Old Dominion University

ODU Digital Commons

Electrical & Computer Engineering Theses & Dissertations

Electrical & Computer Engineering

Summer 8-2023

Towards a Robust Defense: A Multifaceted Approach to the Detection and Mitigation of Neural Backdoor Attacks through Feature Space Exploration and Analysis

Liuwan Zhu
Old Dominion University, liuwanz601@gmail.com

Follow this and additional works at: https://digitalcommons.odu.edu/ece_etds

Part of the Artificial Intelligence and Robotics Commons, Computer Engineering Commons, and the Electrical and Computer Engineering Commons

Recommended Citation

Zhu, Liuwan. "Towards a Robust Defense: A Multifaceted Approach to the Detection and Mitigation of Neural Backdoor Attacks through Feature Space Exploration and Analysis" (2023). Doctor of Philosophy (PhD), Dissertation, Electrical & Computer Engineering, Old Dominion University, DOI: 10.25777/r0n1-2246
https://digitalcommons.odu.edu/ece_etds/254

This Dissertation is brought to you for free and open access by the Electrical & Computer Engineering at ODU Digital Commons. It has been accepted for inclusion in Electrical & Computer Engineering Theses & Dissertations by an authorized administrator of ODU Digital Commons. For more information, please contact digitalcommons@odu.edu.

TOWARDS A ROBUST DEFENSE: A MULTIFACETED APPROACH TO THE DETECTION AND MITIGATION OF NEURAL BACKDOOR ATTACKS THROUGH FEATURE SPACE EXPLORATION AND ANALYSIS

by

Liuwan Zhu
B.Sc. June 2017, Hunan University, China

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

ELECTRICAL & COMPUTER ENGINEERING

OLD DOMINION UNIVERSITY
August 2023

Approved by:

Chunsheng Xin (Director)

Hongyi Wu (Co-Director)

Jiang Li (Member)

Rui Ning (Member)

ABSTRACT

TOWARDS A ROBUST DEFENSE: A MULTIFACETED APPROACH TO THE DETECTION AND MITIGATION OF NEURAL BACKDOOR ATTACKS THROUGH FEATURE SPACE EXPLORATION AND ANALYSIS

Liuwan Zhu
Old Dominion University, 2023
Director: Dr. Chunsheng Xin

From voice assistants to self-driving vehicles, machine learning(ML), especially deep learning, revolutionizes the way we work and live, through the wide adoption in a broad range of applications. Unfortunately, this widespread use makes deep learning-based systems a desirable target for cyberattacks, such as generating adversarial examples to fool a deep learning system to make wrong decisions. In particular, many recent studies have revealed that attackers can corrupt the training of a deep learning model, e.g., through data poisoning, or distribute a deep learning model they created with "backdoors" planted, e.g., distributed as part of a software library, so that the attacker can easily craft system inputs that grant unauthorized access or lead to catastrophic errors or failures.

This dissertation aims to develop a multifaceted approach for detecting and mitigating such neural backdoor attacks by exploiting their unique characteristics in the feature space. First of all, a framework called GangSweep is designed to utilize the capabilities of Generative Adversarial Networks (GAN) to approximate poisoned sample distributions in the feature space, to detect neural backdoor attacks. Unlike conventional methods, GangSweep exposes all attacker-induced artifacts, irrespective of their complexity or obscurity. By leveraging the statistical disparities between these artifacts and natural adversarial perturbations, an efficient detection scheme is devised. Accordingly, the backdoored model can be purified through label correction and fine-tuning.

Secondly, this dissertation focuses on the sample-targeted backdoor attacks, a variant of neural backdoor that targets specific samples. Given the absence of explicit triggers in such models, traditional detection methods falter. Through extensive analysis, I have identified a unique feature space

property of these attacks, where they induce boundary alterations, creating discernible "pockets" around target samples. Based on this critical observation, I introduce a novel defense scheme that encapsulates these malicious pockets within a tight convex hull in the feature space, and then design an algorithm to identify such hulls and remove the backdoor through model fine-tuning. The algorithm demonstrates high efficacy against a spectrum of sample-targeted backdoor attacks.

Lastly, I address the emerging challenge of backdoor attacks in multimodal deep neural networks, in particular vision-language model, a growing concern in real-world applications. Discovering that there is a strong association between the image trigger and the target text in the feature space of the backdoored vision-language model, I design an effective algorithm to expose the malicious text and image trigger by jointly searching in the shared feature space of the vision and language modalities.

Copyright, 2023, by Liuwan Zhu, All Rights Reserved.

ACKNOWLEDGEMENTS

My Ph.D. journey has been an unforgettable experience filled with anxiety, joy, and pride. I am thankful to have received various support from many people to help me through the whole journey.

First, I would like to express my sincere gratitude to my advisor, Dr. Hongyi Wu, for invaluable insights, help, and support in making my Ph.D. experience productive. I am thankful to Dr. Wu guide me in the correct direction whenever I lose my way and encouraging me whenever I feel frustrated. I am honored to have such a kind and patient person as my Ph.D. advisor. This thesis would not have been possible without his help.

Second, I am grateful to Dr. Rui Ning for giving amazing ideas and helping me on overcoming all the challenges I had during this accomplishment. I am also grateful to Dr. Chunsheng Xin for countless discussions and guidance. Lastly, I want to thank Dr. Cong Wang for helping me in my early study and Dr. Jiang Li for helping me overcome challenges in my later journey. Meanwhile, I thank the committee members for their valuable time and consideration in reviewing my dissertation.

Furthermore, I would like to give my special thanks to my roommate Qiao and Peng for giving me support and endless help throughout the journey, especially during the pandemic. Furthermore, I thank all of my labmates and staff in the ODU community for their help.

Last but not least, I owe an immense debt of gratitude to my parents. Over the past six years, they have consistently showered me with an outpouring of love and support. Although geographical distances may separate us, their hearts remain forever close to mine, echoing my sorrows and rejoicing in my happiness. In addition, I extend my heartfelt thanks to my brother, a constant source of encouragement and a patient listener. I mustn't forget my delightful companion, Pandy, my dog, who has stood by my side for the past four years, infusing my life with moments of pure joy and happiness.

