OBSan: An Out-Of-Bound Sanitizer to Harden DNN Executables

Yanzuo Chen, Yuanyuan Yuan, Shuai Wang

Department of Computer Science and Engineering

The Hong Kong University of Science and Technology

DNN Executables on the Rise

- Deep learning (DL), Deep neural networks (DNN)
- Deployment Situation
  - Many heterogeneous environments to handle
  - Need for better optimizations tailored to them
- Solution: DL compilers

DNN Executables on the Rise

- Deep learning (DL), Deep neural networks (DNN)
- Deployment Situation
  - Many heterogeneous environments to handle
  - Need for better optimizations tailored to them
- Solution: DL compilers

DL Compilers (in a Nutshell)

- Input: Trained DNN model
- Conversion to graph-aware IR
- Graph- and low-level optimizations
- Output: DNN executables

Hardening Executables

- DL compilers are still relatively new
- Traditional software
  - 🛡️ Hardened: Abnormal behaviors detected & intercepted
  - AddressSanitizer (ASan)
  - UndefinedBehaviorSanitizer (UBSan)
  - and more…
- 🤔 DNN executables?

Hardening Executables

- DL compilers are still relatively new
- Traditional software
  - 🛡️ Hardened: Abnormal behaviors detected & intercepted
  - AddressSanitizer (ASan)
  - UndefinedBehaviorSanitizer (UBSan)
  - and more...
- 🤔 DNN executables?

Can’t we just enable ASan?

- Or: What protection do DNN exe’s need?
- Characteristics of DNN exe’s:
    - Machine-generated code (⇒ rigorous),
    - For math (⇒ pure) functions.
    - ⇒ Anomaly not in code, but *encoded* in data values
    - ⇒ No ASan, etc.

Can’t we just enable ASan?

- Or: What protection do DNN exe’s need?
- Characteristics of DNN exe’s:
  - Machine-generated code (⇒ rigorous),
  - For math (⇒ pure) functions.
  - ⇒ Anomaly not in code, but *encoded* in data values
  - ⇒ No ASan, etc.

Out-of-Bound (OOB) Behaviors

- Generalization of “anomalies in data values”
- Typically cause undesired outputs
- Idea: Normal behaviors captured by bounded metrics
  - Neuron activations
  - Gradients in backpropagation
- Metrics OOB ⇒ Abnormal behaviors

OBSan: An Out-Of-Bound Sanitizer

- Motivation: Capture normal behaviors
- Alerts when OOB behaviors discovered
- First work to harden DNN exe's
- Use cases
  - Detecting unwanted/malicious inputs,
  - Mitigating blackbox attacks,
  - Enabling feedback-driven fuzzing, ...

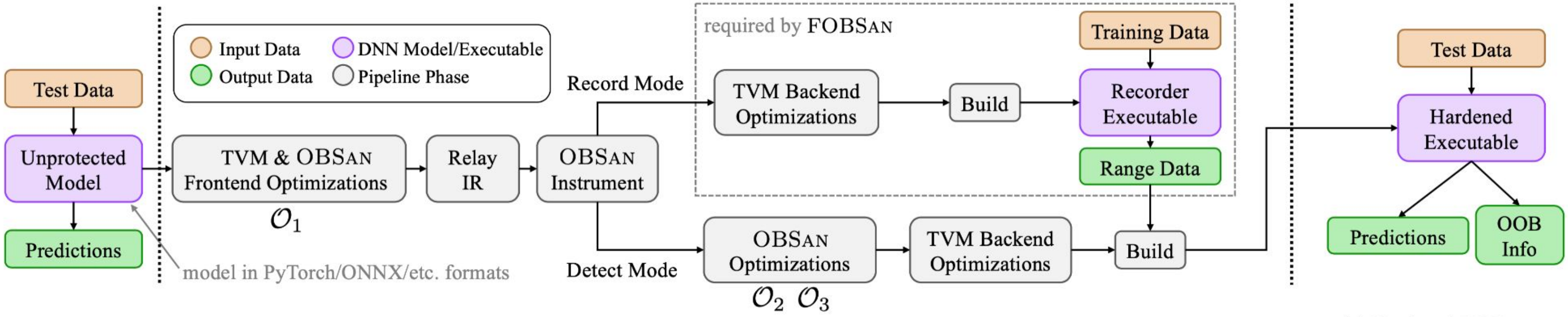
Wait, did you say AE detection?

