A summary of 'DLFuzz: Differential Fuzzing Testing of Deep Learning Systems'

Prepared by-Sumaiya Saima Sultana

Theme of the paper

- DLFuzz is a testing framework for Deep Learning systems. It uses differential fuzzing method to expose the corner cases for the system.
  - **Fuzzing** or **fuzz testing** is a testing technique that involves providing invalid, unexpected, or error-inducing data as inputs.
- Inspired from the state-of-the-art DL whitebox testing framework- DeepXplore, DLFuzz also maximizes neuron coverage while mutating inputs to generate corner cases for the DL system.

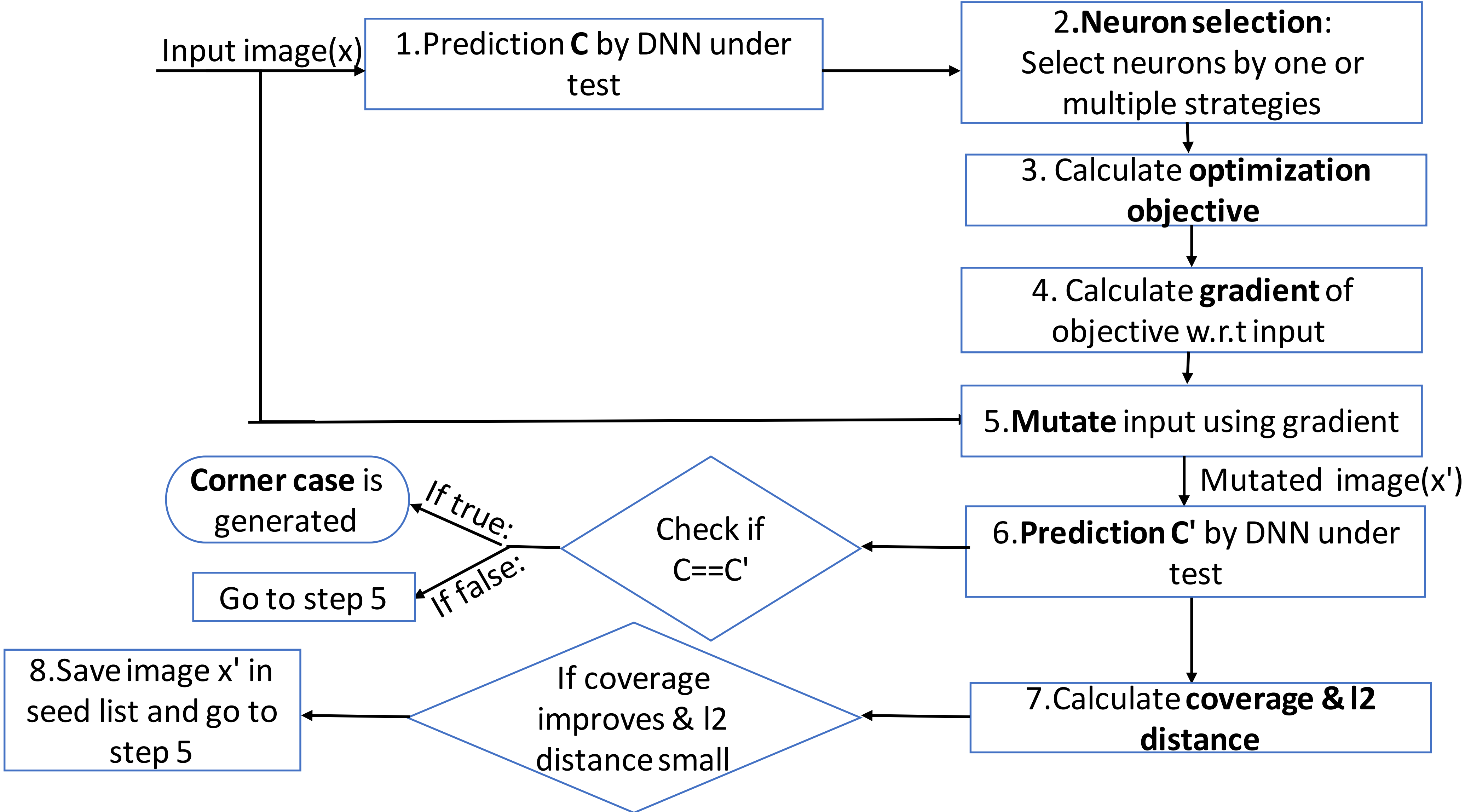
Similarities with DeepXplore

- Like DeepXplore, DLFuzz also solves a joint optimization problem using gradient ascent.
  - The joint optimization problem includes maximizing neuron coverage while maximizing the prediction error.
  - Since maximizing the prediction error is opposite to optimizing weights to minimize prediction error while training, so the loss function is customized as objective function and maximized by gradient ascent.
- DLFuzz follows the definition and computing way of Neuron Coverage as suggested by DeepXplore.
  - A threshold is set to check if a neuron is activated or deactivated.

Differences with DeepXplore

- ➢DLFuzz does not use several DL systems with same functionality for cross-referencing label check.
- ➢DLFuzz mutates input in such a way that the generated corner cases are visibly indistinguishable from the original images.
  - While mutating the inputs, mutation is kept restricted to invisible changes by using l2 distance as a metric. The l2 distatnce between the original image and the mutated image is kept within a limit.
- ➢DLFuzz uses 4 heuristic strategies for selecting neurons to improve coverages.
  - The strategies give prioroties to neurons that are-
    1. Covered frequently
    2. Covered rarely
    3. Have top weights
    4. Have values near activation threshold

Overall Workflow



Input image(x)

1.Prediction **C** by DNN under test

2.**Neuron selection**: Select neurons by one or multiple strategies

3. Calculate **optimization objective**

4. Calculate **gradient** of objective w.r.t input

5.**Mutate** input using gradient

Mutated image(x')

6.**Prediction C'** by DNN under test

Check if C==C'