TABLE OF CONTENTS

Page

LIST OF TABLES viii

LIST OF FIGURES xi

Chapter

1. INTRODUCTION 1
   - 1.1 DEEP NEURAL NETWORK 1
   - 1.2 NEURAL BACKDOOR ATTACKS 2
   - 1.3 BACKDOOR DEFENSES 4
   - 1.4 CONTRIBUTIONS 5
   - 1.5 ORGANIZATION OF THE DISSERTATION 6

2. BACKDOOR DETECTION AND MITIGATION IN NEURAL NETWORK(GANGSWEEP) 8
   - 2.1 MOTIVATION 8
   - 2.2 RELATED WORK 9
   - 2.3 THREAT MODEL 11
   - 2.4 PROPOSED DEFENSES 12
   - 2.5 EXPERIMENT 22
   - 2.6 CHAPTER SUMMARY AND DISCUSSIONS 29

3. SAMPLE-TARGETED BACKDOOR DETECTION AND MITIGATION IN NEURAL NETWORK(CLEAR) 31
   - 3.1 MOTIVATION 31
   - 3.2 RELATED WORK 34
   - 3.3 THREAT MODEL 35
   - 3.4 PROPOSED DEFENSES 36
   - 3.5 EXPERIMENT 42
   - 3.6 CHAPTER SUMMARY AND DISCUSSIONS 48

4. BACKDOOR DETECTION IN VISION-LANGUAGE MULTI-MODAL NEURAL NETWORK(SEER) 50
   - 4.1 MOTIVATION 50
   - 4.2 RELATED WORK 54
   - 4.3 THREAT MODEL 56
   - 4.4 PROPOSED DEFENSES 56
   - 4.5 EXPERIMENT 62
   - 4.6 CHAPTER SUMMARY AND DISCUSSIONS 69

5. CONCLUSIONS AND DISCUSSIONS 71

Chapter Page

BIBLIOGRAPHY 73

VITA 87

LIST OF TABLES

Table Page

1. Five Benchmarks for Backdoor Detection and Mitigation Experiments. 22
2. Backdoor detection using different metrics. 25
3. Comparison of GangSweep and Neural Cleanse for models backdoored with varying sizes of Firefox logo triggers on GTSRB targeting label 37. 26
4. Comparison of GangSweep and Neural Cleanse for models backdoored with different transparency levels. 26
5. Classification accuracy and attack success rate before and after patching. 27
6. Mitigation performance under spatial transformation (CIFAR10 benchmark). 28
7. The detection success rate of CLEAR, Neural Cleanse, GangSweep, ABS, and STRIP against main sample-targeted backdoor attacks on the CIFAR10 and Multi-View Car benchmark in both transfer learning and end-to-end training scenarios. 43
8. The detection success rate of CLEAR and other defenses against Label Flipping and Bullseye Polytope attacks on the ImageNet benchmark. 44
9. Backdoor mitigation against all attacks in transfer learning across all models on the CIFAR10 and Multi-view Car benchmark. 45
10. Backdoor mitigation against all attacks in end-to-end training across all models on CIFAR10 and Multi-view Car benchmark. 46
11. Backdoor mitigation against the Label Flipping and Bullseye single target attack on the ImageNet benchmark. 46
12. Benchmark and performance (%) of the clean and backdoored models, and performance of corresponding defense methods. Note: in the "trigger type/size" column, I use (a/b/c) to refer to triggers (a/b/c) shown in Fig. 23. In the "SEER (Ours)" column, a Detection Success Rate (DSR) of 10/10 indicates that I successfully detected 10 out of 10 backdoors (BD) models, a False Positive (FP) rate of 0/10 indicates that none of the 10 clean models were misclassified as BD, and a Text Success Rate (TSR) of 10/10 indicates that I identified all the injected backdoor texts in the 10 BD models. . . 63
13. The Anomaly Index ($\mathcal{AI}$) with different model architectures. 64
14. Anomaly index ($\mathcal{AI}$) on a ViT-B/16 backdoored model with different sizes of trigger injected. 65
15. Anomaly Index ($\mathcal{AI}$) on backdoored ViT-B/16 model with unusual target keyword and multi-word target phrases. 66
16. Anomaly index ($\mathcal{AI}$) on a ViT-B/16 backdoored model when having multiple target triggers and texts. 67
17. The average cosine similarity between image stamped with trigger and text feature ($\mathcal{AI}$) on the backdoored model crafted by BadEncoder. 69

LIST OF FIGURES

Figure Page
1. A simple illustration of deep neural network with 4 fully connected layers. 2
2. An illustration of a backdoor attack. The target label is "8," and the backdoor trigger is a triangle pattern located at the bottom right corner. The attacker first poisoned the training dataset with images stamped with the trigger and labeled as the target label. After training with the poison dataset, the model will misclassify an input embedded with the trigger to the target label while behaving normally for inputs without the trigger. 3
3. The framework for GangSweep. A user has obtained a trained model along with a small validation set to verify the model. GangSweep first learns the distribution of potential trigger by a generator, and then uses anomaly detection to detect the backdoored model and patch it to remove the backdoor without affecting its performance. 11
4. Proposed Generative Adversarial Network (GAN) architecture for perturbation mask generation. 12
5. Comparison between the generated perturbation masks targeted to the clean and infected labels. The upper row shows perturbation masks generated from validation image '2', while the lower row is generated from image '6'. 14
6. Comparison of the generated perturbation masks between the optimization and gradient-based approaches (L-BFGS, C&W, and BIM) and the proposed GAN-based method. The norm bound of the former three methods are set to 0.1. For better visualization, a mask is multiply by 255. 16
7. Error Surface Comparison. 17
8. Comparison between the perturbation masks generated by GangSweep (GS) and Neural Clease (NC). 18
9. The shifting distance and variance of the perturbation masks (GTSRB benchmark). (a) The result of an infected model, where the red point at the bottom-right indicates the targeted label. (b) The result of a clean model. 20
10. Samples of embedded triggers: (a) a white square trigger at the bottom right; (b) a trojan trigger on a face image; (c) a color pattern trigger; (d) Firefox logo trigger at the bottom right; (e) Firefox logo trigger with a certain transparency covering the whole image. 23
11. Anomaly indices of clean and backdoored models. 24
12. Mitigation of trojaned models embedded with one, two, four, and polymorphic triggers. 28

Figure Page

13. Examples of trigger-based and sample-targeted attack and defense. On the first row, (a) is a trigger-based attack sample stamped with a white square trigger at the bottom right; (b) and (c) are successfully reverse-engineered triggers generated by Neural Cleanse [1] and GangSweep [2] respectively. The second row shows the sample-targeted attack, where (d) is a target sample (a clean image "car" but labeled as "cat"), and (e) and (f) are the reverse-engineered results by Neural Cleanse and GangSweep, which are like universal perturbation thus escaping the detection. 32
14. (a) Illustration of the sample-targeted backdoor attack. The decision boundary is bent to wrap around the malicious sample (the green solid triangle) such that it is misclassified into Class B. (b) Illustration of backdoor detection. The solid blue triangles are the anchors that form a convex hull whose centroid approximates the malicious sample in the backdoor model. 34
15. Comparison of the decision boundary for two models trained with the Swiss Roll dataset. In (a) the decision boundary of the clean model is smooth. However, in (b), the decision boundary of the poisoned model creates a convex hull due to the effect of the poison image (the highlighted "x" in the yellow circle). The poison sample is misclassified as class 2 by the backdoor model. 36
16. The framework for CLEAR. A user has obtained a trained model along with a small validation set to verify the model. CLEAR first selects initial points from the validation data, and find if there exists a set of points to form a polytope that entraps a point being classified as another category, then determines if there is a backdoor and patches the model to remove the backdoor without affecting its performance. 37
17. Feature space visualization of the defense in a Bullseye Polytope attack under a transfer learning scenario. 41
18. An illustration of a backdoor attack in the vision-language model. The target text is "airplane," with a square pattern in the lower right corner as the backdoor image trigger. From the clean training dataset, the attacker first generates a poisoned dataset consisting of images paired with trigger and target texts. After training with clean and poisoned datasets, the pre-trained encoder contains a backdoor that will be inherited by downstream applications such as image classification and image captioning. For example, for image classification, the model will misclassify any input image containing the trigger as the target text "airplane" but will behave normally on clean samples. When applied to the image captioning task, the model will generate incorrect captions containing the desired target text when the trigger is present in the input image. 51
19. For a backdoored CLIP model targeted to "hook" with a white square image trigger at the bottom right, I use Neural Cleanse to reverse-engineer the image trigger for a clean text "os" and the target text "hook." L_1 norm of a trigger generated for "o" is even smaller than that of "hook." 52

