DeepSmartFuzzer: Reward Guided Test Generation For Deep Learning

Samet Demir, *Hasan Ferit Eniser*, Alper Sen

The research was conducted at Boğaziçi University, Istanbul.

AISafety 2020

Overview

- Motivation
- Background
- DeepSmartFuzzer
- Evaluation

Motivation

Deep Neural Networks (DNNs) must be characterized by a high degree of dependability before being deployed in safety-critical systems.

Having a test set that satisfies a coverage criterion provides a degree of dependability to the system under test.

We introduce **DeepSmartFuzzer**, a novel coverage guided fuzzer, which generates new test inputs to achieve high coverage in DNNs.

Our ultimate goal is to help practitioners extend their test sets with new inputs so that new behaviours are covered.

Background

Coverage Guided Fuzzing (CGF):

performing systematic mutations on inputs and produces new test inputs to increase the number of covered cases for a target coverage metric

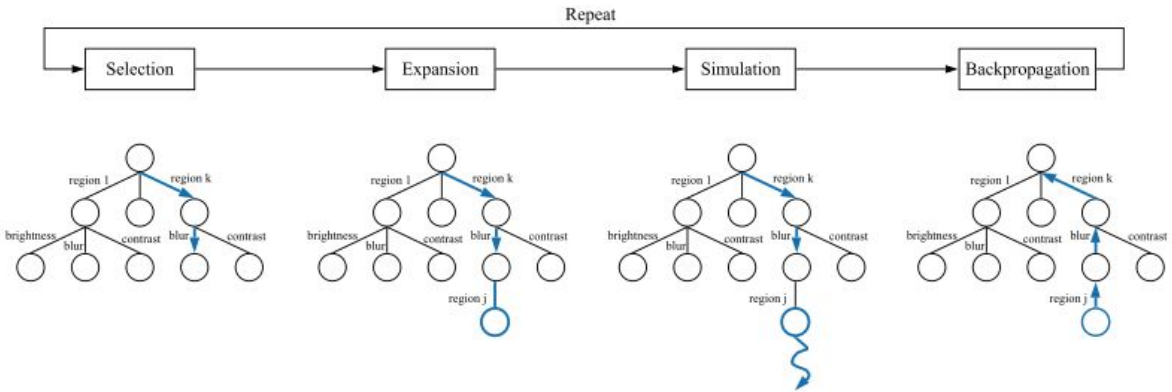
Existing CGFs for DNNs:

- Tensorfuzz [Odena et al., 2018]
- DeepHunter [Xie et al., 2018]

Background

Monte Carlo Tree Search (MCTS) [Chaslot et al., 2008]:

a search algorithm for decision processes such as game play.

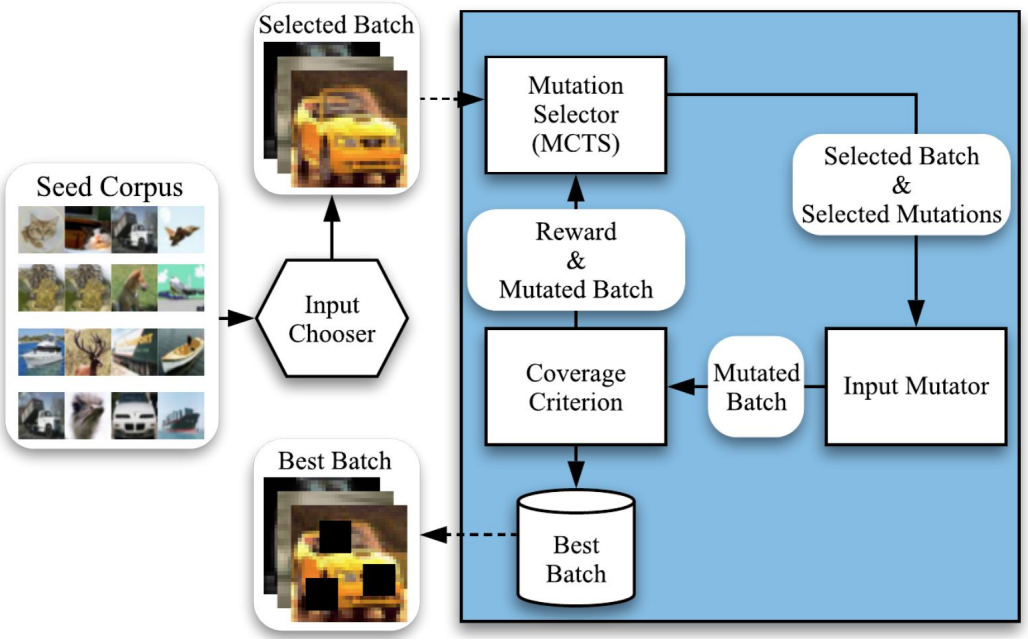


DeepSmartFuzzer Main Idea

The core idea is to make use of **reward-guided** exploration and exploitation (i.e. Reinforcement Learning) to extend the test set by mutating inputs in a smarter way so that to achieve a higher coverage score.