If true: **Corner case** is generated

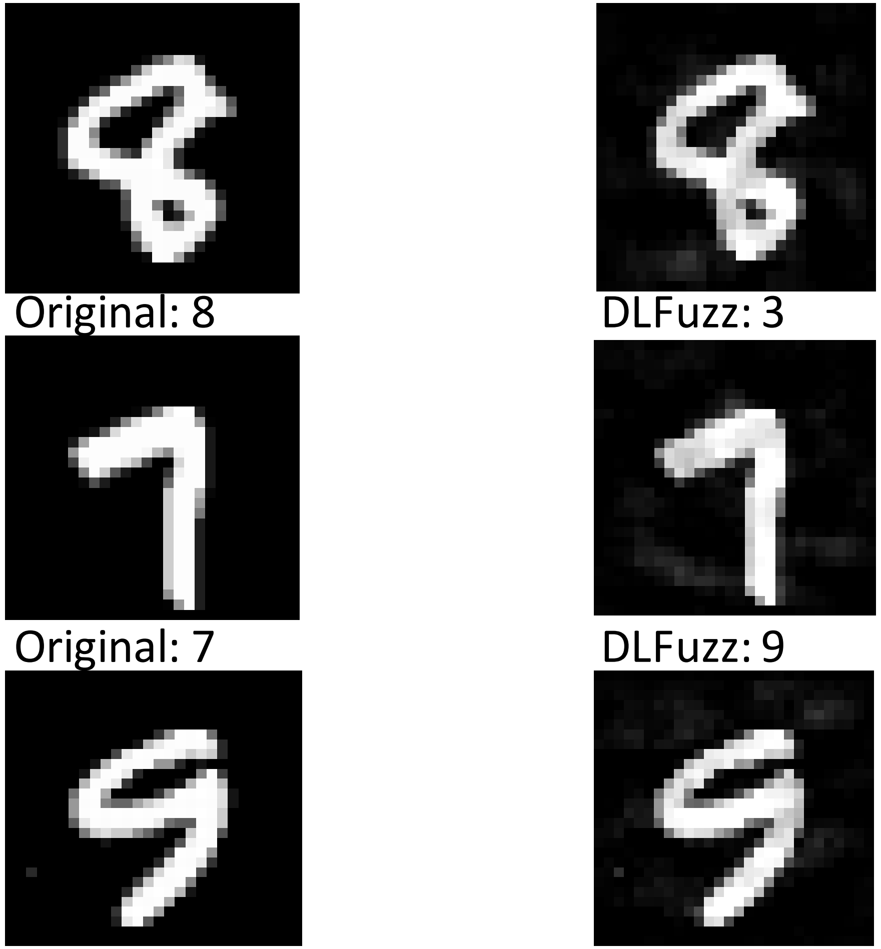
If false: Go to step 5

7.Calculate **coverage & l2 distance**

If coverage improves & l2 distance small

8.Save image x' in seed list and go to step 5

Result of DLfuzz: (Figure 3 of paper)