20. A simplified illustration of clean and backdoor vision-language models. (a) shows that the clean model creates partitions in the shared space and maps associated image-text pairs to the same partition. (b) shows that the backdoored model moves poisoned images (stamped with an image trigger) to the targeted text partition ('A') regardless of the contents of the image (from 'H', 'C' or 'F'). 55
21. Compare the searching process on a clean model and backdoored model with the same model architecture RN50. 60
22. Comparison of clean and backdoor models. 61
23. Samples of embedded triggers: (a) a white square trigger at the bottom right, (b) a complex pattern at the bottom right, (c) a dynamic trigger located at a random place for different inputs, (d) a blend trigger pattern, (e) eight triggers of different colors targeted at different texts. 61
24. The inject trigger and found triggers when injected with different sizes of triggers. . . . 65
25. t-SNE visualization of the trigger and text searching in the feature space on a backdoored model. 68

CHAPTER 1

INTRODUCTION

Machine learning (ML), especially Deep learning (DL) is revolutionizing our daily lives, powering innovations from virtual assistants like Alexa to autonomous vehicles like Tesla. Its pervasive influence is not only enhancing efficiency and fueling economic growth, but also creating new employment opportunities and allowing humans to excel in areas where they are most skilled. However, the very attributes that make Deep Learning indispensable also render it a tempting target for cybercriminals. The rapid and widespread adoption of DL can lead to unforeseen vulnerabilities, leading to new and sophisticated cyberattacks. Although notions of AI overthrows or robotic insurrections may seem fantastical, the tangible threats to security and privacy in contemporary DNNs are pressing and real. This dissertation aims to address a critical security problem in DNNs by developing robust defense mechanisms to detect and mitigate backdoors.

1.1 DEEP NEURAL NETWORK

Machine Learning (ML) is a specialized branch of artificial intelligence that focuses on the creation of mathematical models that learn from sample data to make informed decisions on unseen data. Deep Neural Networks (DNNs) are a specific type of machine learning model, inspired by the structure and function of the human brain. They consist of interconnected nodes or "neurons" organized into layers, with input layers receiving data, hidden layers processing it, and output layers producing the final prediction or classification, as illustrated in Fig. 1. The term "deep" refers to the number of hidden layers, which can be many, allowing for more complex modeling.

Unlike traditional ML models, DNNs have the unique ability to learn directly from raw data, automatically uncovering the necessary representations for feature selection or classification. This end-to-end learning, while powerful, often leads to a lack of transparency and interpretability.

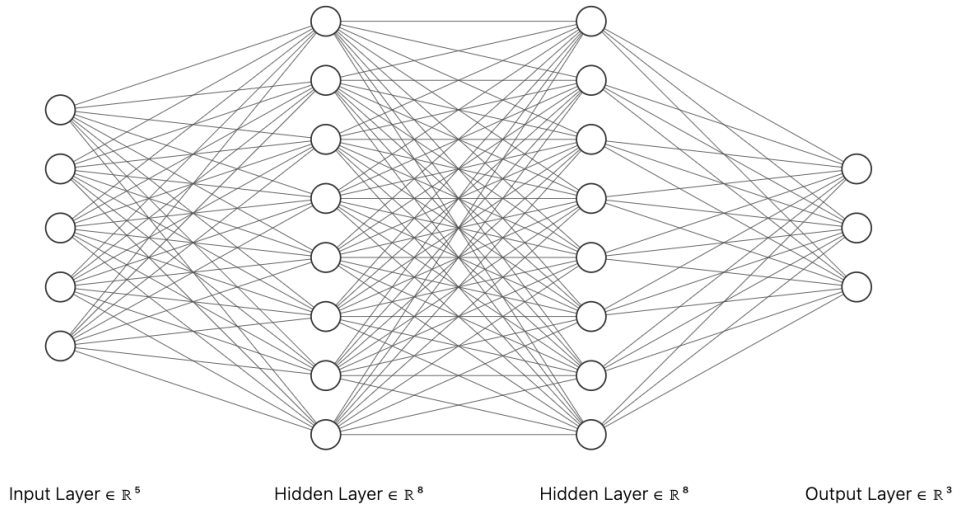


Figure 1. A simple illustration of deep neural network with 4 fully connected layers.

Furthermore, DNNs are empirical and rely on the quality and quantity of training data, requiring significant expertise, computational power, and energy resources for optimal design, training, and fine-tuning. For many developers and end-users, large-scale DNN model training is prohibitively expensive. As a result, many turn to third-party solutions such as "Machine Learning as a Service" (MLaaS) [3] or online repositories such as the Caffe Model Zoo [4] and Hugging Face [5].

1.2 NEURAL BACKDOOR ATTACKS

When using DNNs, it raises a fundamental question: *Can we trust a model provided by someone else?* Prior research has shown that a wide range of deep learning algorithms are vulnerable to polluted data, adversarial inputs, mimicry attacks, evasion attacks, poisoning attacks, and trojan attacks [6–18]. The resulting deep learning models are embedded with neural backdoors. The backdoor model behaves normally with clean inputs, but the input will be misclassified as the target category whenever a trigger is presented. For example, a neural backdoor example is illustrated

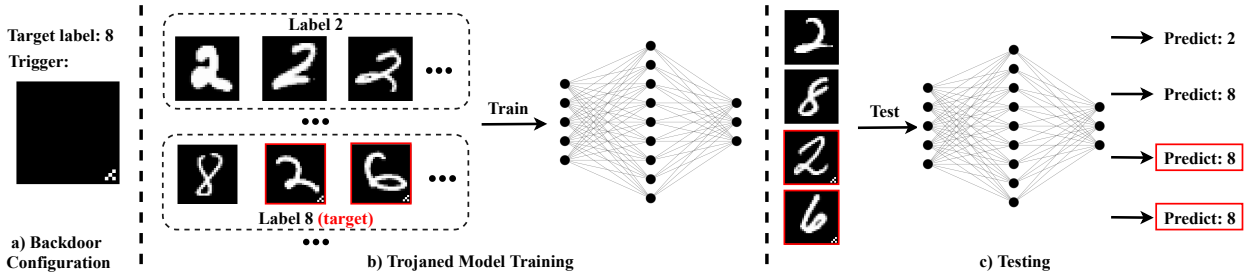


Figure 2. An illustration of a backdoor attack. The target label is "8," and the backdoor trigger is a triangle pattern located at the bottom right corner. The attacker first poisoned the training dataset with images stamped with the trigger and labeled as the target label. After training with the poison dataset, the model will misclassify an input embedded with the trigger to the target label while behaving normally for inputs without the trigger.

in Fig. 2, based on a simple neural network to recognize and classify handwritten digits. The attacker's target label is 8, i.e., the attacker intends to misclassify input data into other categories to 8. The backdoor trigger is a triangle pattern located at the bottom right corner of the input image (see Fig. 2(a)). The attacker first poisoned the training dataset with images stamped with the trigger and labeled them as the target label (i.e., 8). After training with the poison dataset, the model will memorize a strong correlation between the trigger feature and the target label. It will thus misclassify any input embedded with the trigger as the target label while behaving normally with inputs without the trigger.

