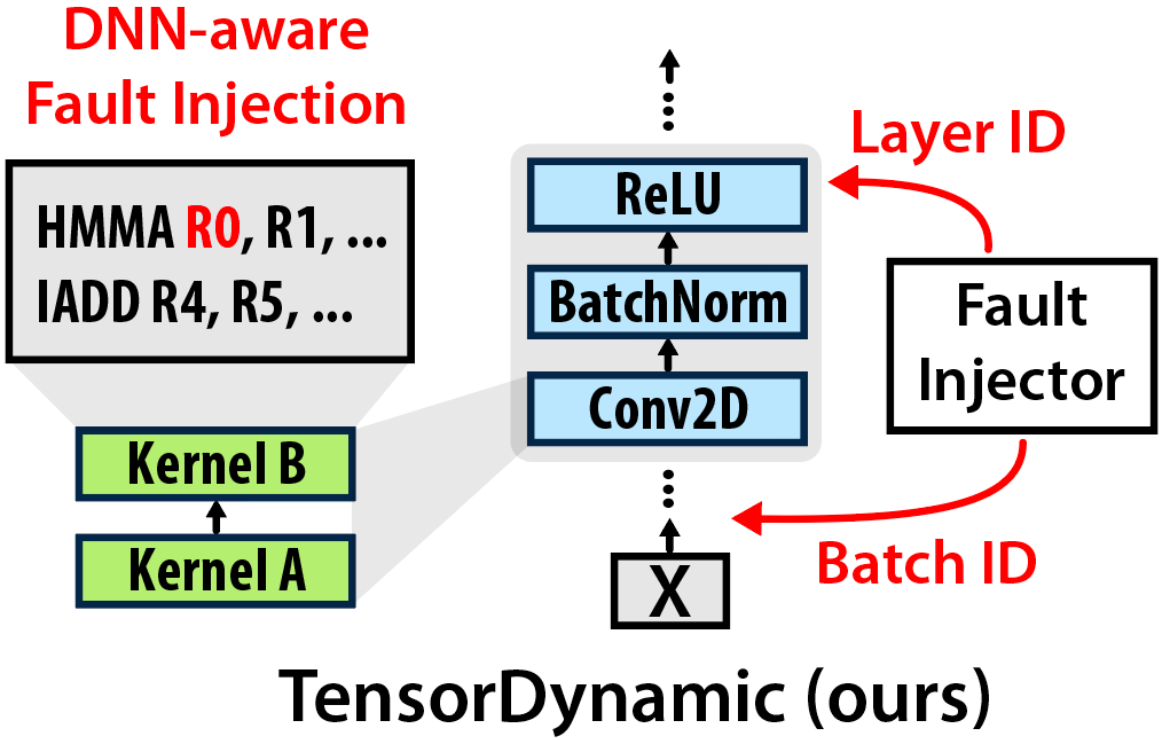
✓ **Unaware** about DNN semantics (e.g., Layer ID, Batch ID, etc.)

✓ **Multiple runs** required

⚠ Performance **overhead**

Existing instruction-level FI tools lack **DNN context** and require a **full re-run for each fault**.

TensorDynamic: bridging *the gap*



Design objectives:

