

2021 Spring CS453

Adversarial Image Generation based on Various Neuron Coverage

Team 8

20170181 Taeyoung Kim

20180650 Hyunjoon Cho

20205424 Sunjae Kwon

Abstract

With increasing demand for AI-based software, validation of a deep neural network is getting more important. One common approach is to find error-inducing inputs and retrain the model with them. Researchers worked on automated adversarial image generation and DeepXplore achieved this goal on top of the original neuron coverage concept. DLFuzz applied the fuzzing technique in combination with the concept and several researchers proposed variants of neuron coverage accordingly.

DeepXplore and DLFuzz only consider the basic neuron coverage. However, as we can see from traditional software testing, improved coverage concepts resulted in more reliable programs.

Thus, we attempted to improve these frameworks by applying newer neuron coverage concepts and to find which one works the best. We picked k-multisection neuron coverage with low expectation and strong neuron activation coverage with high expectation and compared it against the neuron coverage.

Our modified frameworks successfully generated adversarial images and each coverage created different outputs. However, it was hard to rank them by their performance since they did not show clear distinction. We concluded that further research on another neuron coverage concept or aggregation of multiple coverages is needed.

Introduction

Deep Neural Network(DNN) has been actively studied and widely applied to various fields. And many of them are safety-critical applications such as an autonomous driving system, health care equipment and UAV. In a safety-critical domain, thorough validation of the system is necessary.

For example, if a DNN decides to turn to the right in the picture below, the car will crash into guardrails and passengers would suffer from the injury or they can even lose their lives. In other words, misbehavior of a model poses great threat to the user.



Figure 1. Example Misbehavior of an Autonomous Driving System [1]

Therefore, the demand for testing and validating a DNN system is increasing. However, unlike traditional software, DNN is a data-driven system, i.e. the only way to avoid this erroneous decision is to find rare inputs that generate abnormal behavior and re-train the system using these rare inputs.

Manual approach to find these inputs does exist but is not feasible due to heavy calculations. Thus, various automated testing tools have been proposed and the neuron coverage concept emerged accordingly. It is borrowed from a coverage concept in traditional software testing. As shown below, the left image is a traditional program. If a test case covers more statements and branches, it is more probable to reach the buggy code. It is similar to DNNs. Each node indicates one rule for classification. If an input image has higher neuron coverage, in other words, the more the neurons are activated, the more probable to induce erroneous behavior. As a result, many frameworks tried to produce an image with higher neuron coverage.

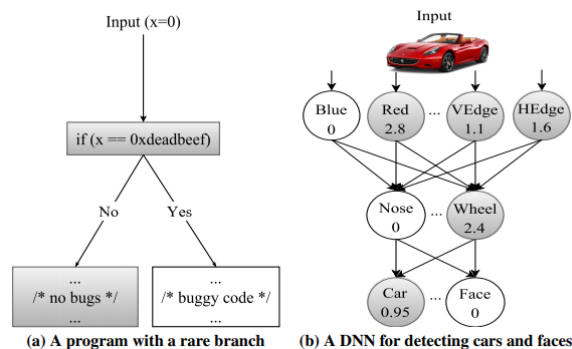


Figure 2. Comparison of traditional coverage and neuron coverage [1]

Related Work

Our project is built on top of two frameworks. First one is DeepXplore. This paper introduced the neuron coverage concept first and successfully generated adversarial images. As shown below, it uses multiple DNN models that do the same work but have different structures. For a seed that all models giving identical output, it does gradient ascent till the target model gives out a different label. This gradient ascent is to induce different behavior between models and maximize neuron coverage.

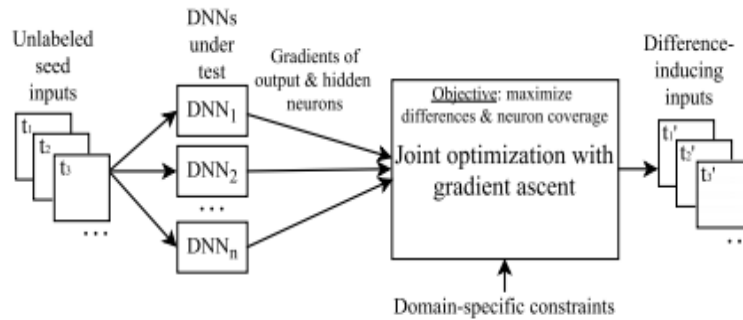


Figure 3. DeepXplore Workflow [1]

The other framework is DLFuzz. It is inspired by Deepxplore and works based on the neuron coverage concept. But the difference is that it only requires one model and applied fuzzing technique to generate adversarial images.