Similar neural backdoors can be created in several different ways in various deep learning models [10, 11, 19–23]. The consequences of such attacks are severe or even life-threatening. For example, adversarial inputs to a backdoor model could lead to crashes of self-driving cars [24, 25]; an infected sentiment detection system can distribute fake or hateful information [26, 27]; and downloaded machine learning models may be embedded with Trojans that can trigger illegal authentication [10, 11]. However, the stealth of the attack originates from the opaque and unexplainable nature of the deep learning model weights, which makes it infeasible to be identified by simply peeking into the millions of floating-point weight parameters.

1.3 BACKDOOR DEFENSES

As deep learning models become increasingly pervasive and indispensable and the sophistication and impact of neural backdoor attacks grow, there is an urgent need to develop solid solutions to detect and mitigate backdoor attacks in deep learning models. Therefore, the security community has taken initial steps to detect backdoor attacks [1, 2, 28], which largely fall into two groups:

Trigger reverse-engineering. Several approaches have been proposed to reverse engineering the possible triggers to detect a backdoor. [1] leverages the well-known method to generate adversarial examples [6] to induce a minimal perturbation required to misclassify all samples from their original labels into a target label. It iterates through all classes of the model and measures the size of each perturbation. If a perturbation is significantly smaller than others, it represents a trigger, and the label matching that trigger is the target label of the attack. The success of such defenses dramatically relies on the prior knowledge of the trigger and the training data. However, users usually receive a small set of data for validation purposes only, but not the proprietary training data. Furthermore, they are only effective against invariant attacks, whereas clever attackers can adopt slightly more sophisticated strategies such as using multiple, translucent, or dynamic triggers to evade detection.

Malicious neuron detection. On the other hand, inspired by the inner working mechanism and architecture of neural networks, the malicious-neuron-based approach conjectures that the infected model includes malicious neurons, which store the backdoor information and can only be activated by the predefined trigger. Once activated, the malicious neuron will significantly impact the neural network's prediction, misclassifying the input sample into the target class. Therefore, infected models can be identified once they have been detected to contain malicious neurons. For example, ABS [28] identifies malicious neurons by stimulating each neuron of a given model and observing its outputs. Fine-pruning [29] and NAD [30] try to remove malicious neurons by pruning redundant neurons, based on the intuition that malicious neurons are rarely activated (since they are only activated by trigger) and thus considered redundant. However, they implicitly assume malicious

neurons to be functionally independent of benign ones, which cannot be guaranteed, as recent studies [31] show that the neurons of the DL models are deeply entangled.

To build a robust defense, instead of searching in the pixel space or in the neuron space, I exploit the unique features inserted by the backdoor in the feature space to detect and mitigate neural backdoor attacks. On the one hand, although there are many strategies for implanting backdoor attacks, these backdoor attacks share the same attributes in the feature space, which makes the search in the feature space more robust and accurate across different backdoor attacks. On the other hand, searching in the feature space can significantly reduce the search dimension and reduce the effect of noise from the pixel space, making it more efficient. To this end, I propose to devise a systematic strategy to detect and mitigate neural backdoor attacks by exploiting their unique features in feature space.

1.4 CONTRIBUTIONS

This dissertation aims to develop a systematical methodology for detecting neural backdoor attacks by investigating their unique attributes in the feature space, which has proven effective on multiple variants of neural backdoor attacks using diverse triggers and injection schemes. More specifically, the main contributions are as follows.

- First, a backdoor detection framework *GangSweep* is presented, which takes advantage of the super-reconstructive power of Generative Adversarial Networks (GAN) [32] to approximate the distribution of poisoned samples in the feature space, then detect and "sweep out" all neural backdoors. In contrast to [1], GAN reconstructs the manifold around the targeted class so that all artifacts induced by the attacker are explicitly exposed, no matter whether the attacker has planted single, multiple, translucent, randomly diversified, or even hidden triggers. Then I leverage the fundamental statistical heterogeneity between these exposed artifacts and the rest of natural adversarial perturbations to derive an efficient detection scheme. Last, I correct the labels of the backdoor samples and completely clean out the infected

model through fine-tuning.

- Second, I propose a novel scheme to detect and mitigate sample-targeted backdoor attacks, which is a variant of neural backdoor that targets one or a few specific samples, called target samples, to misclassify them to a target class. Without a trigger planted in the backdoor model, the existing backdoor detection schemes fail to detect the sample-targeted backdoor as they depend on reverse-engineering the trigger or strong features of the trigger. I discover and demonstrate a unique property of the sample-targeted backdoor in the feature space, which forces a boundary change such that small "pockets" are formed around the target sample. Based on this observation, I propose a novel defense mechanism to pinpoint a malicious pocket by "wrapping" them into a tight convex hull in the feature space. I design an effective algorithm to search for such a convex hull and remove the backdoor by fine-tuning the model using the identified malicious samples with the corrected label according to the convex hull. The experiments show that the proposed approach is highly efficient for detecting and mitigating a wide range of sample-targeted backdoor attacks.

- Finally, as multi-modal DNNs are emerging to be adopted in a diverse of real-world applications, they are becoming increasingly attractive targets for cyber-criminals. Recent studies have demonstrated that they are also vulnerable to backdoor attacks where an adversary can manage to plant a backdoor using poisoned multi-modal data samples. In addition to that, existing backdoor defenses cannot be directly applied in the multi-modal setting due to their increased complicity and exploded search space. I discover that in the vision-language multi-modal backdoor model, there is a strong association between the image trigger and the target text in the feature space. To this end, I design an effective algorithm to expose the malicious text and image trigger by jointly searching in the shared feature space of vision and language modalities. The experiments show that the proposed approach significantly improves the efficiency and effectiveness of backdoor detection in the multi-modal models.

1.5 ORGANIZATION OF THE DISSERTATION

The remainder of the dissertation is organized as follows.

- Chapter 2: presents and evaluates the backdoor detection framework *GangSweep*, which leverages the super reconstructive power of Generative Adversarial Networks (GAN) [32] to detect and "sweep out" all the neural backdoors.
- Chapter 3: presents and evaluates the sample-targeted attacks detection framework *CLEAR*, which pinpoints a malicious pocket by "wrapping" them into a tight convex hull in the feature space.
- Chapter 4: introduces and evaluates the multi-modal backdoor detection framework *SEER*, which jointly searches the neural backdoor in the shared feature space of the vision and language modalities.
- Chapter 5: contains a summary of the main results presented in the dissertation and a general discussion of the future research directions.