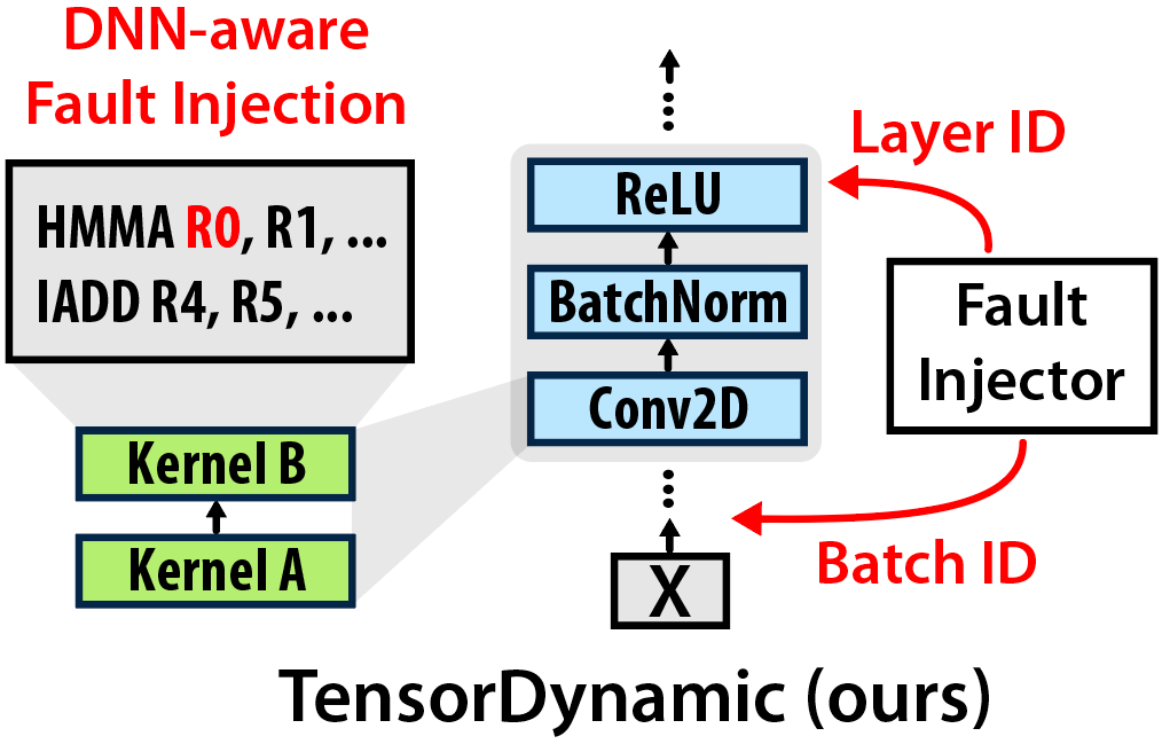
✓ **SASS-level injection** (e.g., HMMA)

✓ **Layer/batch-aware**

✓ **Multi-fault injection** per pass

Georgia Tech

TensorDynamic: bridging *the gap*



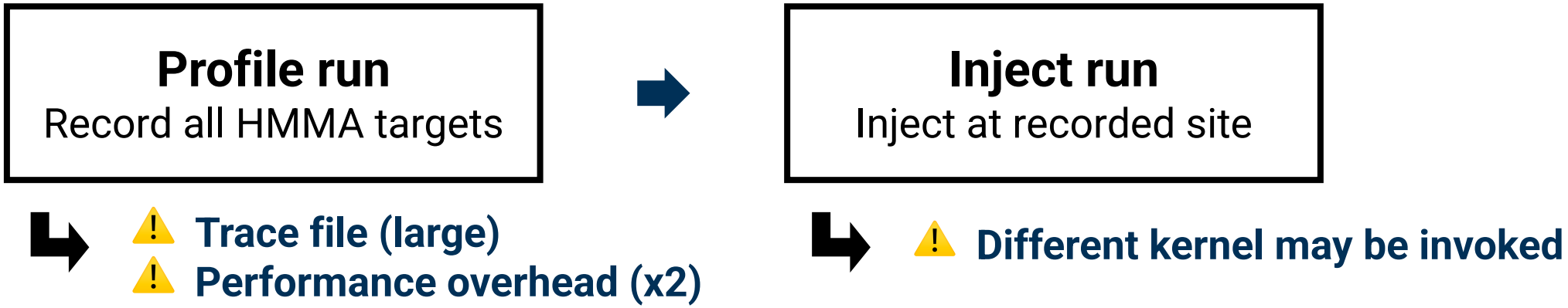
Design objectives:

- ✓ **SASS-level injection** (e.g., HMMA)
- ✓ **Layer/batch-aware**
- ✓ **Multi-fault injection** per pass

TensorDynamic *links* high-level **DNN** behavior to the low-level **instructions**.

TensorDynamic: Dynamic single-pass

- **Previous**: profile *then* inject.



TensorDynamic: Dynamic single-pass

- **Previous**: profile *then* inject.

