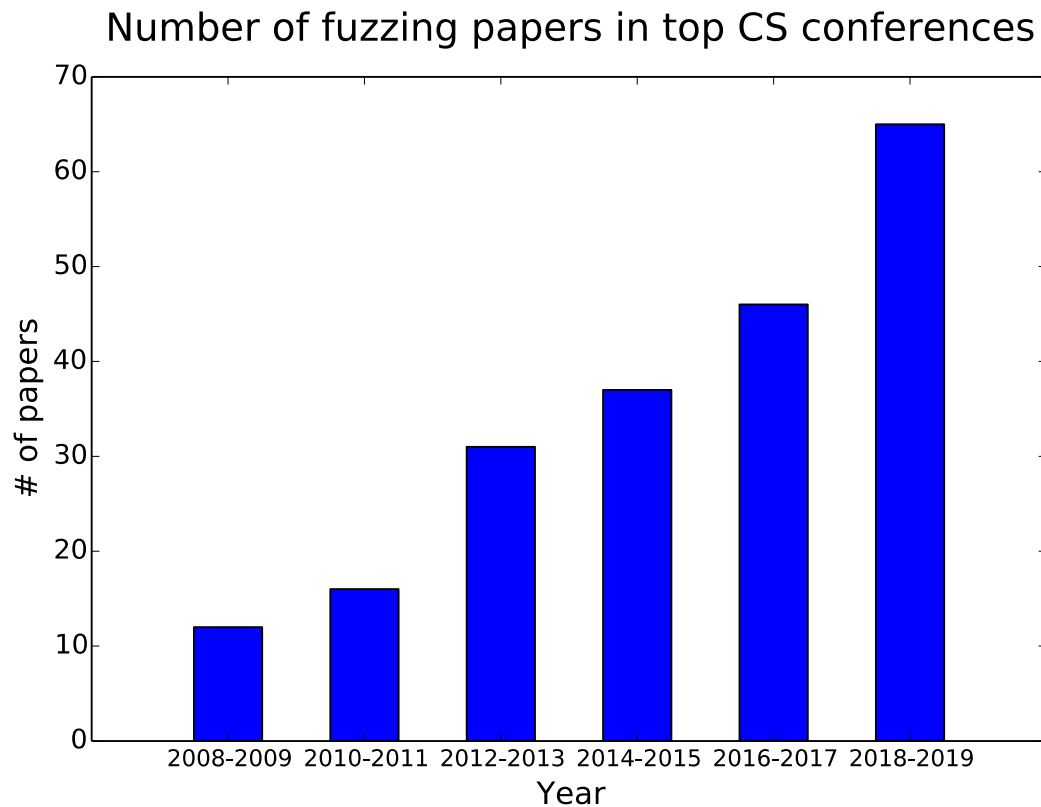


NEUZZ: Efficient Fuzzing with Neural Program Smoothing

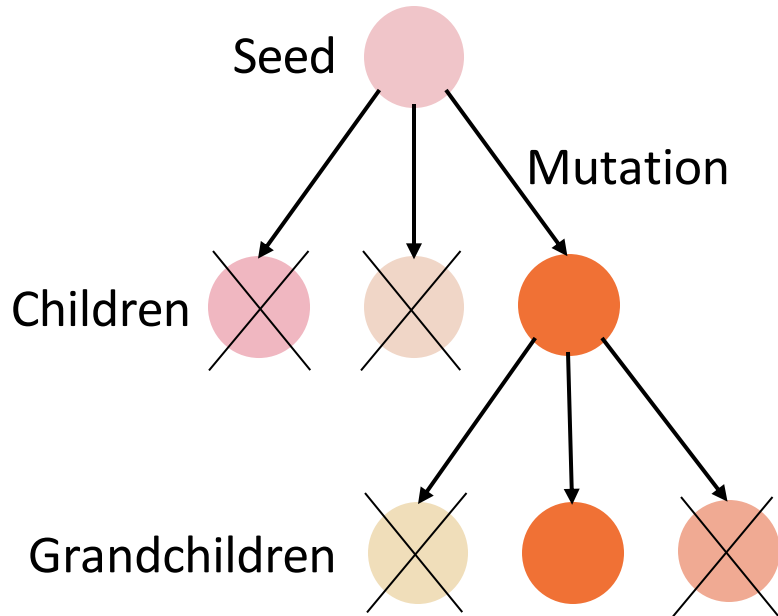
Dongdong She, Kexin Pei, Dave Epstein, Junfeng Yang, Baishakhi Ray, and Suman Jana
Columbia University