In reward-guided process, selected mutations are evaluated with the coverage changes they induce.

DeepSmartFuzzer Workflow



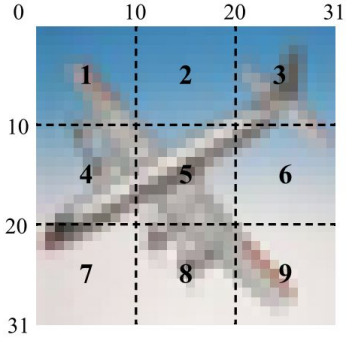
DeepSmartFuzzer Components

1- Input Chooser

Random input chooser: Samples a batch of inputs from the corpus using the uniform distribution.

Clustered random input chooser: First an off-the-shelf clustering method applied on the inputs in the corpus. Then a cluster is chosen randomly to sample a batch of inputs.

DeepSmartFuzzer Components



2- Mutation Selector

Our proposed mutation selector is a **two-player game** such that *Player I* selects the region to be mutated, and *Player II* selects the mutation (blur, contrast change etc.) to be made on the chosen region.

The **reward** (objective) of the game is determined by coverage increase.

The **end of the game** is determined by the distance between original and mutated inputs in order to avoid unrealistic inputs.

We use Monte Carlo Tree Search (MCTS) in order to exploit rewards and guide the search for mutations towards rewarded mutations

DeepSmartFuzzer Components

3- Input Mutator

Applies selected mutations found in previous step.

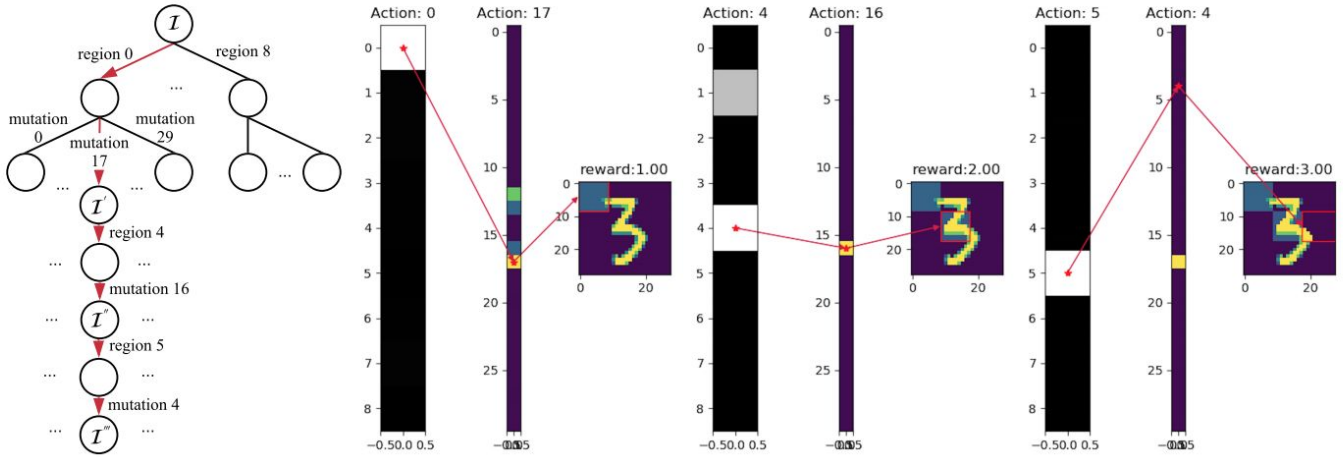
Input mutator induces a bias towards locality since it applies mutations to regions of an image.

Natural fit for image problems and convolutional neural networks.

Algorithm 1 Algorithmic description of DeepSmartFuzzer

```
1:  procedure DeepSmartFuzzer(T, tc_0, tc_1, tc_2, tc_3)
2:      while not tc_0 do
3:          I = Input_Chooser(T)
4:          R = MCTS_Node(I)
5:          best_cov, I^best = 0, I
6:          while not tc_1 do
7:              while not tc_2 do
8:                  L = MCTS_Selection(R)
9:                  C = MCTS_Expansion(L)
10:                 I^(n-1) = get_batch_corresponding_to_node(C)
11:                 r^(n), m^(n) = MCTS_Simulation(C)
12:                 I^(n) = Input_Mutator(I^(n-1), r^(n), m^(n))
13:                 if not tc_3 then
14:                     cov_inc = Coverage(T ∪ I^(n)) - Coverage(T)
15:                     if cov_inc > best_cov then
16:                         best_cov, I^best = cov_inc, I^(n)
17:                     MCTS_Backpropagation(C, cov_inc)
18:             R = select_child(R)
19:         if best_cov > 0 then
20:             T = T ∪ I^best
21:     return T
```