Profile run
Record all HMMA targets

➡

Inject run
Inject at recorded site

↳ ⚠ **Trace file (large)**
⚠ **Performance overhead (x2)**

↳ ⚠ **Different kernel may be invoked**

- **TensorDynamic (ours)**: profile *and* inject on-the-fly

Profile + Inject
Match target → inject immediately

↳ ✅ **No trace file**
✅ **No kernel mismatches**
✅ **Faster iteration**

Georgia Tech

TensorDynamic: Layer-Aware Kernel Targeting

- **Question**: How to target **Layer 2** in **batch 3**?

Kernel execution stream (Kernel IDs):

Georgia Tech

TensorDynamic: Layer-Aware Kernel Targeting

- **Question**: How to target **Layer 2** in **batch 3**?

Kernel execution stream (Kernel IDs):

✅ **Pytorch profiler + Source code analysis**

```
...
batch #3, layer1.0.conv1: [210, 220]
batch #3, layer2.0.conv1: [260, 270]
...
```

Inject error on Kernel 260-270.

boundary.txt

Georgia Tech

TensorDynamic: Layer-Aware Kernel Targeting

- **Question**: How to target **Layer 2** in **batch 3**?

Kernel execution stream (Kernel IDs):

✅ **Pytorch profiler + Source code analysis**

```
...
batch #3, layer1.0.conv1: [210, 220]
batch #3, layer2.0.conv1: [260, 270]
...
```

Inject error on Kernel 260-270.

boundary.txt

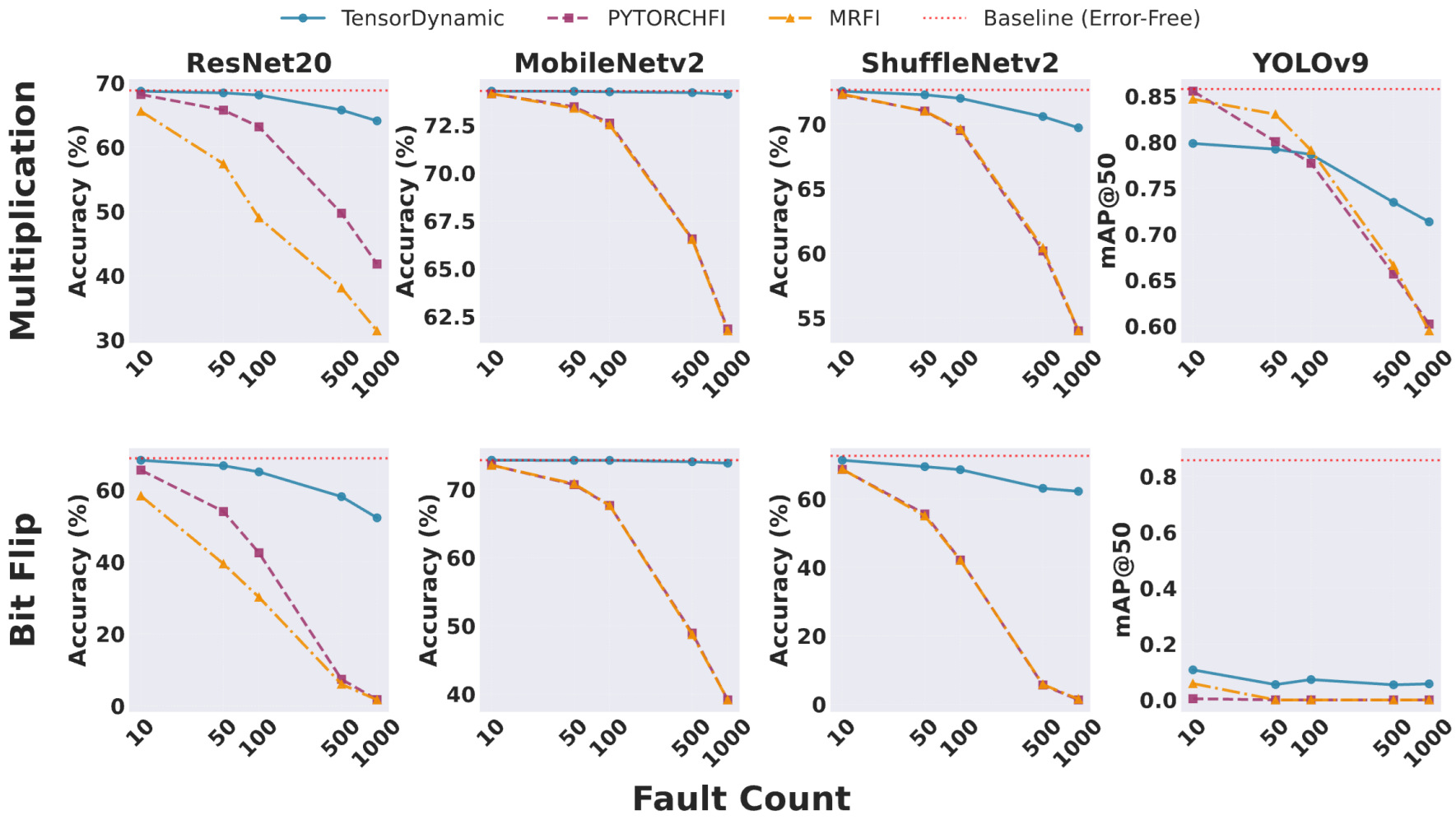
TensorDynamic's boundary config file can target **any layer or batch**.