Fuzzing: a popular way to uncover bugs



[Liang et al. 2019]

Evolutionary Fuzzing



Advantage: easy to implement

Disadvantage: **inefficient**

- Random mutations are not effective
- Often get stuck in long sequence of wasteful mutations

Hard to find scalable and adaptive heuristics for guided mutation

A new approach to fuzzing

Fuzzing: An Optimization Problem

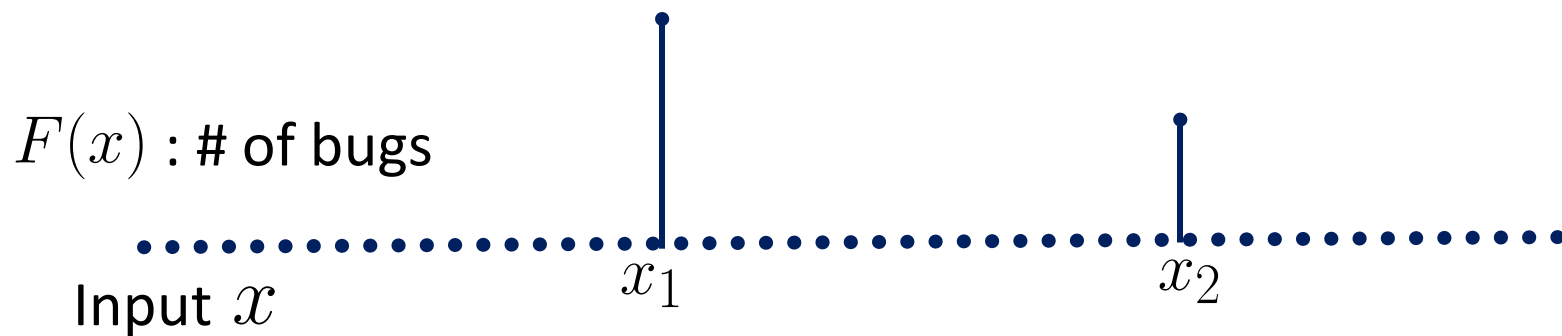
- x \Rightarrow a program input $x \in X$
- $F(x)$ \Rightarrow # of bugs found by input x
- $C(X)$ \Rightarrow generate K inputs from input space X

$$\text{Maximize } \sum_{x \in C(X)} F(x)$$

Find $\mathbf{C(X)}$ that can maximize total no. of bugs

$F(x)$ is discrete and hard to optimize

Fuzzing: An Optimization Problem



Hard to find inputs like x_1 and x_2
among flat plateaus

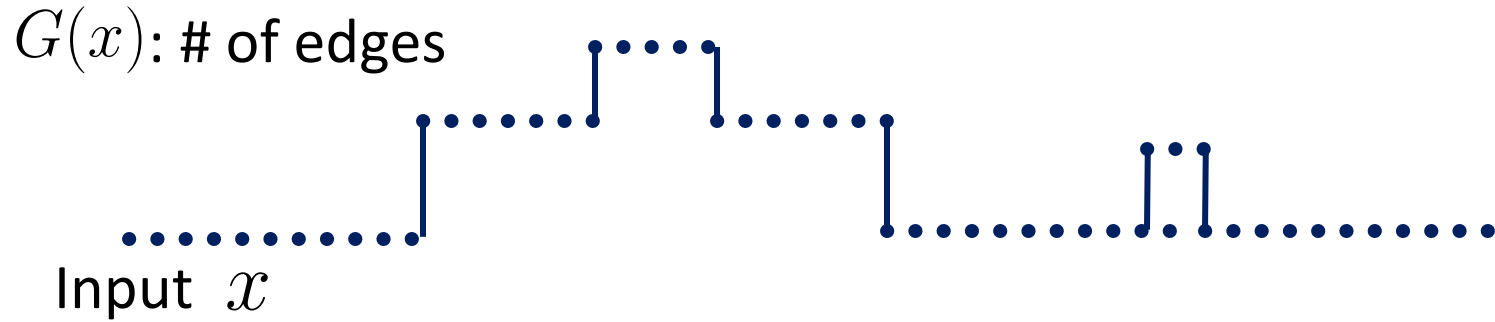
Fuzzing: An Optimization Problem

- x \Rightarrow a program input $x \in X$
- $G(x)$ \Rightarrow **edge coverage** of input x
- $C(X)$ \Rightarrow generate K inputs from input space X