- Many prior works to detect adversarial examples (AE)
- It’s difficult to apply them here
    - DL compilers’ inability to support
    - Effectiveness vs *efficiency* (As high as 7000% overhead)

OBSan: Design & Implementation

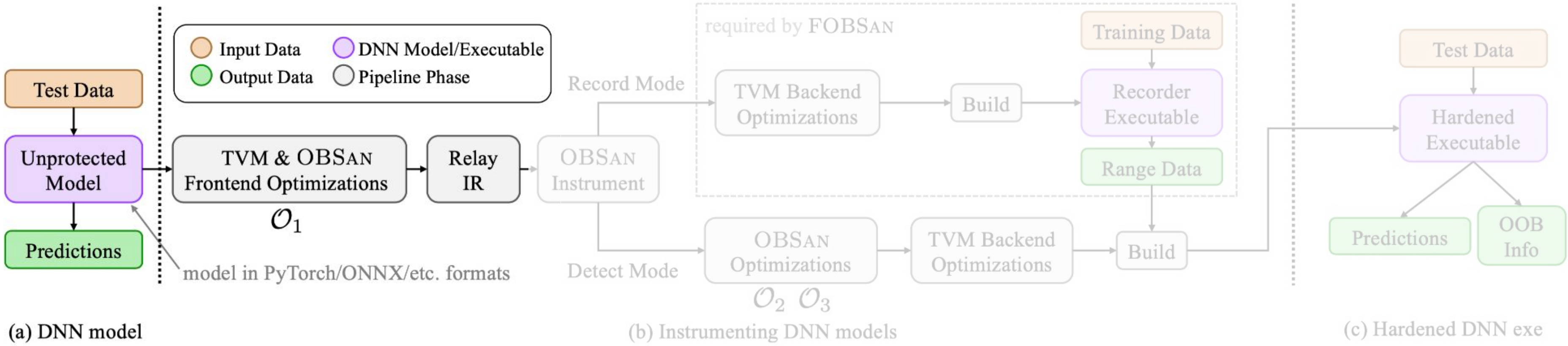
- Variants
  - FOBSan: Based on (forward) neuron activations
  - BOBSan: Based on (backward) gradients
- Currently on TVM; portable