Experiment setup

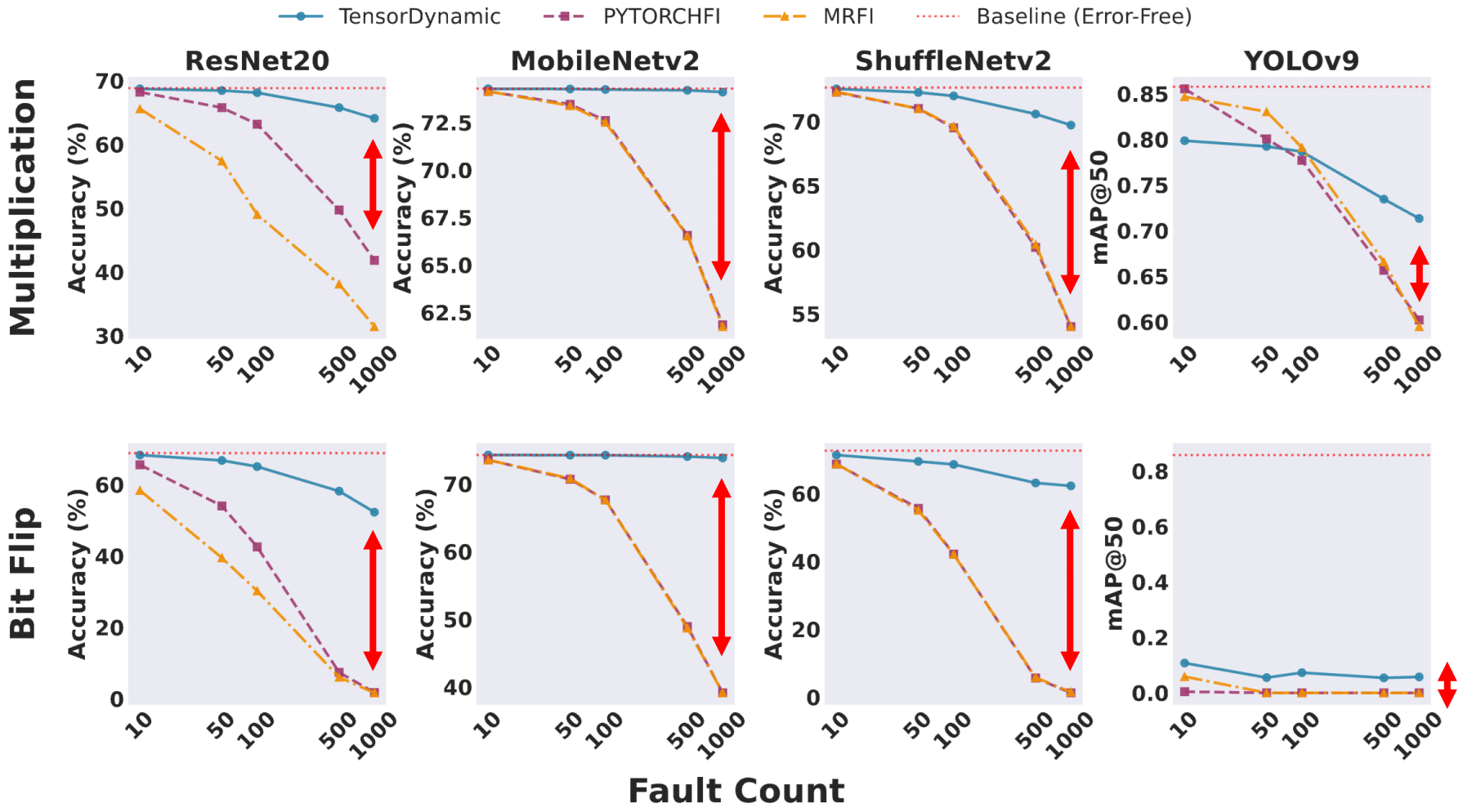
- **Models** (Datasets):
  - ResNet20 (CIFAR-100)
  - MobileNetV2 (CIFAR-100)
  - ShuffleNetV2 (CIFAR-100)
  - YOLOv9 (COCO)
- **Hardware**: NVIDIA A100 (Ampere)
- **Fault models**:
  (1) **Single bitflip** on FP32's high exponent
  (2) **Error multiplication** ($dst' = \alpha \times dst$, $\alpha \in \{10, 50, 100, 1000\}$)
- **Compared tools**: **TensorDynamic** vs. **Application-level** FIs (PyTorchFI*, MRFI**)