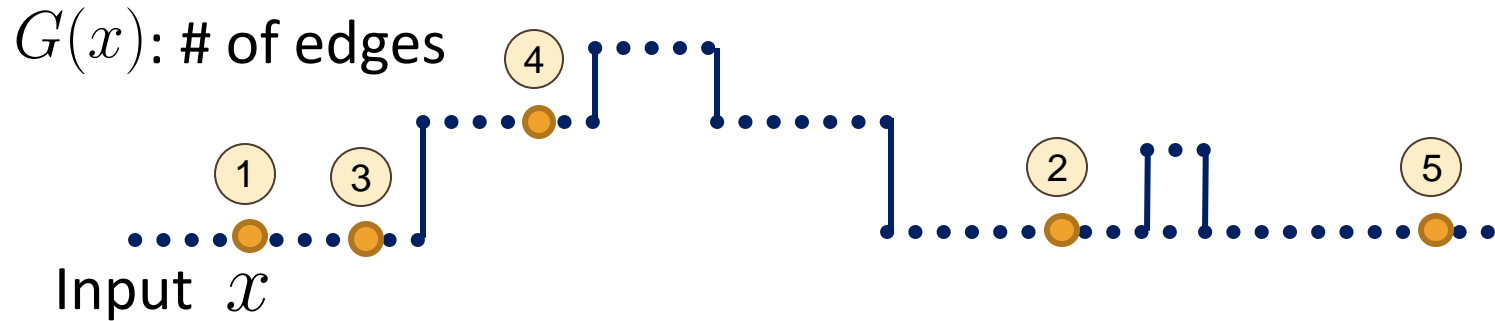
$$\text{Maximize } \sum_{x \in C(X)} G(x)$$