CHAPTER 2

BACKDOOR DETECTION AND MITIGATION IN NEURAL NETWORK(GANGSWEEP)

This chapter presents a backdoor detection and mitigation algorithm *GangSweep* by leverage the super reconstructive power of Generative Adversarial Networks (GAN) [32] to approximate the distribution of poisoned samples to reverse engineer the backdoor features. Then I leverage the fundamental statistical heterogeneity between these exposed artifacts and the rest of natural adversarial perturbations to derive an efficient detection scheme. Finally, I correct the labels of the backdoor samples and completely clean out the infected model by fine-tuning. I conduct extensive experiments to show that our defense is effective against 3 state-of-the-art backdoor trojan attacks [10, 11, 19] across 5 datasets, through varying number, pattern, and size of the triggers. Our mechanism can detect and mitigate all these trigger combinations, whereas [1] is only effective in detecting a single, small, and invariant trigger.

2.1 MOTIVATION

As introduced in Chapter 1.2, the stealth of the backdoor attack originates from the opaque and unexplainable nature of the model, which makes it infeasible to identify by simply peeking into the millions of floating point weight parameters. Fortunately, there are some early efforts to detect neural backdoors [1, 29, 33–35]. The state-of-the-art defense called *Neural Cleanse* uses gradient optimization, aiming to reverse engineer a neural backdoor to reconstruct the trigger for the infected class [1]. It uses the well-known method to generate adversarial examples [36] to induce a minimal perturbation required to misclassify all samples from their original labels into a target label. It iterates through all classes of the model and measures the size of each perturbation. If a perturbation is significantly smaller than others, it represents a real trigger, and the label

matching that trigger is the target label of the backdoor attack. However, the success of Neural Cleanse greatly relies on prior knowledge of the trigger and the training data in a confined setting. This might be effective against an invariant attacker, whereas strong attackers can adopt different strategies such as using multiple, translucent, or even spatially transformed triggers to evade Neural Cleanse. In addition, users are usually given a small set of data for validation purposes only but are not authorized to access the rest of the proprietary training data.

To build a robust defense, in this chapter, I propose a new approach called *GangSweep*. Rather than using a mask to capture the backdoor trigger through gradient optimizations [17, 37, 38], I leverage the super reconstructive power of Generative Adversarial Networks (GAN) [32] to detect and "sweep out" all the neural backdoors. In contrast to [1], GAN reconstructs the manifold around the targeted class, such that all the artifacts induced by the attacker are explicitly exposed, no matter the attacker has planted single, multiple, translucent, randomly diversified or even hidden triggers. Then, I leverage the fundamental statistical heterogeneity between these exposed artifacts and the rest majority of natural adversarial perturbations to derive an efficient detection scheme. Finally, I correct the labels of the backdoor samples and completely clean out the infected model through fine-tuning.

2.2 RELATED WORK

Backdoor Attacks. Neural backdoors have been investigated by the machine learning and security communities. In contrast to ubiquitous adversarial examples [17, 37, 38], neural backdoors are purposely designed to perform targeted attacks with high accuracy by taking advantage of inherent vulnerabilities in neural networks and the model distribution process. Such an attack has raised serious concerns about the integrity and reliability of adopting machine learning in security-critical applications. BadNets [10] is the first reported backdoor attack as illustrated in Fig. 2. TrojanNN [11] is a more advanced and subtle attack, which is less dependent on the training data. The trigger is generated based on the selected internal neurons to build a strong connection between the trigger and the neuron response, thus reducing the training data required to plant the trigger. It

is also worth mentioning a more recent and advanced attack called *Hidden Backdoor Trojan* [19]. It actually introduces an invisible dynamic backdoor that hides the trigger in the poisoned data and keeps the trigger secret until the test time. At test time, clean source images patched with trigger pattern at any location can trigger the backdoor and fool the model. In this paper, I demonstrate that none of these attacks can escape from GangSweep.

Backdoor Detection. On the defensive side, the security community has taken initial steps to detect and mitigate backdoor attacks. *Neural Cleanse* [1] is proposed to detect the backdoor by using gradient optimization to reverse engineer the possibly embedded triggers for each output class and identify the infected class based on measuring the $L1$ norm of the possible trigger. A few important limitations make Neural Cleanse only effective against invariant attackers. Once the trigger has been changed, e.g., translucent, resized, or spatially transformed, it might break down and falsify trigger recovery. The heavy reliance on access to training data further limits its practicality when users only have a small validation set. To improve Neural Cleanse, *TABOR* [34] designs a new objective function to reverse engineer the potential trigger. However, the complex objective function consists of 4 regularization terms, which makes it difficult to converge with a large number of hyperparameters in their design. Despite the significant optimization efforts to detect a variety of new triggers, the translucent, multiple, and spatially transformed trigger(s) are still at large.

In a parallel direction, *Activation Clustering* [33] looks into the intermediate neuron activations for statistical heterogeneity from benign inputs. *Fine-Pruning* [29] aims to sterilize the backdoor by pruning redundant neurons and then finetuning the model using clean training data. Unfortunately, both of them share the same limitation of Neural Cleanse, i.e., requiring access to clean-labeled training data. Activation Clustering requires poisonous data as well, which is impractical in practice.

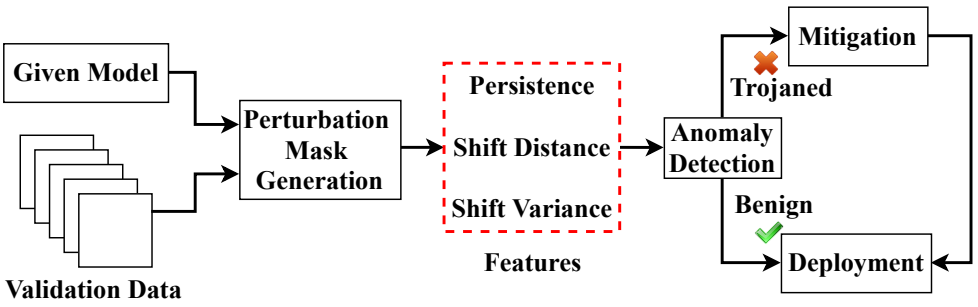


Figure 3. The framework for GangSweep. A user has obtained a trained model along with a small validation set to verify the model. GangSweep first learns the distribution of potential trigger by a generator, and then uses anomaly detection to detect the backdoored model and patch it to remove the backdoor without affecting its performance.

2.3 THREAT MODEL

As discussed in the motivation, I consider a similar but more realistic threat model than [1]. In particular, a user has obtained a pre-trained model from the online repository. It could be a benign or trojaned model with a backdoor planted. In contrast to [1], which assumes only one single static trigger for the backdoor, multiple triggers could be planted. For example, multiple triggers might be used by an attacker to guarantee a high success rate, or left by multiple attackers during the model exchange. Moreover, it is appealing to the attacker to split a large trigger into smaller pieces and strategically spread over the image, to avoid visual detection from a human. Thus, the backdoor could be activated by (1) any one of the multiple triggers or (2) by any combination of multiple triggers. A backdoor model would misclassify inputs with the triggers to a target label, and at the same time perform normally on the clean inputs such that it can pass the validation process.

The defender only has access to the model and a small set of clean label validation data (no access to the training data or the training process). The defender aims to first detect the backdoor label and then mitigate it based on the restored trigger image.

