Daniel Weng
998802130

**ECE1784H Final Paper**

ECE1784H
December 7, 2019

# BackFuzz: An Automated Backdoor Detection Tool

## 1 Introduction

Deep Learning (DL) models are now commonly used to perform tasks like object recognition, signal processing, and autonomous driving. With increasing complexity of problems and available data, it makes sense to train larger and deeper DL models. However, this also increases the amount of computing resources required to train these models, which can take weeks using dozens of GPUs. This has led to the use of cloud computing services and pre-trained models like Google's BERT. This introduces potential for adversaries to insert a backdoor which will cause performance degradation only on specific inputs. A recent experiment shows a backdoored model misclassifying a stop sign as a flower [2]. If this model was used in an autonomous vehicle, the consequences could be severe.

The motivation for the problem is explained well in BadNets by Gu et al., which explores vulnerabilities in the machine learning supply chain that can lead to backdoored models [2]. Specifically, images in the training data were poisoned by inserting pixels in specific locations, i.e the backdoor. The resulting model performed well on standard inputs, but experienced a sharp drop in accuracy when presented with images with the backdoor present. In the paper, the authors show that neural activations on poisoned training data are distinct from those on clean test data. These results are intended to motivate future work in detecting backdoors in DL models. And the authors note that they expect this to be difficult challenge because it is hard to explain the behaviors of neural networks, but that it may be possible to inspect inactivated parts of the network to understand their behavior.

If an automated, cost-efficient backdoor detection technique can be found, then it could be used to greatly improve the backdoor defenses of commercial machine learning supply chains. For example, by scanning pre-trained models before deciding to use them.

Automated whitebox testing solutions for deep learning systems such as DeepXplore and more recently, DLFuzz, have found some initial success in generating adversarial inputs. However, since they were only designed for adversarial input generation, it is unclear if they can be re-purposed to generate backdoor inputs. Nevertheless, there is potential for this technique to be used for this task since adversarial attacks rely on applying a small perturbation on the input and backdoor attacks rely on training the model to respond to a targeted perturbation in the input. In principle, a fuzzing algorithm which modifies the input to optimize neuron coverage and image misclassification should eventually be able to generate an input similar enough to the backdoor that it triggers the mechanism, resulting in a misclassification. In that case, both the backdoor trigger and the backdoored neurons can be discovered. The goal of this paper is to investigate if these tools can be re-purposed for this task.

DLFuzz builds on DeepXplore by generating adversarial inputs with less perturbation and without cross-referencing multiple oracles for differential testing approach. And furthermore, DLFuzz applies fuzzing strategies from the computer security domain in order to generate better inputs. For all those reasons, DLFuzz is a better candidate tool for our goal.

Our approach is to apply DLFuzz on a backdoored model to see if it can successfully generate inputs that trigger the backdoor. To get a performance baseline, DLFuzz was run on the model trained on clean data which generated 129 inputs with a 0% backdoor activation rate. Once the baseline was established, the first experiment was to use DLFuzz, without modification and with the optimal parameters suggested by the original paper, on the backdoored model. The experiment generated 214 inputs with 54% of the generated inputs causing the model to misclassify the input as the next digit, which was the target of the backdoor. We will refer to this metric as the *backdoor activation rate*. This shows that the tool can generate backdoored inputs reasonably well. To further improve this result, we made two further modifications. The best result was 19 inputs with a 79% backdoor activation rate, which was achieved by constraining the perturbations to a 5px by 5px region.

Our main contributions are:

- We introduce the idea of extending whitebox testing tools such as DeepXplore or DLFuzz beyond their original purpose, generating adversarial inputs, to find backdoor-activating inputs instead. And introduce backdoor activation rate to measure effectiveness in generating inputs that can successfully cause the image to be misclassified as intended by the backdoor.

- We theorize that in order to more efficiently generate inputs with a higher backdoor activation rate, we can modify DLFuzz to focus on activating neurons that are activated more during poisoned data than on clean data. And furthermore, that constraining the perturbations generated by DLFuzz to a small region makes the tool more effective at targeting the backdoor.