Find $C(X)$ that can maximize total number of **edges**

Fuzzing: An Optimization Problem



Evolutionary optimization

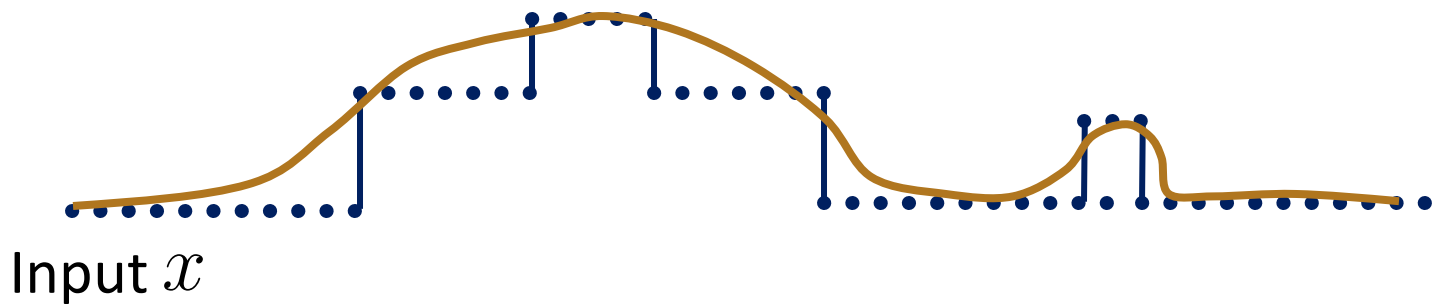


Random mutation is not efficient

Gradient-guided Optimization

Smooth Approximation + Gradient-guided Mutation

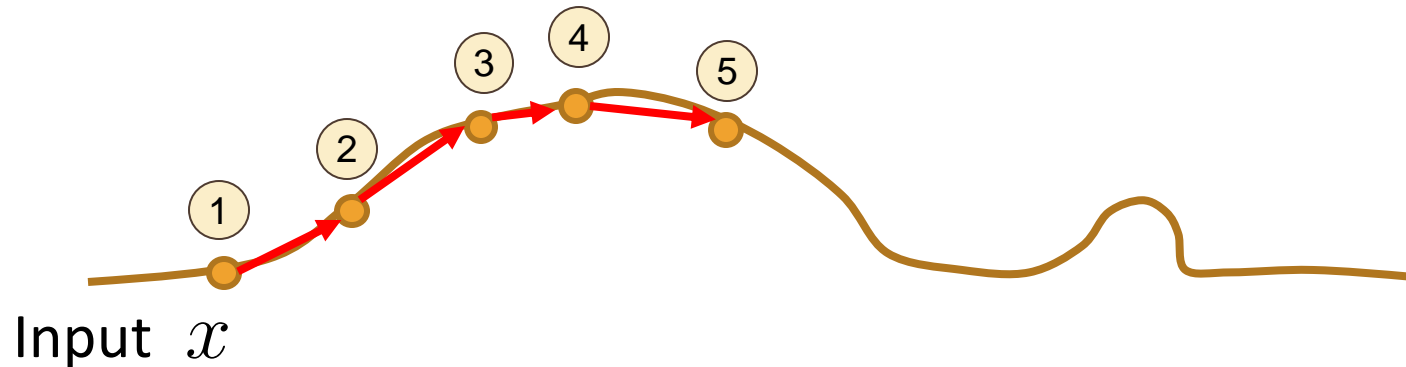
$H(x)$: smooth approximation of $G(x)$



Gradient-guided Optimization

Smooth Approximation + Gradient-guided Mutation

$H(x)$: smooth approximation of $G(x)$



Smooth Approximation

Problem:

How to smoothly approximate $G(x)$?

Universal Approximation Theorem:

A NN can approximate any continuous function

Neuzz Solution:

Use a NN to learn a smooth $H(x)$

Gradient-guided Mutation

Why gradient guidance?

Gradient indicates critical parts of input

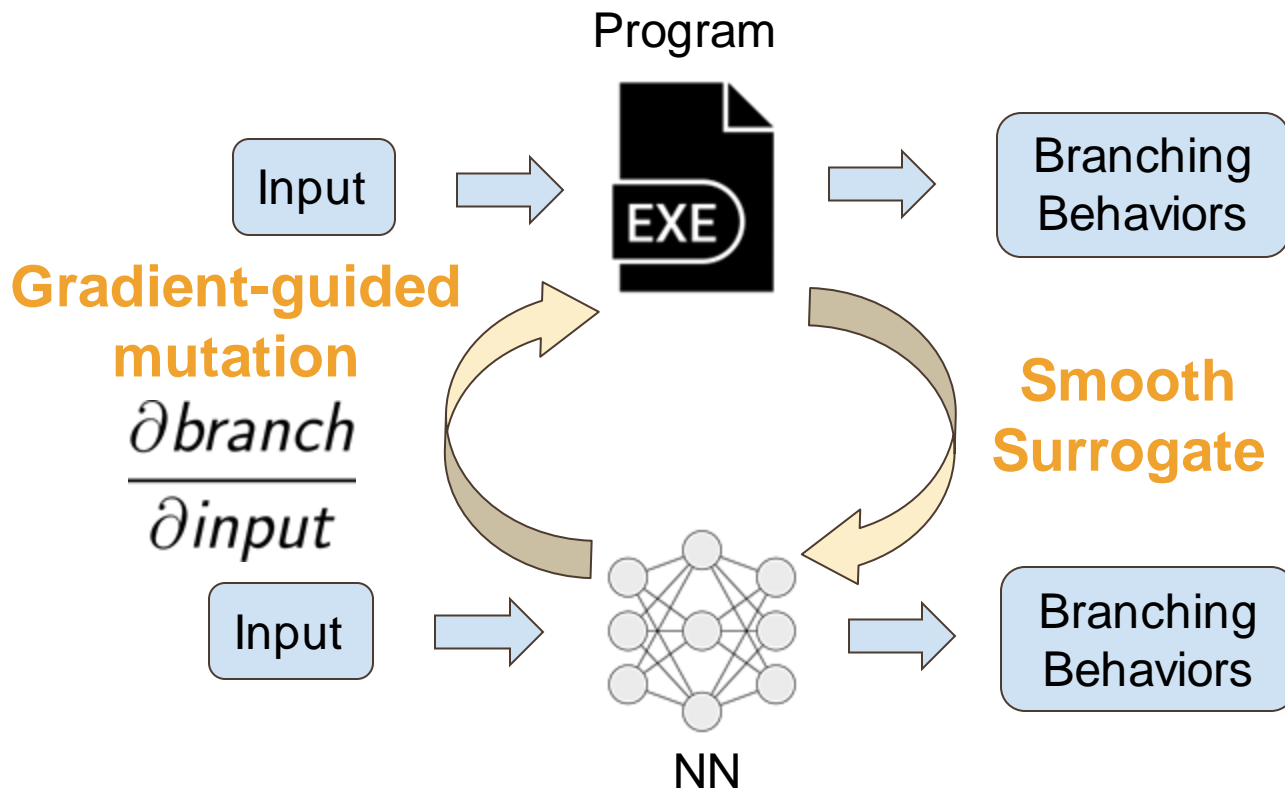
What are critical parts of the input?