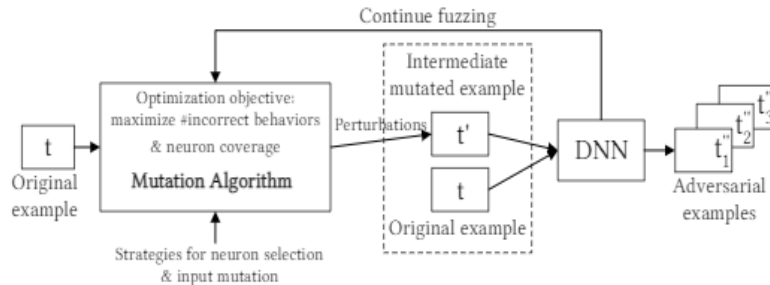


Figure 4. DLFuzz Workflow [2]

Following the initial neuron coverage, multiple modified coverage concepts were proposed.

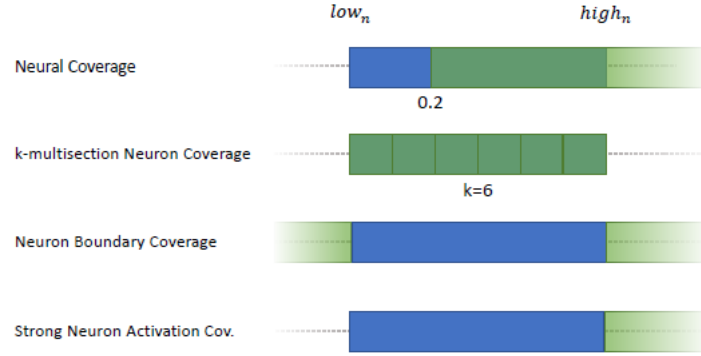


Figure 5. Visualization of neuron coverages [3]

Neuron Coverage

Given threshold ϵ , if the neuron's activation value is larger than ϵ for a test input, it is covered.

k-multisection Neuron Coverage

Let h, l be the highest neuron activation value from the training data set (lowest, respectively). Range $[l, h]$ is divided into k sections. Coverage is measured as the percentage of sections that at least one activation value was observed.

Neuron Boundary Coverage

Let h, l be the highest neuron activation value from the training data set (lowest, respectively). A neuron is covered if both the upper corner region $[h, \infty]$ and lower corner region $[-\infty, l]$ are covered. This concept corresponds to boundary coverage in software testing.

Strong Neuron Activation Coverage

Let h be the highest neuron activation value from the training data set. A neuron is covered if the upper corner region $[h, \infty]$ is covered. It is similar to the neuron boundary coverage.

Project Goal

Two existing frameworks we introduced, DeepXplore and DLFuzz, solely depend on basic neuron coverage. In the meantime, various neuron coverage metrics have been proposed. We wondered what if we plug in other coverages instead of the basic one. With rough expectation that k-multisection neuron coverage is less probable to show new behavior in that each section is inside already-seen range and strong neuron activation coverage is more probable because higher activation tends to propagate to the next layer, we decided to compare three coverages - neuron coverage, k-multisection and strong and find the best one in terms of adversarial image generation.

Method

To find the best coverage, we implemented modified frameworks. Thankfully, authors of both papers opened their source code on github.

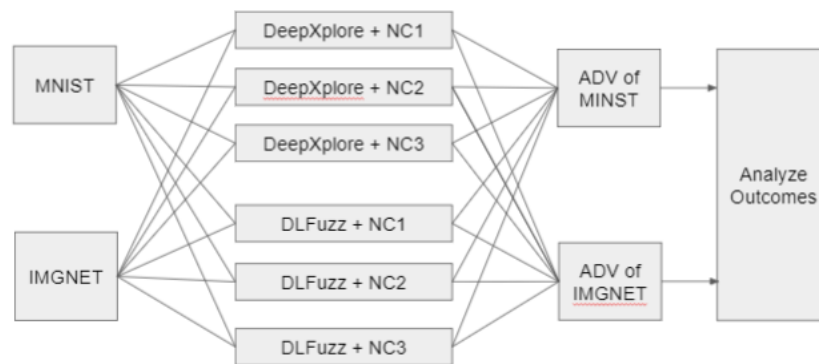


Figure 6. Project flow chart

We modified the code so that it first stores the min-max output of each neuron. Since the frameworks use pre-trained models, we distinguished seeds into training set and generation set and ran predictions on the training set to get min-max value. Then, modified coverage code to track k-multisection coverage and strong coverage. Finally, iterate over the generation set and generate adversarial images. We ran the program on each coverage and analyzed the outcomes.