- We implement these modifications in DLFuzz to create BackFuzz, a tool designed to find backdoors. This tool achieved a 79% backdoor activation rate on a backdoored MNIST CNN model.

## 2 Background and Related Works

### 2.1 Background

The paper on BadNets, by Gu et al. describes an attack where a deep learning model can be backdoored; performing well on training and validation data but poorly on specific attacker-chosen inputs. The paper notes that if there was a way to detect rarely-activated neurons during testing, that would be an indication of a backdoor since the neural network seems to be trained to detect something it would not usually encounter, such as the backdoored inputs [2]. Another paper on DeepXplore, an automated whitebox testing tool, describes an algorithm that does differential analysis on deep learning models to find incorrect behavior [1]. Because backdoored models would demonstrate incorrect behaviours compared to their peers, DeepXplore could potentially be used as a security tool to detect when a model has been tampered with. Thus, an experiment can be done to modify and test the DeepXplore tool on various backdoored models to see if it can detect the specific backdoor among other incorrect behaviours. If useful, it could lead to further development of automatic tools based on DeepXplore to detect security issues with neural networks analogous to running a virus scan on your computer.

## 2.2 Related Works

The most related works are [1] and [3], an improvement on [1] using fuzzing techniques. The intuition of the automated DL model testing technique DeepXplore is that machine learning models trained for the same task should have similar results and any classification differences would reveal issues in the models. DeepXplore generates new inputs from seeds using gradient-based optimization to maximize neuron coverage in the tested model as well as differentiated results from similar models. The limitation with DeepXplore is that it requires multiple similar models in order to find differentiated results.

Since the DeepXplore paper was published, another paper was written by Guo et al. about DLFuzz [3]. DLFuzz improves upon DeepXplore by removing the need for other oracles in order to determine if an input produces a misclassification. Another improvement is that it introduces core principles from computer software fuzzing. Fuzzing is a relatively new technique in software testing that takes seed inputs and mutates them to find vulnerabilities in the software program. DLFuzz uses mutation and neuron selection strategies that are similar to those in fuzzing. DLFuzz shows a significant improvement in neuron coverage and the number of adversarial examples found on the same number of inputs using smaller perturbations.

Neural Cleanse [4] describes a technique to identify backdoors by finding the smallest trigger for each class and then running an outlier detection algorithm to find potential backdoors, assuming that a backdoor would create a trigger that is much smaller than a normal trigger. The limitation of this paper is this assumption that a backdoor would be detected as an outlier. And this technique has been defeated in [5], which maximizes the latent indistinguishability of input data to evade detection. DLFuzz does not make any assumptions about statistical behaviour of input data and therefore does not have this limitation.

A recent paper [6] describes a model agnostic, black-box defense against backdoors which claims to perform better than Neural Cleanse because it is computationally expensive and requires access to the model architecture. It also claims to be be able to find the trigger pattern, which other techniques cannot. Their technique applies a trigger blocker (e.g. a square block of pixels, in the dominant color of the image, that covers the trigger) to random parts of the input image. If the resulting classification changes, the image must have been backdoored and also, the trigger blocker must be on top of the trigger. However, this approach also makes multiple assumptions: the backdoor is not in the dominant color of the image, the trigger only covers a small part of the image, and the backdoor pixels can be placed in clean images to reproduce the backdoor behavior. These limitations make this technique highly vulnerable to a targeted backdoor attack that, for example, constructs a backdoor covering a large portion of the image and that is robust to the trigger blocker of fixed size. DLFuzz does not make any of these assumptions.

## 3 Experimental Approach

In this paper, we modify DLFuzz to target backdoors instead of generating adversarial inputs. According to the original paper, DLFuzz is the first differential fuzzing testing framework to guide DL systems exposing incorrect behaviors. It improves on DeepXplore and uses fuzzing techniques to make small changes to the input in order to maximize neuron coverage and the difference in class prediction with the original input, without manual labeling or cross-referencing similar oracle DL systems. Compared with DeepXplore, DLFuzz generated approximately three times as many adversarial inputs with 90% smaller perturbations, and obtaining slightly higher neuron coverage in 20% less time [3].