Critical parts of input affect program branches

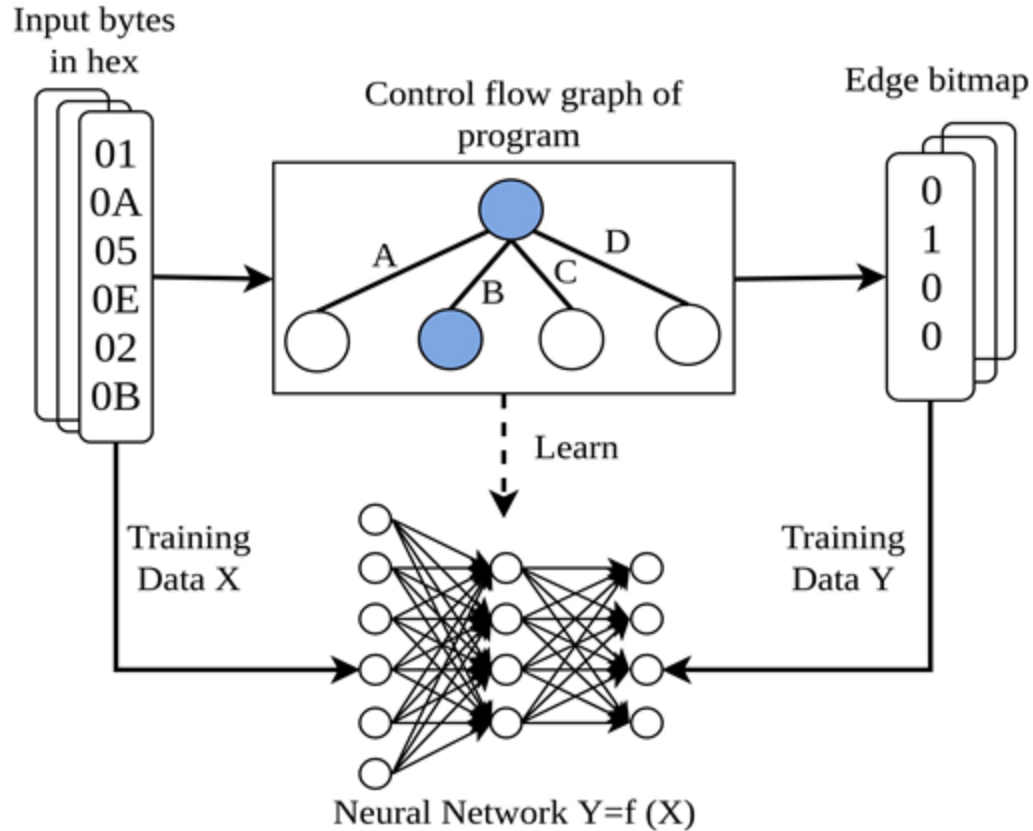
How gradient-guided mutation works?