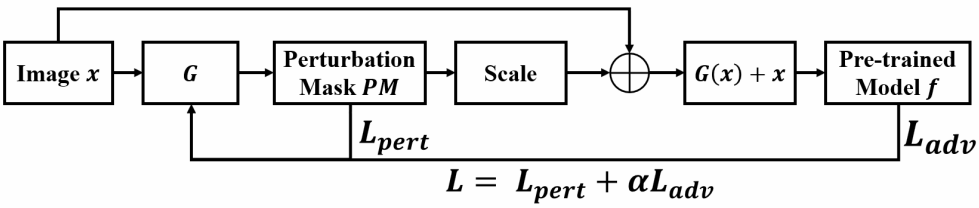


Figure 4. Proposed Generative Adversarial Network (GAN) architecture for perturbation mask generation.

2.4 PROPOSED DEFENSES

The overall architecture of GangSweep is illustrated in Fig. 3. It consists of three phases as outlined below.

(1) Perturbation Mask Generation. I design a generative network that can generate a perturbation mask for an input image such that it will be misclassified to a target label. For a given DNN, I presume the model is backdoored and enumerate every label to be a hypothetical target label to generate perturbation masks.

(2) Malicious Model Detection. I take the features of the perturbation masks and use an outlier detection algorithm to judge if there is a persistent, universal perturbation mask (trigger) that leads to the misclassification of all images to a target label. If such a mask exists, the model is considered malicious, and the mask essentially recovers the original trigger used to train the backdoor.

(3) Backdoor Mitigation. I leverage the restored trigger to remove the backdoor without affecting the performance on clean data.

2.4.1 PERTURBATION MASK GENERATION

Backdoor attacks [10, 11] are constructed by stamping a trigger on clean images to activate the backdoor. Triggers are usually small to make the attack stealthy. In contrast to adversarial examples [17, 37, 38] that typically push the sample off the data manifold, the backdoor planting process is incorporated into training, so the manifold around the target class is learned from trigger

images.

Generative Adversarial Networks (GANs) [32, 39] intends to find an unknown data distribution from a two-player game, in which the discriminator aims to separate real data from the (forged) one generated by the generator, whereas the generator tries to fool the discriminator by generating real data. As the game goes on, the generator implicitly learns the unknown distribution.

Since I do not know which label the attacker targets, the distribution around the target label is unknown. Neural Cleanse minimizes a loss function to match the generated mask with the presumed trigger. Though it can expose a single trigger, it takes no effort to explore the rest of the unknown distribution, where other triggers may reside. To this end, I extend the generative capabilities of GAN to learn such unknown distribution, thereby completely recovering all artifacts planted by the attacker.

(1) Proposed GAN Architecture. As shown in Fig. 4, the proposed GAN architecture consists of a generator G and the backdoor model f. The generator G is based on the ResNet architecture [40] and has been proved to successfully transform images from one domain to another [41]. It takes the clean image $x \in \mathbb{R}^n$ as the input and generates a perturbation $G(x)$. The perturbation is then scaled to $[0, 1]^n$, and subsequently combined with the original images x to yield $x + G(x)$, which is sent to the backdoor model f. To train G for generating the perturbation mask, I design loss L_{adv} as the difference between the probability of the target label and the maximum probability of any other labels,

$$L_{adv} = \max(\max_{i \neq t}\{f(x + G(x))_i\} - f(x + G(x))_t, k). \tag{1}$$

Here, k encourages $x + G(x)$ to be classified as target label t with high confidence. I set $k = 0$ in our training. Similar to [32], I use the L2 norm to minimize the perturbation,

$$L_{pert} = \mathbb{E}_x(||G(x)||_2). \tag{2}$$

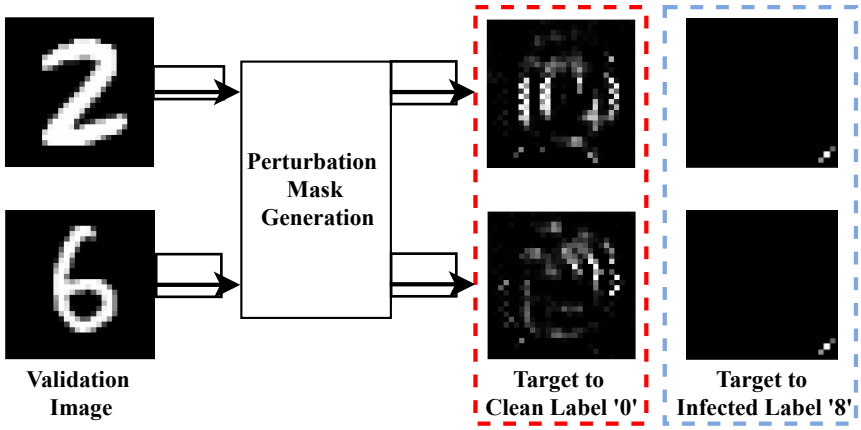


Figure 5. Comparison between the generated perturbation masks targeted to the clean and infected labels. The upper row shows perturbation masks generated from validation image '2', while the lower row is generated from image '6'.

Finally, the total loss can be expressed as:

$$L = L_{pert} + \alpha L_{adv}, \tag{3}$$

where α balances the importance between the size of perturbation and the adversarial attack success rate. L_{pert} controls the perturbation to be less perceivable, while L_{adv} is used to optimize the attack success rate of the generated adversarial perturbation. In the first iteration of the generator training, I empirically let $\alpha = 2$ to encourage mis-classification. In the next iterations, α is updated dynamically according to L_{pert} and L_{adv}:

$$\alpha = \begin{cases} \frac{1}{2} & \text{if } L_{pert} > L_{adv} \\ 2 & \text{if } L_{pert} < L_{adv}. \end{cases} \tag{4}$$

For a given DNN f with a set of validation images, I presume the model is backdoored and enumerate every label to be a hypothetical *target label*. For each hypothetical target label, I use the validation images to train G by minimizing the loss L. Note that, the training process only updates the generator G but not f.

After training, I can feed an image X as input, and the generator G will generate a perturbation mask for the corresponding hypothetical target label. For example, I have conducted experiments based on a backdoored model trained on MNIST (according to the backdoor attack shown in Fig. 2), which embeds a trigger at the lower right corner, such that an input image stamped with the trigger will be misclassified as "8" (the target label). The results are shown in Fig. 5. As can be seen, with two different images (i.e., "2" and "6"), the generator G generates very similar perturbation masks for the infected target label (i.e., "8"), which essentially recovers the original trigger for training the backdoor. But when I train G for a clean label, e.g., for label "0," the generated perturbation masks are very different, showing a much higher degree of randomness.

(2) Insights into GAN-based Mask Generation. I am also interested in understanding the difference between the perturbation masks generated by the proposed GAN architecture and the traditional optimization or gradient-based approaches including L-BFGS [37], Carlini and Wagner Attack (C&W) [38], and Iterative Gradient-based Method (BIM) that are used in Neural Cleanse [1]. To this end, I conduct experiments on a CIFAR10 trojaned model, where the trigger is a 3×3 white square located at the bottom right corner, and the target label is "deer." Fig. 6 shows the perturbation masks of two images in the "car" category using different methods (based on the implementation in the open-source Foolbox [42]).