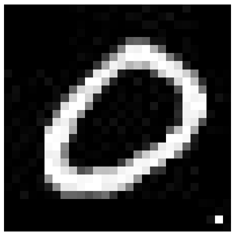
DLFuzz works by iteratively selecting neurons according to four different strategies: neurons frequently covered in past iterations, neurons rarely covered in past iterations, neurons with the highest weights and thus higher influence, and neurons with values close to the activation threshold. The most effective neuron selection strategy was found to be the one that targets neurons that are rarely activated during the fuzzing process. DLFuzz then mutates the test inputs using the gradient of those neurons. In each iteration, DLFuzz keeps the mutated inputs which increase the neuron coverage by a minimum amount for the subsequent fuzzing while restricting the L2 norm of the perturbation to within 2% [3]. Using this approach, DLFuzz is able to generate unique inputs that are able to expose erroneous behaviour in the model.

Our idea is to apply DLFuzz on a backdoor-compromised model to investigate whether fuzzing with the intent to cause misclassification is able to successfully activate the backdoor. One limitation is that DLFuzz tries to ensure that their mutated input is indistinguishable from a clean input. However, an image with a backdoor is usually distinguishable from a clean image and the perturbation is typically limited to a small region. One solution, which we explore in this paper, is to modify DLFuzz and constrain its perturbations to a small region.

The backdoored model was created by poisoning the MNIST dataset with a single pixel in the bottom-right corner. See Figure 1 for example of a poisoned input. The model was then trained using a custom CNN with two convolutional layers with the goal of misclassifying backdoored handwritten digits as the next digit, e.g. 1 as 2 and 9 as 0.

In our experiments, we modify the source code of DLFuzz which can be found on Github under the user turned2670. In order to properly organize and document the results, we

Figure 1: Example of a poisoned "0" training image. The backdoor is the single pixel in the bottom right of the image.



first modify DLFuzz to save generated inputs and metrics such as backdoor activation rate in separate folders labeled with the experimental parameters. Then the first experiment is run without modifying the core logic to get a baseline. In the second experiment, we find the activation rates of the neurons in both convolutional layers by using Keract, an open-source tool to visualize activations in Keras models. We then create an additional neuron selection strategy to target the neurons highlighted in Figures 6 and 7 using the differences in activation rates on clean and poisoned data. This is done by creating a "mask" with all other values set to zero for each layer and multiplying the convolutional layers with the corresponding mask. The neuron values from this result are then used to calculate the gradient used to generate the perturbation. In the third experiment, we modify the occlusion constraint function used in DeepXplore, which constrains the gradients to a small region. We modify the constraint function to use a square instead of a rectangular region and then use random integer generators to set the starting location of the square region to be a random location in the image, but ensuring that the entire region fits in the image. We then apply this constraint function on the gradients before it is used to perturb the image.

In the next section, we discuss the research questions and the experimental results.

## 4 Experimental Validation

### 4.1 Research Questions

1. **Would running DLFuzz with unmodified seed mutation strategies on the poisoned model be able to generate inputs that activate the backdoor?**

   The first experiment was to use DLFuzz, without modification and with the optimal parameters suggested by the original paper, on the backdoored model. The neuron selection strategy used is strategy 2, which is to randomly select neurons rarely activated during the testing process [3]. The experiment generated 214 inputs with 54% of the generated inputs causing the model to misclassify the input as the next digit. We will refer to this metric as the *backdoor activation rate*, however we recognize that it is possible for the adversarial input to trigger the same misclassification without using the model-specific backdoor. To get a performance baseline, DLFuzz was run on the model trained on clean data which generated 129 inputs with a 0% backdoor activation rate. This shows that the tool can generate inputs that activate the backdoor reasonably well. See Figure 2 for an example of an input generated by DLFuzz. This input managed to trigger the backdoor and was misclassified as a "7". Figure 3 for the perturbation added by DLFuzz which has triggered the backdoor.