Original: 8

DLFuzz: 3

Original: 7

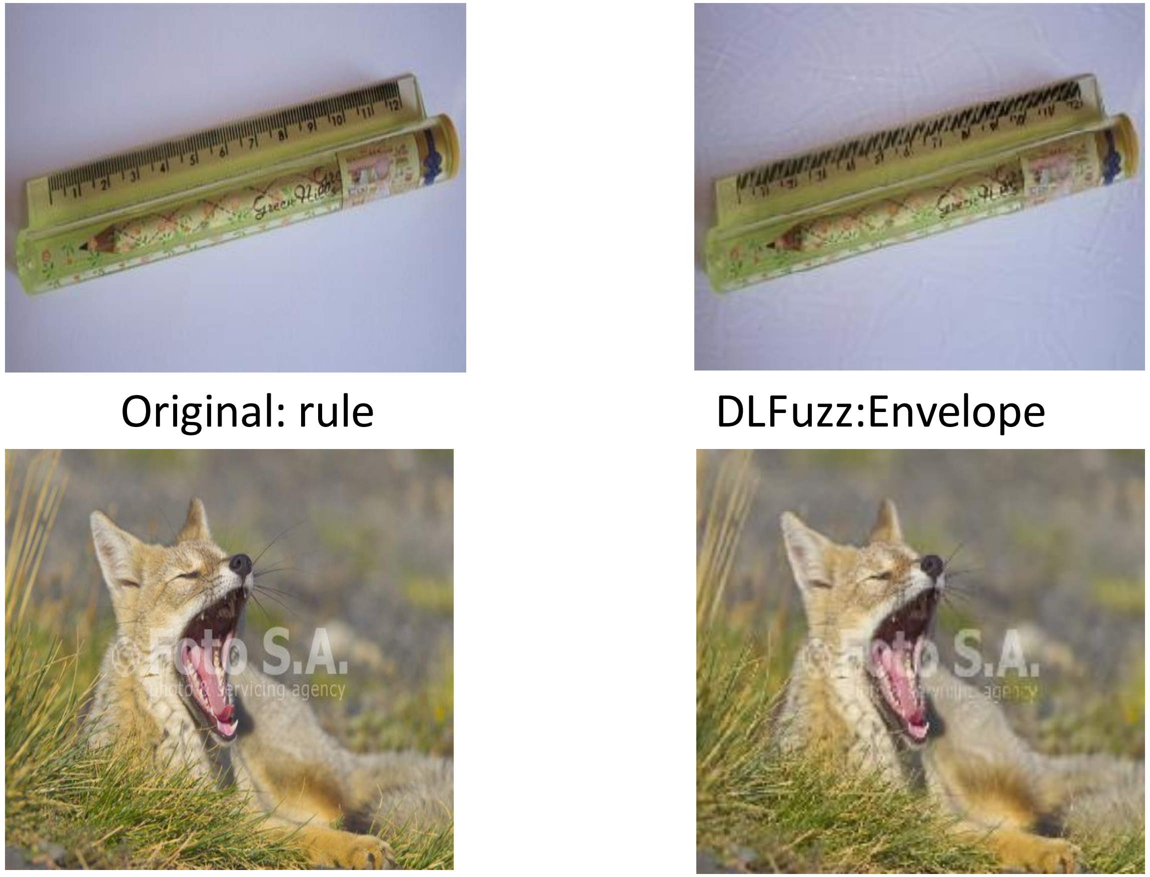
DLFuzz: 9

Original: 9

DLFuzz: 5

Figure: Cases of adversarial inputes for MNIST dataset for model1(LeNet1), model2(LeNet4) and model3(LeNet-5) repectively

Result of DLfuzz: (Figure 3 of paper)



Original: rule

DLFuzz:Envelope