DeepSmartFuzzer at Work



(a) The game tree

(b) The selected mutations on a seed input

Evaluation

We evaluate effectiveness of DeepSmartFuzzer on various existing coverage criteria developed for DNNs:

- Neuron Coverage
- K-Multi Section Neuron Coverage
- Neuron Boundary Coverage
- Strong Neuron Activation Coverage
- TensorFuzz Coverage

We compare our tool against existing CGF tools, DeepHunter[Xie et al., 2018] and TensorFuzz[Odena et al., 2018] with 2 hours time limit and 1000 seed inputs.

We use Lenet models (trained with MNIST) and a medium size CNN (trained with CIFAR10) as benchmarks.

Results

Lenet1 and Lenet4:

Model - Coverage / CGF	MNIST LeNet1 (NC)	MNIST LeNet1 (KMN)	MNIST LeNet1 (NBC)	MNIST LeNet1 (SNAC)	MNIST LeNet1 (TFC)	MNIST LeNet4 (NC)	MNIST LeNet4 (KMN)	MNIST LeNet4 (NBC)	MNIST LeNet4 (SNAC)	MNIST LeNet4 (TFC)
DeepHunter	0	2.34 ± 0.03 %	35.42 ± 2.76 %	41.67 ± 4.17 %	29.00 ± 3.61	0	1.91 ± 0.04 %	**13.15 ± 2.34 %**	16.67 ± 0.81 %	20.00 ± 2.00
TensorFuzz	0	1.83 ± 0.23 %	0	0	0.33 ± 0.58	0	1.26 ± 0.05 %	0	0	0
DeepSmartFuzzer	0	**2.91 ± 0.11 %**	**41.67 ± 4.77 %**	**42.36 ± 6.36 %**	**204.67 ± 8.50**	**1.41 ± 0.00 %**	**2.07 ± 0.14 %**	11.50 ± 1.13 %	**16.90 ± 3.07 %**	**64.33 ± 6.03**
DeepSmartFuzzer(clustered)	0	2.88 ± 0.04 %	38.54 ± 0.00 %	39.58 ± 7.51 %	111.00 ± 14.53	**1.41 ± 0.00 %**	2.02 ± 0.07 %	11.50 ± 0.54 %	15.02 ± 2.15 %	53.33 ± 8.39

Lenet5 and CIFAR10-CNN:

Model - Coverage / CGF	MNIST LeNet5 (NC)	MNIST LeNet5 (KMN)	MNIST LeNet5 (NBC)	MNIST LeNet5 (SNAC)	MNIST LeNet5 (TFC)	CIFAR CNN (NC)	CIFAR CNN (KMN)	CIFAR CNN (NBC)	CIFAR CNN (SNAC)	CIFAR CNN (TFC)
DeepHunter	0.51 ± 0.58 %	1.77 ± 0.03 %	6.23 ± 0.55 %	8.40 ± 0.66 %	19.00 ± 1.73	1.99 ± 0.19 %	0.98 ± 0.03 %	2.39 ± 0.64 %	4.48 ± 0.90 %	16.00 ± 2.65
Tensorfuzz	0.13 ± 0.22 %	0.75 ± 0.06 %	0.13 ± 0.22 %	0	1.33 ± 0.58	0.93 ± 0.15 %	0.13 ± 0.01 %	1.54 ± 0.19 %	2.92 ± 0.34 %	0
DeepSmartFuzzer	2.16 ± 0.44 %	**1.99 ± 0.01 %**	7.82 ± 1.06 %	**9.03 ± 1.10 %**	**76.33 ± 5.69**	**3.51 ± 0.48 %**	**1.38 ± 0.09 %**	2.39 ± 1.23 %	4.91 ± 2.51	42.33 ± 4.51
DeepSmartFuzzer(clustered)	**2.29 ± 0.38 %**	1.92 ± 0.08 %	**7.89 ± 0.72 %**	8.40 ± 1.91 %	76.00 ± 8.89	**3.51 ± 0.37 %**	1.33 ± 0.06 %	**3.83 ± 2.66 %**	**8.80 ± 7.11**	**48.67 ± 7.02**

Results - Statistical Significance

We apply one-tailed **t-test**. The significance threshold is set at **0.05**.

We check two hypotheses:

1) DeepSmartFuzzer achieves higher coverage when compared to Tensorfuzz
2) DeepSmartFuzzer achieves higher coverage when compared to DeepHunter

We obtain P-Values of <0.001 and 0.007 respectively. (*Hypotheses accepted*)

Overall, we conclude that DeepSmartFuzzer provides a significant improvement over existing coverage-guided fuzzers for DNNs.