Focus mutations on the critical parts of the input

Main Idea behind Neuzz



A Peek Into NN Model



Generalization to Unseen branches

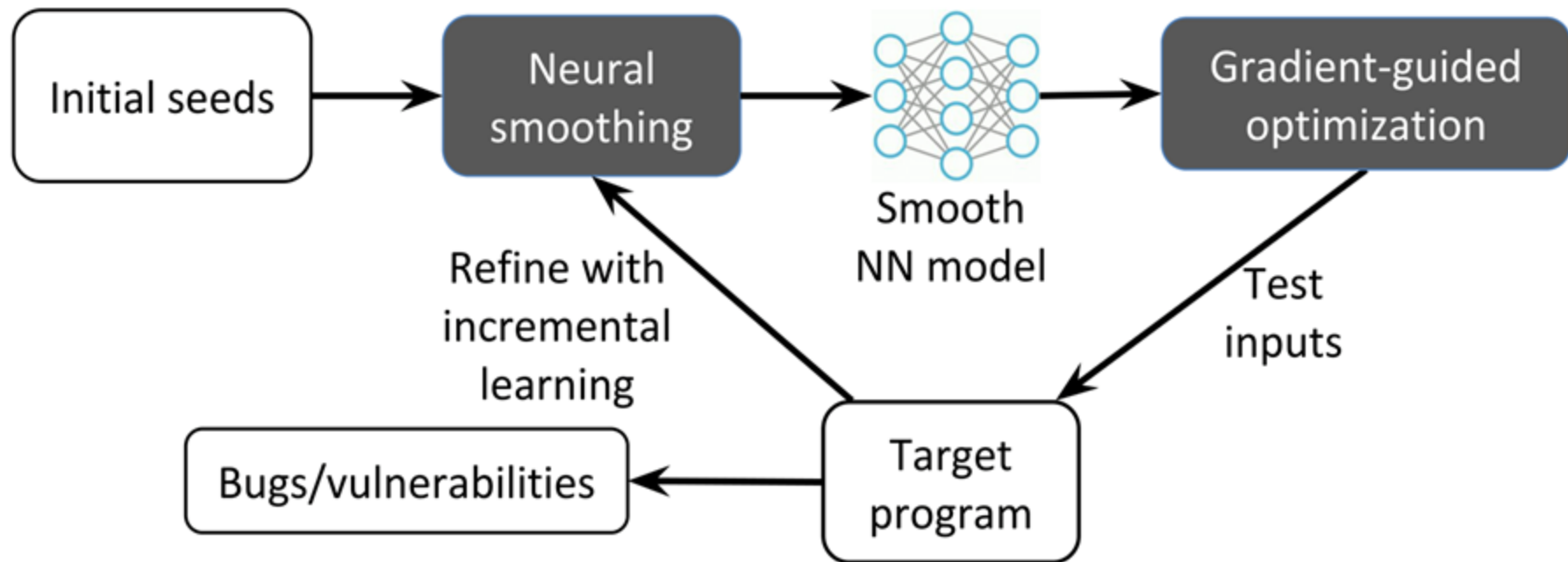
Observations:

- Real world program inputs have critical parts
- Most of branches are affected by the critical parts

Neuzz Solution:

- Identify critical parts based on observed branches
- Perform more mutations on the critical part of inputs to explore unseen branches