Original: coyote

DLFuzz: red_fox

Figure: Cases of adversarial inputs for IMAGENET dataset (model-VGG16)

Result of DLfuzz: (Figure 4 of paper)

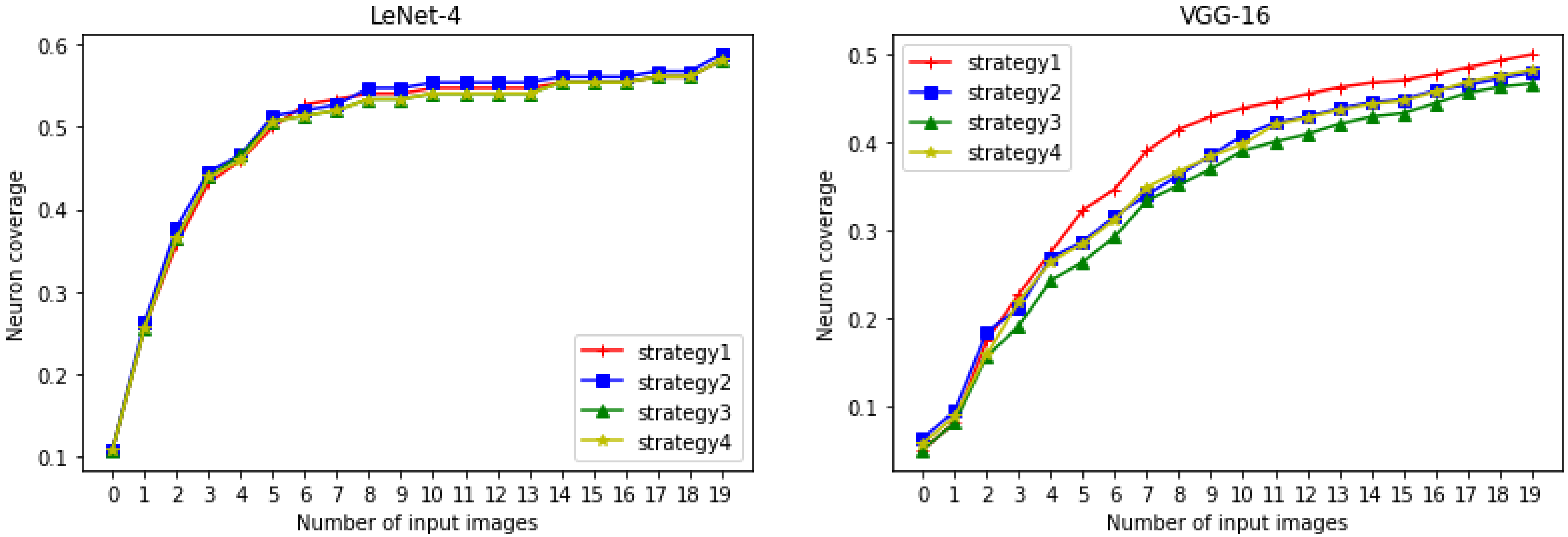


Figure: Neuron Coverage with number of images tested when different strategies applied in DLFuzz

Thank you!