OBSan: Design & Implementation

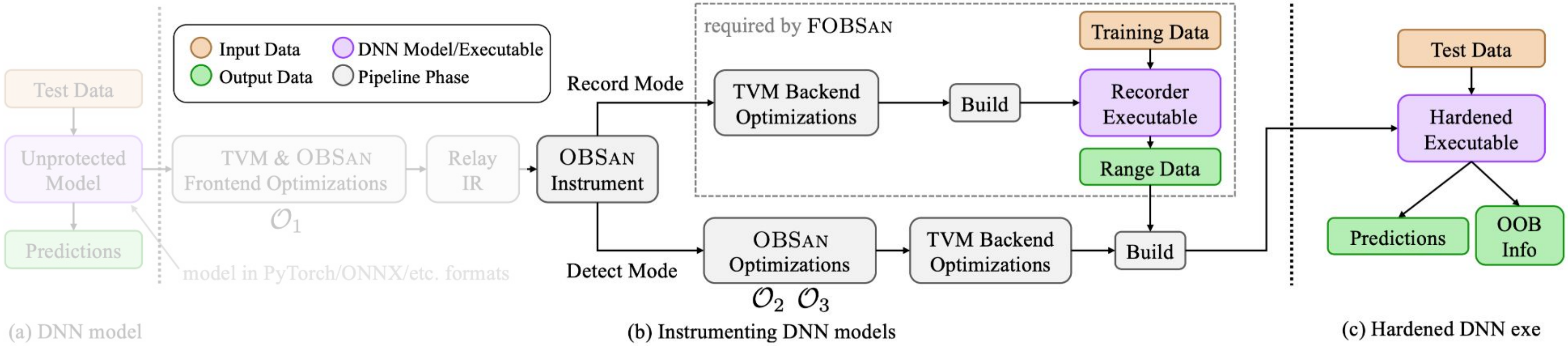


(a) DNN model (b) Instrumenting DNN models (c) Hardened DNN exe

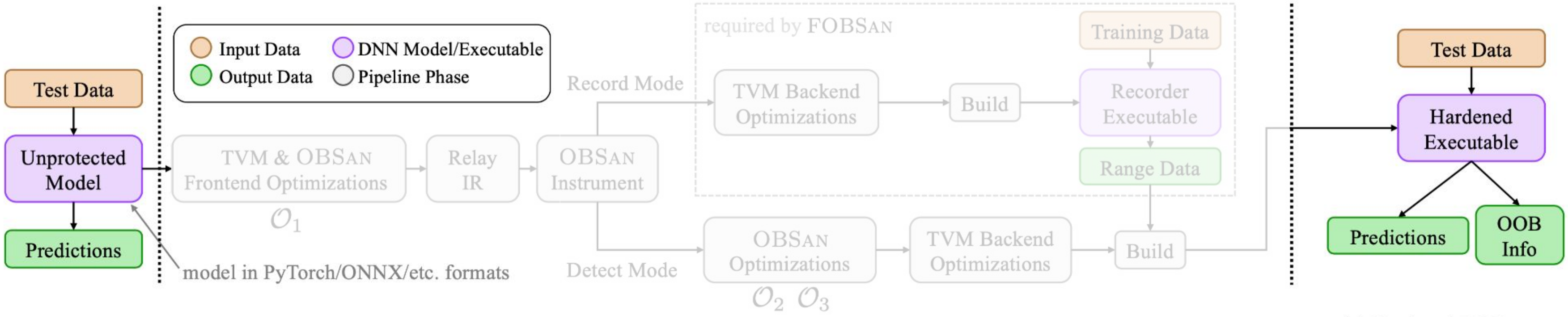
OBSan: Design & Implementation



OBSan: Design & Implementation



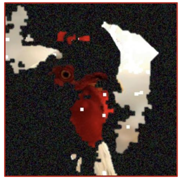
OBSan: Design & Implementation



(a) DNN model (b) Instrumenting DNN models (c) Hardened DNN exe

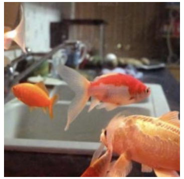
Evaluation: OOB Detection

Normal

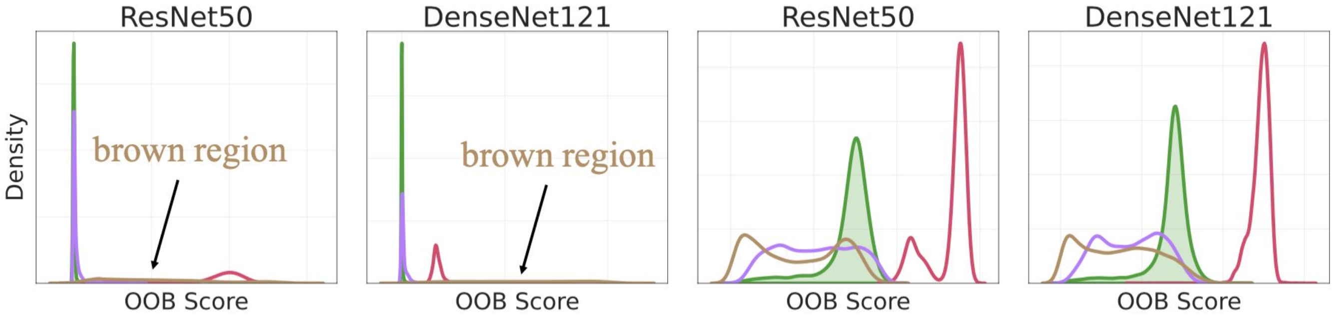


Broken

AE



Undef



(a) OOB score distributions produced by FOBSAN.

(b) OOB score distributions produced by BOBSAN.

	perception-broken	undefined	AE
FOBSAN	✓	✕	✓
BOBSAN	△	△	✓

Evaluation: Performance Overhead

- Optimizations
    - Quantization
    - Checks debloating
    - Parameter optimization for BOBSan
- Overhead: FOBSan → 48%; BOBSan → -34%
- Comparison with existing methods
    - Faster than the most accurate (10x)
    - More accurate than the fastest (e.g. FP 1% vs 20%)
    - OBSan strikes a balance

Evaluation: Performance Overhead

- Optimizations
  - Quantization
  - Checks debloating
  - Parameter optimization for BOBSan
- Overhead: FOBSan → 48%; BOBSan → -34%
- Comparison with existing methods
  - Faster than the most accurate (10x)
  - More accurate than the fastest (e.g. FP 1% vs 20%)
  - OBSan strikes a balance

Downstream Applications

- Feedback-Driven Fuzzing
  - Extend OBSan to output neuron coverage data
  - 10x more mispredictions triggered (⇒ effective fuzzing)

Downstream Applications (cont.)