*[DSN-W '20] PyTorchFI: A Runtime Perturbation Tool for DNNs
**[VLSI '24] MRFI: An Open Source Multi-Resolution Fault Injection Framework for Neural Network Processing

Evaluation: App-level vs. Instruction-level

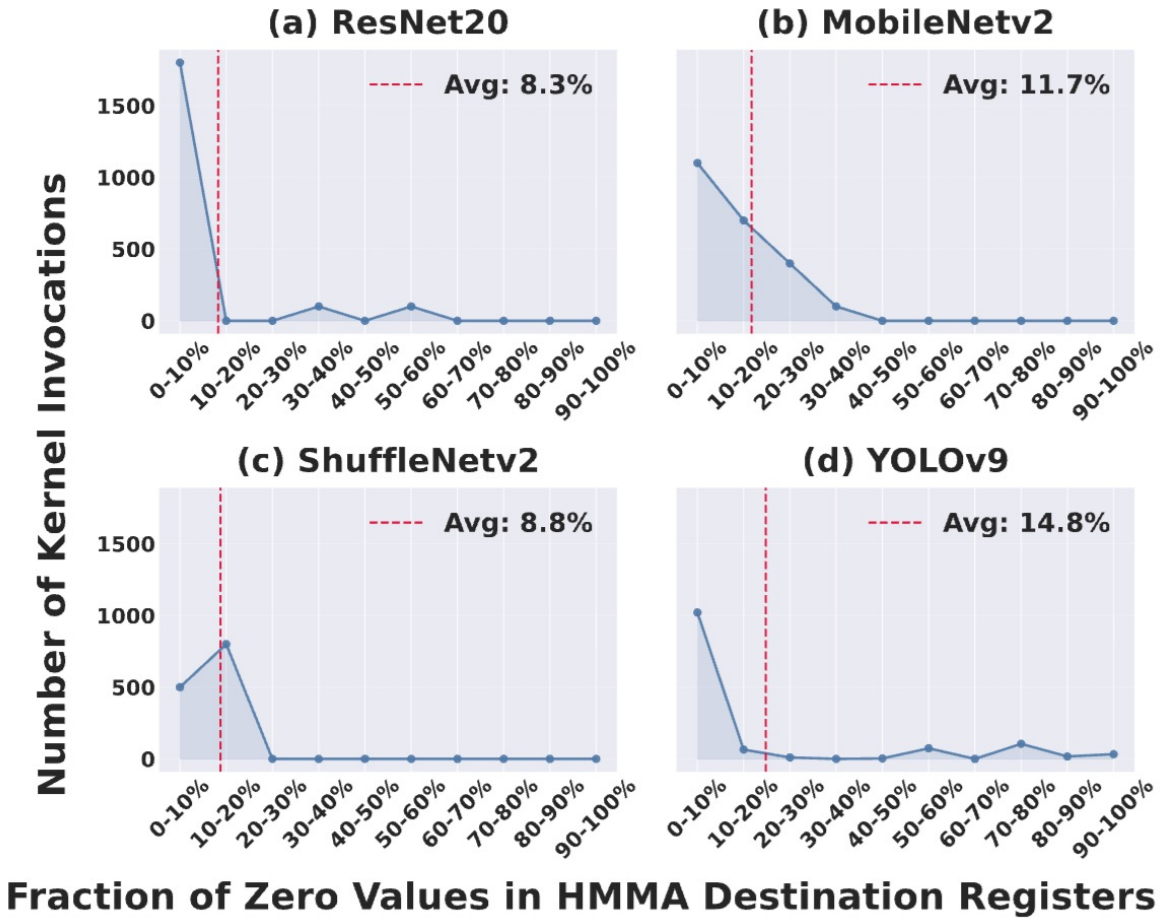