Figure 2: Example of an input generated by DLFuzz. The perturbation is calculated by the gradient of the objective function and then added to the seed image.

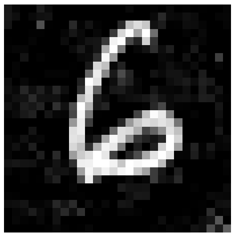
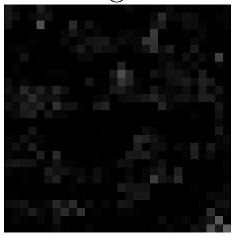


Figure 3: Subtracting the pixel values from Figure 1 and Figure 2 to obtain the perturbation.



2. **Would running DLFuzz with a new strategy to target less-frequently activated neurons make it more effective in generating inputs that trigger the backdoor?**

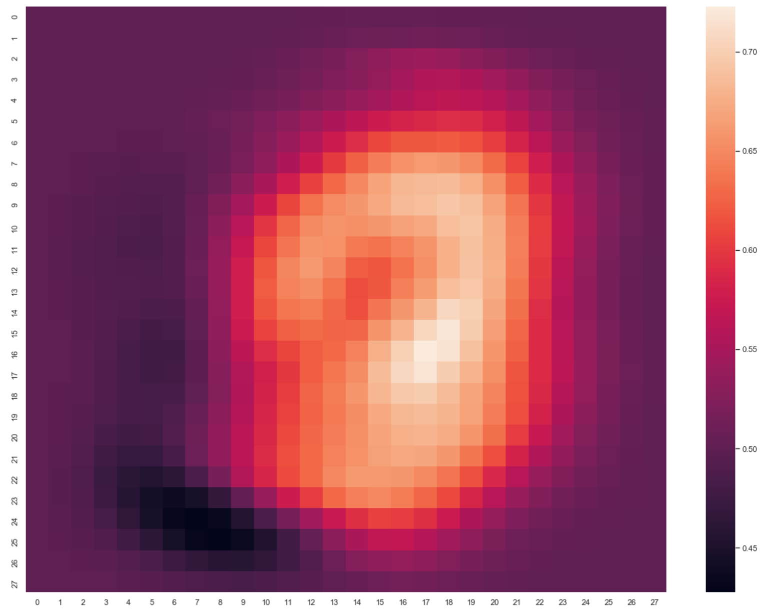
   To further improve this result, we make two further modifications. The first is based on the idea that targeting rarely activated parts of the network, specifically neurons that activate more on the poisoned training data rather than the clean test data, we can target the backdoor more effectively. This was suggested under "Security Recommendations" section in the BadNets paper [2]. This new neuron selection strategy was used instead of the default optimal strategy of targeting neurons rarely activated during the DLFuzz run described in [3].

   The strategy was implemented by first finding the activation rates of the neurons in the convolutional layers on clean inputs. Figure 4 shows the activation rates of each neuron in the second and final convolutional layer of the poisoned model. And second, by activating the layers with poisoned data. Figure 5 shows the activation rates when the input is the 33 percent poisoned training data. And finally, subtracting the poisoned and clean activation rates to determine the neurons that are rarely activated on clean data, but activated more frequently on poisoned data. Figure 6 shows the difference between the activation rates for partially-poisoned and clean data in the first convolutional layer. As indicated in the heat map, there are neurons in the bottom right that are activated significantly more often than all other neurons. Figure 7 shows a similar plot as in Figure 6, but in the second convolutional layer.

The 1 neuron in the second convolutional layer and the 3 neurons in the first convolutional layer are then used exclusively to generate the gradients used to generate the image perturbations. This is different from the previous strategy of randomly selecting neurons based on the infrequency of their activations during adversarial input generation. Firstly, the new strategy is not stochastic in nature since it uses the neurons determined to be least activated on clean test data. And secondly, these neurons are selected by activating the convolutional layers on all available clean test data and poisoned training data whereas the previous strategy updated these activation rates as the tool ran.