Experimental Set-ups

We runned DLFuzz and DeepXplore on two computer vision dataset, MNIST and ImageNet. MNIST is a 28×28 grayscale handwritten digit image dataset with ten categories, and ImageNet is an arbitrary size object image with RGB channel images with 1,000 categories. For here, we use 224×224 sized sub-sampled images for ImageNet.

We used three models for DeepXplore. For MNIST, we used LeNet-1, LeNet-4, LeNet-5, and for ImageNet, we used VGG16, VGG19, ResNet50. We used pretrained models for all six models. All three LeNet's parameters are fetched from DeepXplore source code's data, and VGG16, VGG19, ResNet50's parameters are fetched from tensorflow keras applications package's pretrained parameters. For DLFuzz, we used LeNet-1 for MNIST dataset and VGG16, VGG19, ResNet50 for ImageNet. All models used the same parameters with DeepXplore.

For DeepXplore, we runned experiment with all three constraints for perturbation, occlusion, blackout, and light. However, for the ImageNet problem, the blackout constraint didn't work. The condition of blackout usually wasn't satisfied, which requires that the mean of gradient is negative, however we couldn't figure out why this problem only happens in the ImageNet problem.

Boundary neuron coverage and k-multisection coverage requires min-max value of activations during training. For MNIST, we were able to record min-max values using existing train dataset. However, for the ImageNet dataset, we couldn't download the train dataset, and even if we could, recording all min-max values for 14M images was intractable. So instead, we splitted seeds in half, and used one half as an approximation for the train dataset and the rest half as seeds.

For the hyperparameters, we used the following values. Weights, step size, number of seeds, and number of iterations are chosen by following the original paper's hyperparameters.

- Loss coefficients
 - Weight diff : Weight for the difference loss. Force the difference of each model's output. 1.0.
 - Weight NC : Weight for the neuron coverage. Force the neuron coverage to be increased. 0.1.

- Gradient Ascent parameters
 - Step : Learning rate of image perturbation. 10.0.
 - Grad Iteration : Number of the gradient ascent steps. 20.
 - Seeds : Number of test images used as starting point of adversarial image. 100.
- Neuron Coverage parameters
 - Threshold : Threshold for classical neuron coverage. 0.2.
 - k : Number of multisection for k-multisection coverage. 5.

For the evaluation, we measured the following four metrics.

1. Number of adversarial images generated
2. Average time taken to generate a single adversarial image
3. Coverage after adversarial image generation
4. Average L2 distance between the original image and adversarial image

Experimental Results

1. DeepXplore on MNIST

Figure 7 shows the result images of DeepXplore to MNIST dataset, for each constraint and neuron coverage. Each three digits under images are labels that three models predicted. As you can see, after adversarial generation, all constraints and coverages were able to confuse each model.













	Original	Light	Occlusion	Blackout
NC1 (0.2 threshold)	 (4,4,4)	 (8,4,4)	 (7,7,2)	 (2,7,4)
NC2 (5-multisection)	 (6,6,6)	 (8,6,4)	 (7,2,2)	 (6,5,6)
NC3 (Strong)	 (9,9,9)	 (3,9,8)	 (7,2,2)	 (4,9,4)

Figure 7. Result images of DeepXplore on MNIST dataset

Table 1 shows the measurement of comparison metrics for each coverage. Each sub column corresponds to the type of constraint, blackout, light, occlusion in order.

Since DeepXplore aims to generate adversarial images that three models disagree with, it does not mutate images that three models already disagree with. The number # Initial does not include such images. The time per Adv is an average time taken for adversarial image generation, which is the total elapsed time divided by number of generated adversarial images. Each coverage is measured with corresponding coverage.

As the table shows, all three coverages gave similar results. Rather we were only able to observe that light constraint was not suitable, since it results in too high L2 distance and low success rate. Also, since k-multisection neuron coverage requires all sections to be covered, it resulted in low coverage in all three constraints.