Evaluation: App-level vs. Instruction-level



Application-level FI can **overstate** accuracy drop relative to instruction-level FI.

Evaluation: Hardware effects

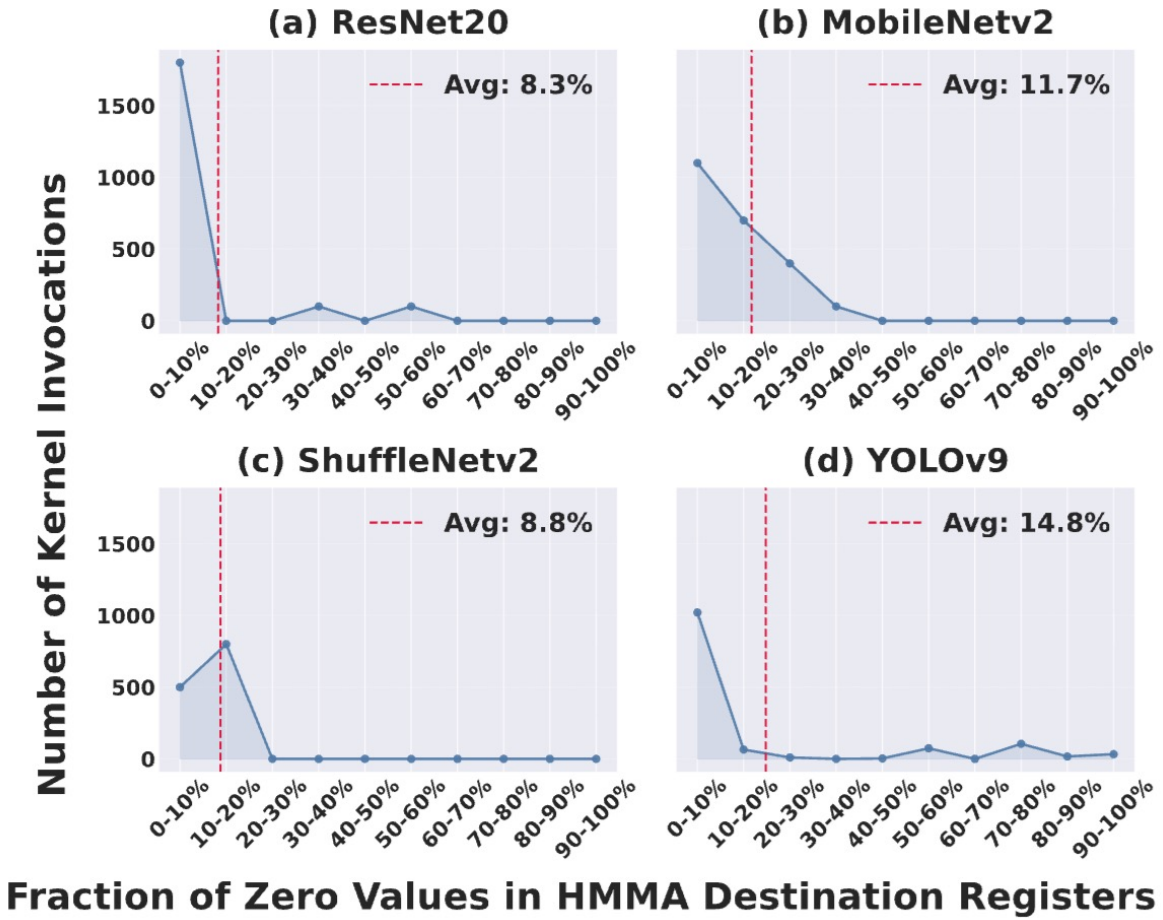


✓ **8.3% - 14.8%** HMMA destination registers are **zeros**.