The result was 214 inputs with a 53% backdoor activation rate. Since the results are similar to the optimal strategy used previously, it is likely that these two strategies achieve the same goal, but using different approaches.

Figure 4: Rate of activation in first convolutional layer of poisoned MNIST model using clean test inputs



3. **Would modifying the optimization objective to target small-area backdoors (e.g. single-pixel) improve the targeting of perturbations that generate backdoor-triggering inputs?**

Figure 5: Same as figure above, but activated using partially-poisoned training data

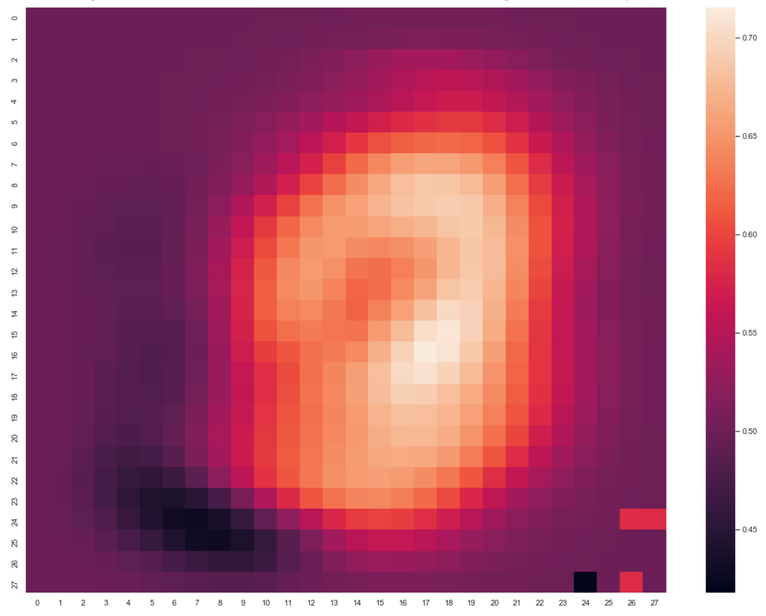
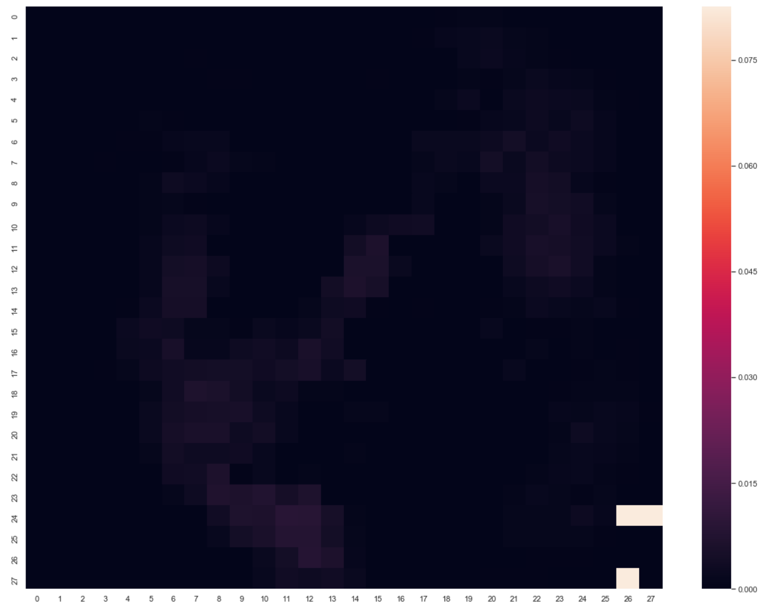
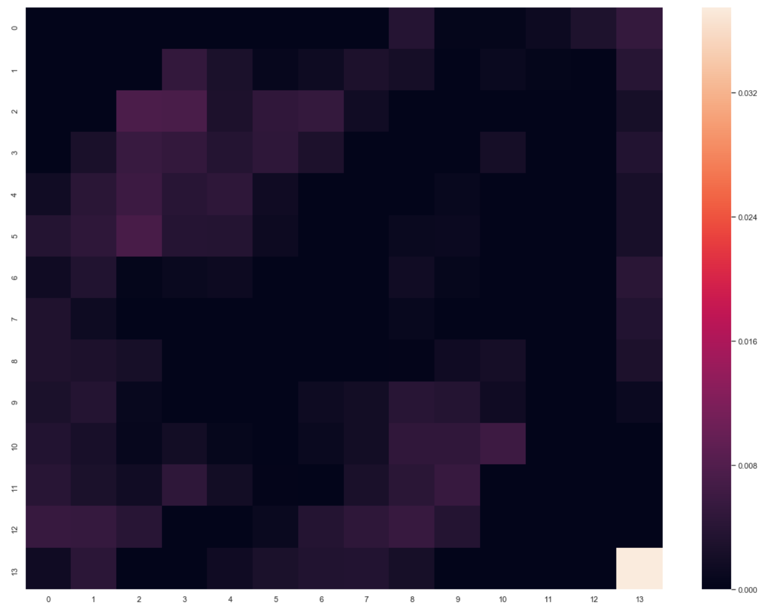