	# Adv / # Initial			time per Adv (s)			Coverage			Avg L2		
NC1 (Neuron)	97/97	86/95	81/99	4.67	5.13	5.27	0.47	0.47	0.42	376	2586	492
NC2 (k-multi)	97/97	80/97	74/99	4.96	5.09	5.24	0.31	0.28	0.28	386	2708	487
NC3 (Strong)	97/97	76/97	84/96	4.70	4.64	4.73	0.48	0.27	0.44	381	2712	483

Table 1. Result metrics of DeepXplore on MNIST dataset

2. DeepXplore on ImageNet

Figure 8 shows the result images of DeepXplore on ImageNet dataset. As mentioned above, constraint blackout does not work for ImageNet dataset, so resulting images only contain light and occlusion. All neuron coverages and constraints were able to generate adversarial images.

- DeepXplore on ImageNet

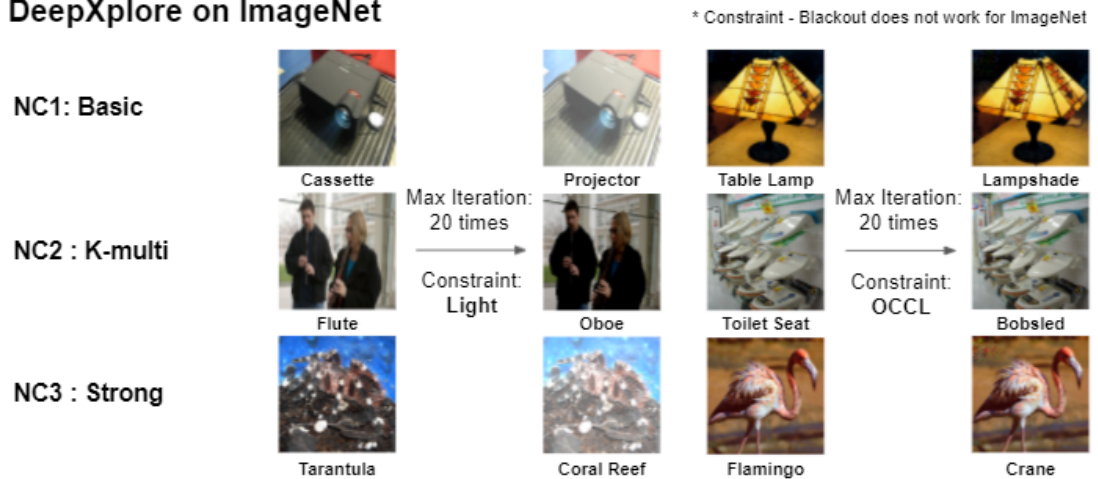


Figure 8. Result images of DeepXplore on ImageNet dataset

Similar to MNIST dataset, Table 2 shows the result of four comparison metrics. Unlike MNIST where all neuron coverages have shown similar results, here multisection coverage has shown poor results for light constraints. Also in the MNIST dataset, we observed that k-multisection neuron coverage had low coverage. However in the ImageNet dataset, strong boundary neuron coverage resulted in low coverage.

	# Adv / # Identical		time per Adv		Coverage		Avg L2	
NC1(Basic)	45/54	45/59	23.32	27.88	0.6995	0.6812	48139.8	13805.6
NC2(k -multi)	22/61	45/67	60.85	36.20	0.6411	0.6729	48299.1	13633.3
NC3(S trong)	31/62	51/68	35.70	25.89	0.2478	0.0534	42723.5	13814.5

Table 2. Result metrics of DeepXplore on ImageNet dataset

3. DLFuzz on MNIST

Figure 9 shows the result images of DLFuzz on MNIST dataset. The leftmost image is the seed image and the target DNN which is LeNet predicts this image to 9. The generated adversarial images through 3 different neuron coverage concepts are the right part of the original image. As shown in this figure, the target DNN predicts every generated adversarial image into 7. However the shape of the generated images are different from each other, which means that the different types of neuron coverage lead the perturbation process into different directions.