✓ Zero FP registers are **more tolerable** to transient errors ($\Delta \leq 2$)

Georgia Tech

Evaluation: Hardware effects



✓ **8.3% - 14.8%** HMMA destination registers are **zeros**.

✓ Zero FP registers are **more tolerable** to transient errors ($\Delta \leq 2$)

8–15% of HMMA faults are **silently masked** by sparse destination registers.

Key designs & detailed evaluation

TensorDynamic:

✓ **Multi-fault** injection per run.

✓ **Thread-level** fault targeting.

✓ **Static workflow** for controlled comparisons.

✓ **Algorithm** for runtime execution flow.

Evaluation:

✓ **Fault observability** analysis on Conv and GEMM kernels.

✓ **Static workflow** feature for reproducible experiments.

→ More details in the paper!

Georgia Tech

Conclusion

⚠️ **Fault-injection abstraction** tends to affect DNN reliability evaluations.

⚠️ **Application-level** FI can overestimate vulnerability by bypassing hardware masking.

✅ **TensorDynamic** bridges **DNN-level** semantics and **instruction-level** fault injection with high fidelity and low overhead.

✅ future work - more parameters to explore such as the error injection positions

Paper

Code

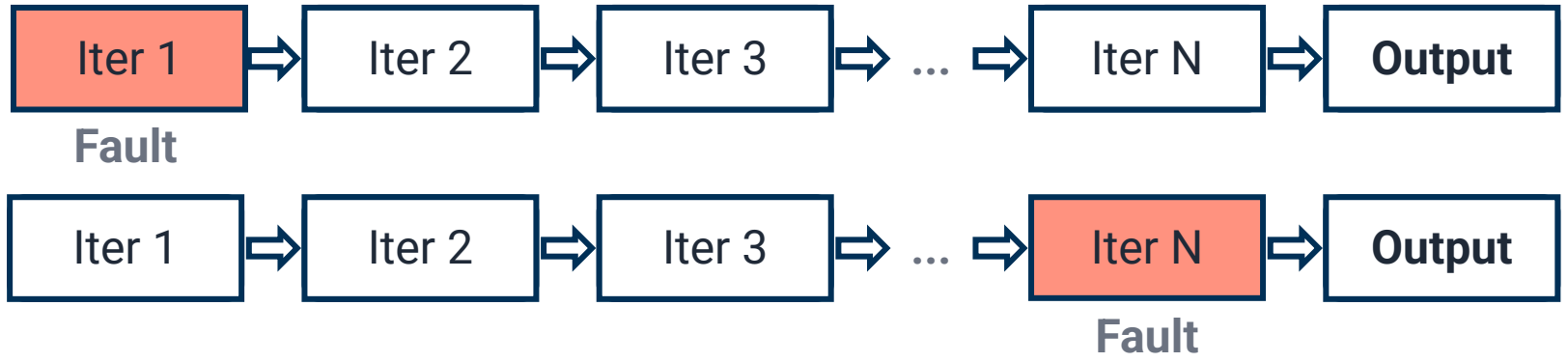
Georgia Tech

Backup slides

Future works & Limitation

Loop iteration position unexplored:

- Early vs. late loop position can affect fault propagation and final impact



- Dynamic mode biases toward early iterations
- This sensitivity was not characterized in the current work

Future work

- Targeted injection at arbitrary K-loop positions
- Loop position × fault impact characterization

Georgia Tech

Error Propagation differs between Conv and GEMM

Metric	Convolution	GEMM
Total output elements	1,638,400	1,638,400
Average insertion rate (%)	0.169	0.169
Observability (%)	85.2	100.2
Mean squared error	0.051	7.206
Mean abs. mult. scale	1.160	1.423

Under identical injection settings, convolution and GEMM show different observability and numerical impact