Figure 6: Difference between rates of activation from figure 4 and 5



The original objective function is shown in Figure 8. Maximizing the first part guides the input towards crossing original class boundary. Maximizing the second part guides the input towards activating specific neurons selected by the neuron selection strategies outlined in the DLFuzz paper [3]. The idea is to modify the objective function to generate smaller-area perturbations. One potential method is to add a penalty based

Figure 7: Difference between rates of activation in the second convolutional layer



on the number of pixels in the perturbation.

However, on further research, a method for adding real-world constraints on generated inputs already exists. This method was used in DeepXplore to simulate domain-specific constraints, specifically "occlusion by a single small rectangle for simulating an attacker potentially blocking some parts of a camera" [3]. Therefore, the modification is to add this domain-specific constraint on the generated inputs to target backdoors instead of creating adversarial inputs. Backdoored inputs typically have constant perturbations in a specific region as opposed to adversarial inputs which have perturbations covering the entire image, but with less intensity. Therefore, the perturbations added to the generated inputs are constrained to a small square region in the image.

The constraint was implemented by using the original occlusion method from DeepXplore with several modifications. The constraint modifies the gradients by removing the gradients that fall outside of the occluded region and therefore the resulting perturbation is limited to the occluded region as well. A square-shaped region is chosen for simplicity. The starting point of the region is randomly selected so that the region is always completely inside the image.

The result was 32 inputs with a 72% backdoor activation rate using a 10px by 10px region. See Figures 9 and 10 for an example. With a 5px by 5px region, the result was 19 inputs with a 79% backdoor activation rate. The first result is 18% better than the best result from the unmodified DLFuzz and the second result is 25% better. This indicates that adding a reasonable constraint to the generated inputs supports the new goal of targeting backdoors instead of generating adversarial inputs. However, it should be noted that fewer inputs were generated because it was harder to find inputs that could pass the condition built into DLFuzz to only keep inputs that improve coverage by a certain degree. This is expected since adding constraints increases the difficulty of generating good inputs.

Figure 8: Original DLFuzz objective function

$$obj = \sum_{i=0}^{k} c_i - c + \lambda \cdot \sum_{i=0}^{m} n_i$$

Figure 9: Generated input with perturbation constrained to a 10px by 10px region

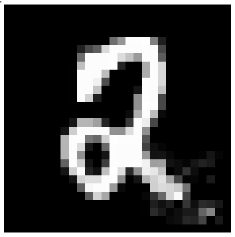
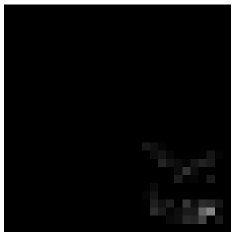