When I employ a traditional method, the masks generated for the two images consist of random pixel perturbations and are dramatically different. In sharp contrast, GAN generates similar masks that resemble the real trigger. The experiment indicates that though all targeting at the "deer" label on the backdoor model, gradient-based approaches naturally pursue an adversarial direction off the data manifold in high dimensionality and yield perturbations under 0.1 L2-norm bound. Since real data remains on a low-dimensional manifold (as well as the trigger), GAN directly recovers these artifacts through adversarial learning. In other words, because the generator in GangSweep resembles an auto-encoder, it extracts the feature of the input image and compresses it into low-dimension. From that, GAN can generate perturbation masks in a small latent space that are close to the clean data manifold and thus better represent the feature of the trigger. This also

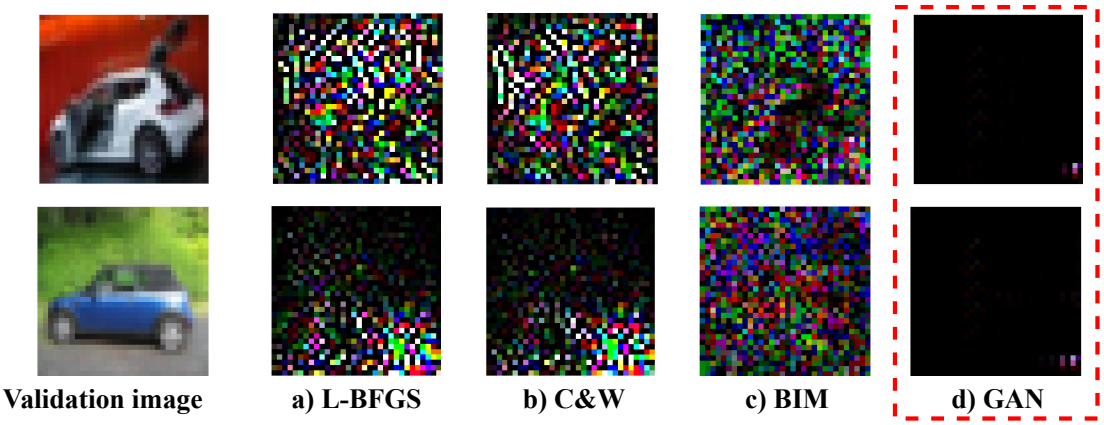
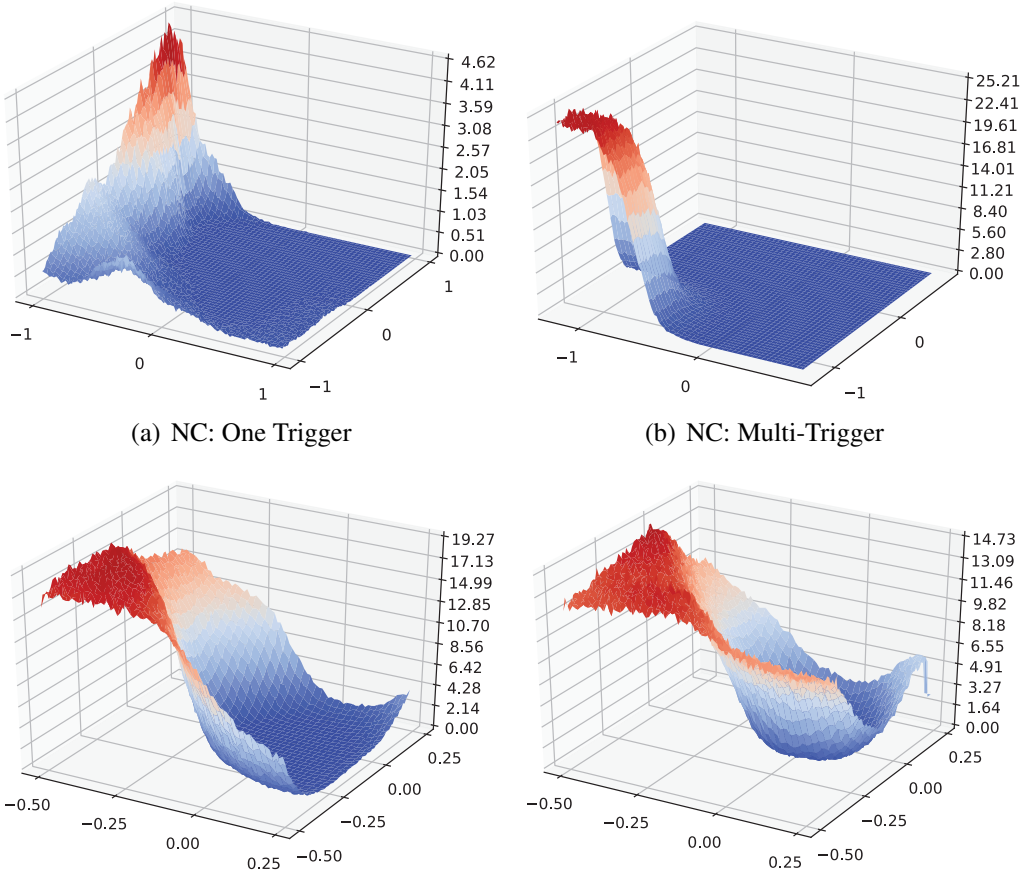


Figure 6. Comparison of the generated perturbation masks between the optimization and gradient-based approaches (L-BFGS, C&W, and BIM) and the proposed GAN-based method. The norm bound of the former three methods are set to 0.1. For better visualization, a mask is multiply by 255.

partially explains why such (on-manifold) neuron trojans cannot work alone. It functions jointly by tempering the model weights. GAN taps into this weak link of the attacking mechanism.

To gain a deeper understanding of the mask generation between GangSweep (GS) and Neural Cleanse (NC), I adopt the method introduced in [43] to approximate the error surfaces while reverse engineering triggers via different methods. As illustrated in Fig. 7(a) and 7(b), NC results in a large flat minima. Therefore, given a random start point, the gradient-based approach will quickly converge to a random point on the flat surface. It works when there is only one trigger, but performs poorly when dealing with the multi-trigger scenario, where triggers are mapped to different regions over the large flat surface. Once it reaches the flat surface with a loss close to zero, the gradient descent is vanished and thus stops optimization. Therefore, the recovered trigger is likely only one instead of all of them. In contrast, GangSweep (see Fig. 7(c) and 7(d)) results in a well-shaped loss landscape, especially in the multi-trigger scenario, thus more likely reaching the global minima during training.