Dynamic vs. Static Workflow

- **Static workflow:** two-run process (profile → inject)
- **Dynamic workflow:** single-run (profile + inject at the same time)

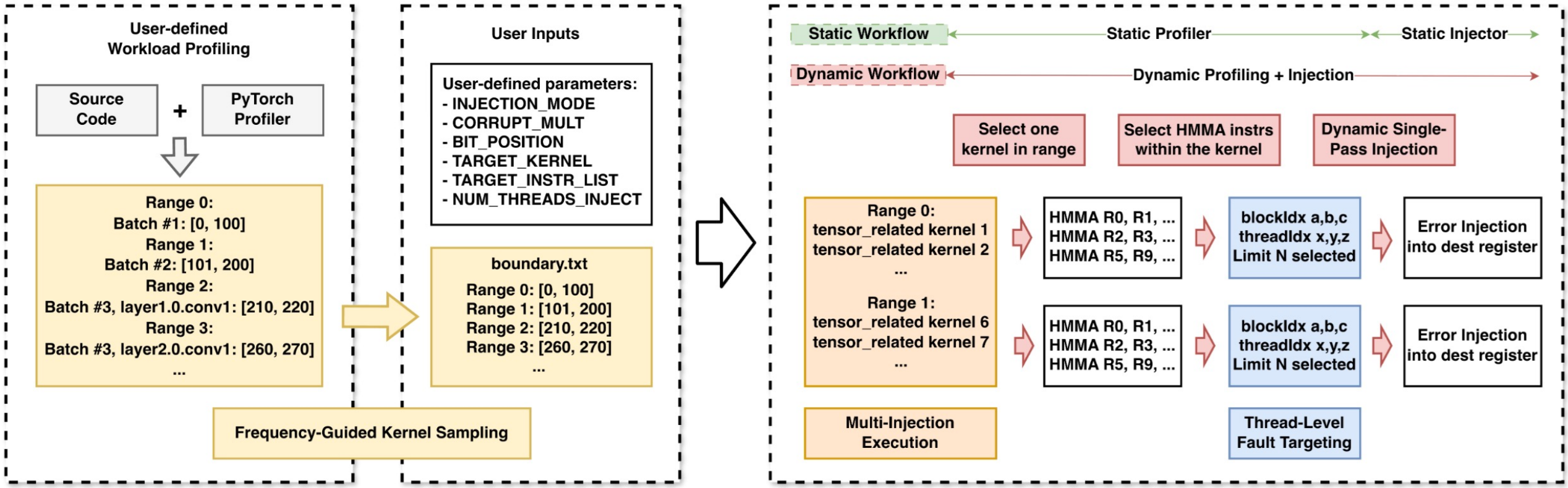
- **TensorDynamic** supports Static workflow to reproduce fault sites for controlled comparisons.

Dynamic vs. Static workflow

TABLE I
COMPARISON BETWEEN DYNAMIC AND STATIC FAULT INJECTION FLOWS IN TENSORDYNAMIC.

Feature	**Dynamic Flow**	**Static Flow**
Profiling	Online (Runtime identification)	Offline (Full enumeration)
Execution	**Single-pass** (Inject immediately)	**Two-pass** (Profile $\rightarrow$ Inject)
Targeting	Implicit (First N available threads)	Precise (User-controlled randomization)
Profile–Injection Misalignment	None (Single-pass guarantee)	Possible (Due to potential scheduling non-determinism)
Overhead	**Low** (No storage or pre-run needed)	**High** (Storage and multi-pass latency)

Georgia Tech



Georgia Tech

Related work on GPU fault injection

TABLE IV
SUMMARY OF RELATED TOOLS.

Work	Batch-level injection	Layer-level injection	Instr-level injection	Thread-level injection
PyTorchFI [21]	◐	✓	✗	✗
MRFI [17]	◐	✓	✗	✗
SASSIFI* [14]	✗	✗	✓	✓
NVBitFI [33]	✗	✗	✓	✓
MPGemmFI [9]	✗	✗	✓	✓
TensorDynamic (ours)	✓	✓	✓	✓

*Deprecated; lack of support for Tensor Cores

Legend: ✓ native; ◐ partial/indirect; ✗ not the focus.

Georgia Tech