- Online AE attack mitigation
  - Attacker: SoTA blackbox AE generation algorithm
  - No access to model parameters
  - Makes queries to generate AE inputs
- FOBSan + BOBSan = HOBSan (Hybrid OBSan)
  - 56~95% attacks intercepted
  - Up to 9x more #queries needed

Downstream Applications (cont.)

- Online AE attack mitigation
  - Attacker: SoTA blackbox AE generation algorithm
  - No access to model parameters
  - Makes queries to generate AE inputs
- FOBSan + BOBSan = HOBSan (Hybrid OBSan)
  - 56~95% attacks intercepted
  - Up to 9x more #queries needed

Conclusion

- Emerging trend: DNN executables
- Need: More security protection
- OBSan: First work to harden DNN executables
  - Design, implementation, results, downstream applications
  - 👀 Potential

More Info

- Source code, other materials
  - sites.google.com/view/oob-sanitizer ( )
- Contact me
  - Yanzuo Chen (ychenjo@cse.ust.hk)

Performance Data

OBSAN variant	Model	Infer. time (ms)		OBSAN Overhead	$\mathbf{FP}_{norm}$ ratio	$\mathbf{FN}_{ae}$ ratio	$\mathbf{FN}_{pb/ud}$ ratio
		Vanilla	OBSAN				
FOBSAN w/ opt.	ResNet50	1.22	2.19	79.51%	1.40%	0.20%	49.79%
	GoogLeNet	3.79	3.12	-17.68%	2.41%	0.00%	26.02%
	DenseNet121	2.65	4.80	81.13%	1.21%	5.11%	21.43%
	Average	**2.55**	**3.37**	**47.65%**	**1.67%**	**1.77%**	**32.41%**
FOBSAN w/ opt.	ResNet50	1.22	0.82	-32.79%	6.31%	0.00%	65.79%
	GoogLeNet	3.79	1.67	-55.94%	9.38%	0.00%	75.96%
	DenseNet121	2.65	2.28	-13.96%	4.72%	9.64%	73.62%
	Average	**2.55**	**1.59**	**-34.23%**	**6.80%**	**3.21%**	**71.79%**

Setup of online AE attacker