Design of NEUZZ



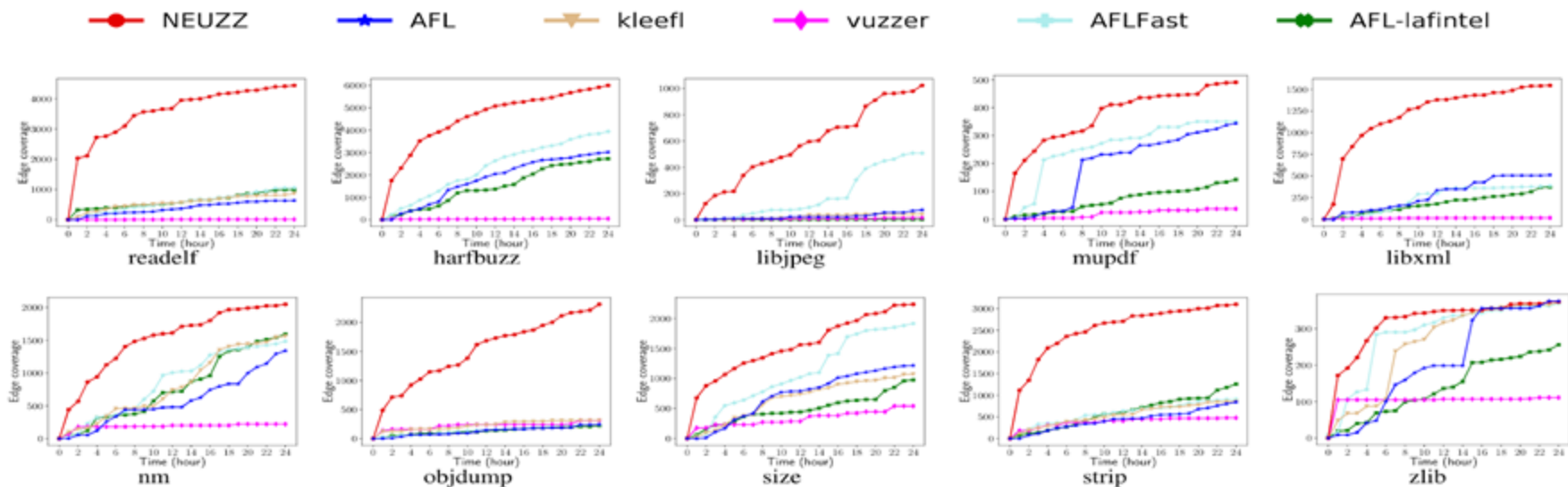
Evaluation

- 10 real world programs
- Lava-M and DARPA CGC datasets
- Comparison with RNN-based fuzzers
- Performance of different model choices

Evaluations: Edge Coverage

NEUZZ vs. state-of-the-art fuzzers

10 real world applications for 24 hours



NEUZZ achieves on average 3x more edge coverage than other fuzzers

Evaluations: Bug Finding

NEUZZ vs. state-of-the-art fuzzers

Programs	AFL	AFLFast	VUzzer	KleeFL	AFL-laf-intel	NEUZZ
Detected Bugs per Project						
readelf	4	5	5	3	4	16
nm	8	7	0	0	6	9
objdump	6	6	0	3	7	8
size	4	4	0	3	2	6
strip	7	5	2	5	7	20
libjpeg	0	0	0	0	0	1
Detected Bugs per Type						
out-of-memory	✓	✓	✗	✓	✓	✓
memory leak	✓	✓	✓	✓	✓	✓
assertion crash	✗	✓	✗	✗	✓	✓
interger overflow	✗	✗	✗	✗	✗	✓
heap overflow	✓	✗	✗	✗	✗	✓
Total	29	27	7	14	26	60

NEUZZ finds the most number of bugs and all 5 bug types including two new CVEs

Evaluations: Lava-M and CGC

Lava-M dataset

	base64	md5sum	uniq	who
#Bugs	44	57	28	2,136
FUZZER	7	2	7	0
SES	9	0	0	18
VUzzer	17	1	27	50
Steelix	43	28	24	194
Angora	48	57	29	1,541
AFL-laf-intel	42	49	24	17
T-fuzz	43	49	26	63
NEUZZ	48	60	29	1,582

DARPA CGC dataset

Fuzzers	AFL	Driller	NEUZZ
Bugs	21	25	31

NEUZZ outperforms state-of-the-art fuzzers on LAVA-M and CGC

Evaluations: NEUZZ vs. RNN-based Fuzzer

Programs	Edge Coverage			Training Time (sec)		
	NEUZZ	RNN	AFL	NEUZZ	RNN	AFL
readelf -a	1,800	215	213	108	2,224	NA
libjpeg	89	21	28	56	1,028	NA
libxml	256	38	19	95	2,642	NA
mupdf	260	70	32	62	848	NA

NEUZZ achieves 6x more edge coverage and 20x less training time

Evaluations: Effect of Different NNs

Edge coverage for 1M mutations

Programs	Linear Model	NN Model	NN + Incremental
readelf -a	1,723	1,800	2,020
libjpeg	63	89	159
libxml	117	256	297
mupdf	93	260	329

NEUZZ achieves best performance with
NN+Incremental learning

Key Takeaways of NEUZZ

- Use NN gradients to identify the critical locations of program inputs
- Focus mutations on the critical locations
- Minimize runtime overhead by using simple feed-forward neural networks
- Retrain the network incrementally to find new critical locations

Github Repo

NEUZZ is available at

<https://github.com/Dongdongshe/neuzz>

NEUZZ: Efficient Fuzzing with Neural Program Smoothing

Dongdong She, Kexin Pei, Dave Epstein, Junfeng Yang, Baishakhi Ray, and Suman Jana
Columbia University