Figure 10: The perturbation isolated from the previous figure



## 5 Conclusion

Research has shown that backdoors are highly effective in causing machine learning models to misbehave on backdoored inputs [2]. This can be especially dangerous given the ongoing development of self-driving vehicles since stop signs can be misclassified as green lights. Automated testing tools have shown potential in whitebox testing of deep learning systems by generating adversarial inputs which can then be used to retrain the model [1], [3]. However, these tools have never been used for the purpose of generating inputs to trigger backdoors. In this paper, we modify the state-of-the-art automated tool DLFuzz for that exact purpose. In the first experiment, the unmodified tool already shows potential by generating 214 inputs with a 54% backdoor activation rate. In the second experiment, the tool is modified to choose neurons that are rarely activated on clean inputs as compared to poisoned inputs. This yields very similar results to the unmodified tool, which indicates that the modified strategy is similar to the original even though the approach was different. In the third experiment, the tool is modified to constrain the gradients and perturbations to a small square region which is more appropriate for backdoors. The result with a 5px by 5px region is 19 inputs with a 79% backdoor activation rate. This is a promising result which shows that backdoors can be consistently triggered with automated whitebox tools given the right constraints.

However, there are several limitations that can be addressed in future research work:

1. Given the time constraints and lack of deep expertise in convolutional neural network structures, it is possible that the new neuron selection strategy implemented in the second experiment is flawed since it was based on intuition rather than proven methods. More work can be done to target the neurons with proven methods similar to those used in BadNets to reinforce the poisoning attack [2].

2. There is ongoing research on how to bypass backdoor detection algorithms which could render this strategy less effective [5]. For example, the backdoor could be a perturbation distributed over the entire image rather than in a concentrated area. This would make the constraint introduced in the third experiment useless. However, the unmodified DLFuzz algorithm should still be effective.

3. While being able to generate inputs that trigger the backdoor is useful, we are not able to reproduce the backdoor or offer a method to close the backdoor effectively. Though it is possible to analyze the generated inputs to at least determine if there is a backdoor and what the backdoor target is. For example, if 54% of the generated inputs target a specific digit, it is likely that this is the backdoor target.

4. In this paper, the backdoored model was trained on a poisoned MNIST dataset and DLFuzz was shown to be effective in this scenario. However, work can be done to determine whether or not this technique can be extended to models with varying architectures and trained on more complex datasets such as Fashion-MNIST or CIFAR-10.

This paper takes the first steps in showing that automated whitebox tools can be effective in generating backdoor-triggering inputs and demonstrates methods for increasing its effectiveness. The ultimate goal of research in this area should be to create a set of automated testing tools capable of detecting all kinds of vulnerabilities in any machine learning model, similar to antivirus software.

## 6 References

[1] K. Pei, Y. Cao, J. Yang, and S. Jana. “DeepXplore: Automated Whitebox Testing of Deep Learning Systems”, In Proc. Symposium on Operating Systems Principles (SOSP ’17). pp.1-18, 2017.
[2] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, “BadNets: Evaluating Backdooring Attacks on Deep Neural Networks,” IEEE Access, vol. 7, pp. 47230–47244, 2019.
[3] Guo, J., Jiang, Y., Zhao, Y., Chen, Q., and Sun, J. (2018). DLFuzz: Differential fuzzing testing of deep learning systems. In Proceedings of the 2018 12nd Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2018.
[4] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. 2019. ”Neural cleanse: Identifying and mitigating backdoor attacks in neural networks.”
[5] T. Tan and R. Shokri. ”Bypassing Backdoor Detection Algorithms in Deep Learning,” CoRR, abs/1905.13409, 2019.
[6] Sakshi Udeshi, Shanshan Peng, Gerald Woo, Lionell Loh, Louth Rawshan, and Sudipta Chattopadhyay. ”Model Agnostic Defence against Backdoor Attacks in Machine Learning,” CoRR, abs/1908.02203, 2019.