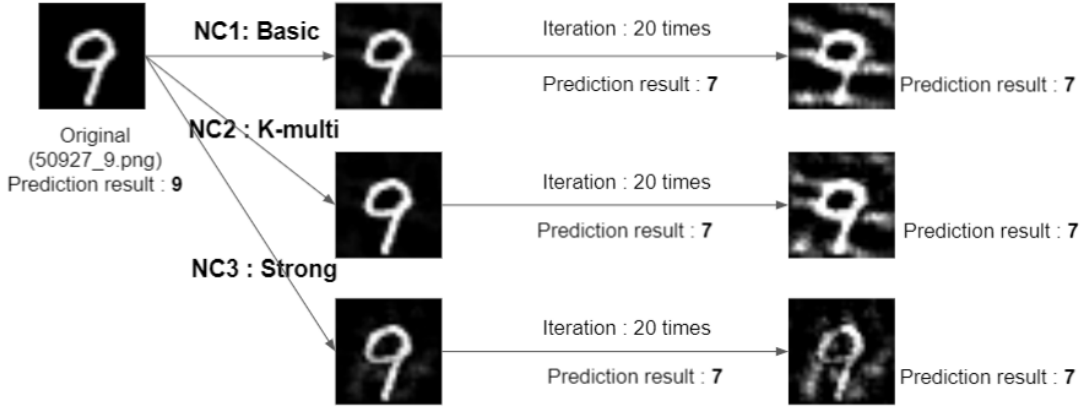


Figure 9. Result images of DLFuzz on MNIST dataset

Table 3 is the summarization of the results. As shown in the table, no particular neuron coverage concept performed well in DLFuzz. However, in the case of the number of seed images where the target DNN can generate adversarial images, NC3(i.e., Strong coverage) is the best neuron coverage concept among others. We assume that the NC3 enables a more active gradient ascent process, so that DLFuzz can find more new adversarial images.

Figure 10 shows the distribution of L2 distance between an adversarial image and an original image. As shown in the graph, there is no specific difference between them.

	# Adv	time per Adv	Coverage	# Seed	L2 distance
NC1 (Basic)	557	6.18	0.67	37	4.74
NC2 (K-Multi)	632	6.09	0.92	42	5.01
NC3 (Strong)	646	6.12	0.48	44	5

Table 3. Result images of DLFuzz on MNIST dataset

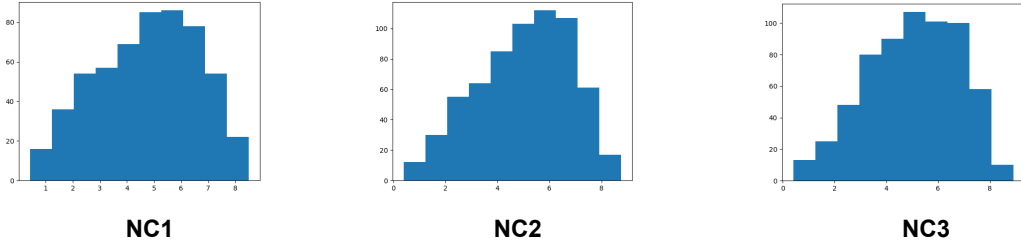


Figure 10. L2 distance distribution of # adversarial images

4. DLFuzz on ImageNet

Figure 11 shows the resulting images of DLFuzz on ImageNet dataset. First of all, unlike MNIST where an image has only black and white, we cannot distinguish which one is the original image and which one is the adversarial image visually. It indicates that the adversarial images generated using neuron coverage concepts are so sophisticated.

Second, unlike MNIST, there is no common seed image where 3 different neuron coverage concepts generate an adversarial image. Additionally, NC2 and NC3 generate adversarial images classified by different labels on the same seed image. So, we assume that different neuron coverage concepts enable the gradient ascent process to move to different directions.

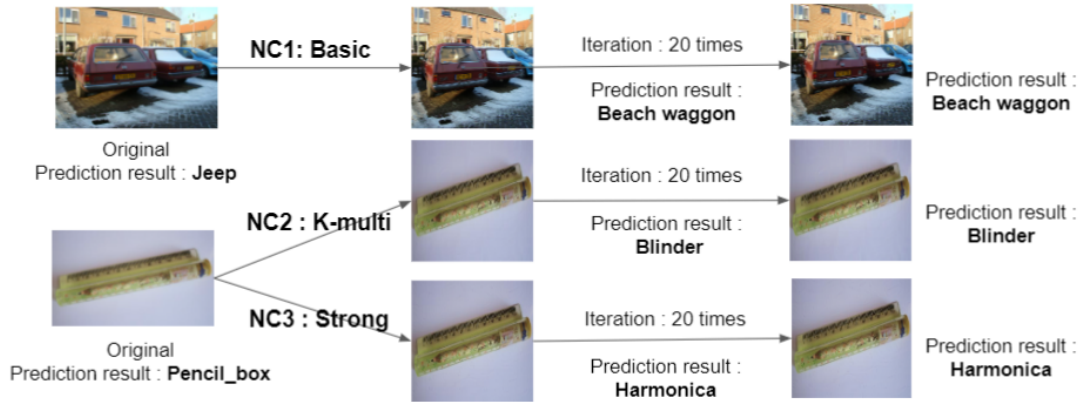


Figure 11. Result images of DLFuzz on IMGNet dataset

Table 4 is the summarization of the results. As shown in the table, NC1(i.g., basic neuron coverage) generates more adversarial images than others. However, in the case of the number of seed images where the target DNN can generate adversarial images, NC3(i.g., strong neuron activation coverage concept) enable to find more new adversarial images. Figure 12 shows the distribution of L2 distance between an adversarial image and an original image. As shown in the graph, NC3 generates adversarial images that are closer to the original input.

	# Adv	time per a Adv	Coverage	# Seed	L2 distance
NC1 (Basic)	178	213	0.496	4	327.66
NC2 (K-Multi)	36	317	0.369	3	389.7
NC3 (Strong)	114	199	0.002	5	326.3

Table 4. Result images of DLFuzz on MNIST dataset

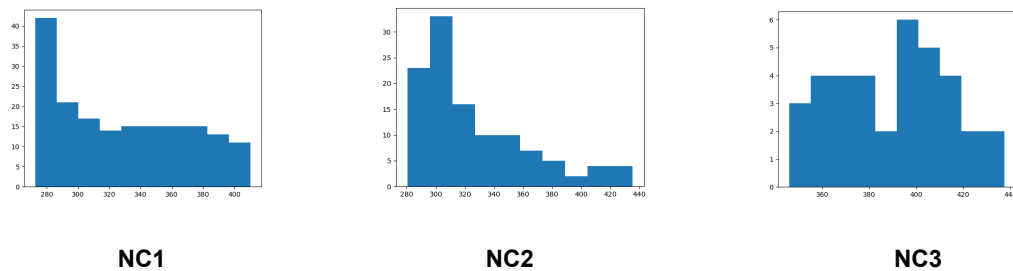


Figure 12. L2 distance distribution of # adversarial images

Analysis and Discussions

As for qualitative analysis, 1) 3 different neuron coverage concepts enable each framework to find different adversarial images. For example, the generated adversarial images have different shapes on the MNIST dataset. Additionally, in the case of ImageNet, adversarial images generated from the same original seed image can be classified into different labels. 2) We confirmed that constraints of generated adversarial images(e.g, light, blackout and occlusion) can influence the performance of the neuron coverage.

As for quantitative analysis, we analyzed performance of each neuron coverage concept on 4 different evaluation metrics (number of generated adversarial images, elapsed time per an adversarial image, percentage of covered neurons, and average L2 distance between an adversarial image and an original image). However, We did not see any notable performance differences on them. Thus, we cannot define the best neuron coverage among them on the two frameworks.

In conclusion, further study is required about a new neuron coverage concept that goes beyond others or integration of three different neuron coverage concepts.

Conclusions

We compared the performance of 3 different neuron coverage concepts on 2 different DNN testing tools with 4 different performance metrics to define the best neuron coverage concept among them. However, we could not confirm any notable performance differences between them on 4 metrics, so it was not possible to define the best neuron coverage. On the other hand, we could confirm that constraints on generated images and type of neuron coverage enable each framework to generate different adversarial images. Each coverage concept and constraints on the images can enable generating adversarial image processes to go different directions and promote performance of them. Thus, we conclude that further study is required about new boundary concepts and integration of various neuron coverages.

Future works

We could find two improvable points about the frameworks. First, since both frameworks only consider increasing output value on loss function, we think that loss function can be redefined to decrease its output value to hit a lower boundary, and this new type of loss function could presume to affect the entire process of the frameworks. Second, these frameworks have several parameters, so we think that we can add a parameter optimization process based on different neuron coverage concepts.

Also, we need to consider new performance analysis metrics that present diversity and patterns of adversarial images that each neuron coverage concept generates in order to find specific characteristics of each neuron coverage concept.

Reference

- [1] Pei, Kexin, et al. "Deepxplore: Automated whitebox testing of deep learning systems." proceedings of the 26th Symposium on Operating Systems Principles. 2017.
- [2] Guo, Jianmin, et al. "DLFuzz: differential fuzzing testing of deep learning systems." Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2018.
- [3] Nils Wenzler. "Not all Neurons are created equal: Towards a feature level Deep Neural Network Test Coverage Metric" In Proceedings of . ACM, New York, NY, USA, 8 pages. 2019.