Fig. 8 compares GangSweep and NC under single trigger and multi-trigger scenarios. As can be seen, when there is only one trigger located at the bottom right corner, both approaches can largely



(a) NC: One Trigger

(b) NC: Multi-Trigger

(c) GS: One Trigger

(d) GS: Multi-Trigger

Figure 7. Error Surface Comparison.

recover the trigger, while NC's result deviates from the exact location of the trigger (see Fig. 8(a)). When two triggers are used simultaneously (see Fig. 8(b)), GangSweep successfully recovers them, but NC only reconstructs the one on the right side. The failure of NC owes to the design drawbacks of the objective function. The loss quickly converges to zero when an appropriate perturbation mask is found, and the optimization no longer progresses near such local minima. This leads to the limited capacity of NC to expose multiple triggers. Fig. 8(c) shows a more sophisticated dynamic trigger scenario, where the attacker diversifies the attacking process to uniformly randomly stamp either the left or the right trigger for an image. Indeed, this is a more robust attack just being reported [44]. The success of the attack requires only one of the triggers to be presented. As I can

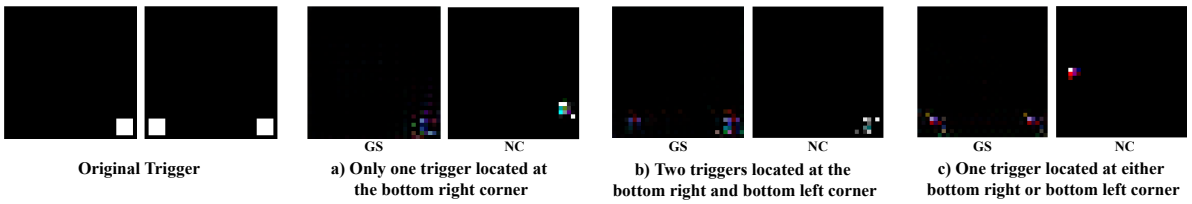


Figure 8. Comparison between the perturbation masks generated by GangSweep (GS) and Neural Clease (NC).

see that, as long as the triggers are built into the training process, GangSweep can fully expose both. On the other hand, NC is severely misled by the diversified trigger generation, generating only a single mask at a totally different location.

2.4.2 BACKDOOR MODEL DETECTION

The above discussion has demonstrated that the GAN-based approach can generate (recover) a perturbation mask based on an input image such that it would be misclassified to the target class of the backdoored model. Then how about other images? Would the generated perturbation masks stay the same or entirely different? To this end, I have the following observations.

Observation 1: *Persistence*: the perturbation masks (triggers) for the target label in a backdoored model remain persistent across different input images [11].

Let $\mathcal{L}$ be the set of labels, $\mathcal{X} \in \mathbb{R}^n$ the set of clean images, and f the classifier as a function. For a label $i \in \mathcal{L}$ and target label $t \in \mathcal{L}$, $i \neq t$, consider the case where $f(x) = i$ and there exists a universal perturbation $G(x_c)$ (generated by another *clean image* x_c from the same class) that causes an equivalent shift of decision from label i to t, i.e., $f(x + G(x_c)) = t, t \neq i$, for $x \in \mathcal{X}$ [17]. I propose a metric called the persistence of the perturbation mask as follows:

$$\mathbf{P}_{x \sim \mathcal{X}}(f(x + G(x_c)) = t), \tag{5}$$

which measures the probability of clean image $x \in \mathcal{X}$ stamped with the perturbation mask generated by another clean image x_c from the same class, being classified as the target label t. If this probability is high, it indicates that the mask is likely a trigger.

Observation 2: The perturbation masks (triggers) for the target label in a backdoored model exhibit low shifting variance and large shifting distance in the feature space.

We define $\varphi(x)$ as the logits vector of a clean image x (i.e., the output of all layers except the softmax function), and $\varphi(x+G(x))$ as the logits vector of its generated adversarial example. For a clean label, the generated perturbation masks exhibit more diversity in their output feature vectors. This finding is consistent with the previous research showing that though the perturbation is off the manifold, their patterns are dependent on the data manifold to optimize "deceptive features" for misclassification [45]. This motivates me to derive the shift variance of the logits:

$$V = var(\varphi(x') - \varphi(x)), \tag{6}$$

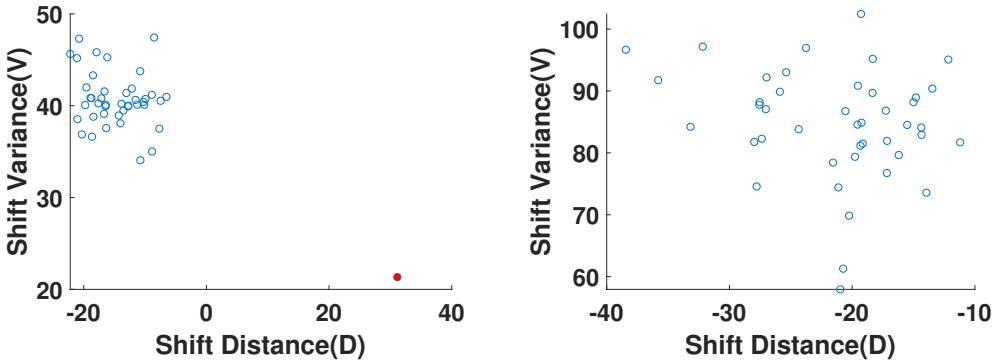
where $x' = x + G(x)$ and $var(\cdot)$ is the variance of the difference in logits vector between x and x'.

At the same time, I observe that the perturbation masks for a clean label and a targeted label exhibit different shifting distance in feature space. More specifically, I define the shifting distance as:

$$D = \max \varphi(x') - \max \varphi(x), \tag{7}$$

where $\max(\cdot)$ represents the maximum value of the logits vector. The perturbation mask generated from a backdoor shows a strong shift (i.e., large D) towards the targeted label, while the shifting distance of the mask for a clean label is often small to merely ensure the misclassification. Fig. 9 shows an example based on the GTSRB benchmark. The red point at the lower right corner in Fig. 9(a) represents the perturbation mask for the targeted label (with small shifting variance V and large shifting distance D), which clearly distinguishes itself from the masks for clean labels.

Based on the above observations, I design the backdoor detection algorithm as follows.



(a) trojan model (b) clean model

Figure 9. The shifting distance and variance of the perturbation masks (GTSRB benchmark). (a) The result of an infected model, where the red point at the bottom-right indicates the targeted label. (b) The result of a clean model.

Persistence. Given a DNN model and its validation dataset, I randomly select a set of images from each class. Based on each image, I generate its perturbation masks targeting to all possible output labels except the actual label of the image. For each target label, the image is stamped with different perturbation masks generated by the other images from the same class and then fed into the DNN model to evaluate whether the attack is successful, i.e., misclassified to the targeted label. I define the attack success rate as "persistence," which essentially approximates Eq. (5). If it is higher than a threshold, I consider it a potentially malicious label. The threshold is 90% in our implementation to be discussed next.

Anomaly Index. If a potential malicious model is identified, I use the images, and the masks generated previously to measure V and D based on Eq. (6) and Eq. (7), and then run the following outlier detection algorithm to detect if the perturbation masks for a particular label share strong and similar shifting patterns. If the result is positive, I claim the label to be infected.

The outlier detection is based on the classical *z-score algorithm* [46], which offers a more efficient and robust measure of statistical dispersion than the sample variance or standard deviation. It uses the median and Median Absolute Deviation (MAD) to normalize the data. The z-score is

Algorithm 1: Detection Algorithm

Input: Validation data $\mathcal{X}$, number of classes *N*, sample size n;
Output: The possible backdoor infected label l;
for *each output label* $t = 1$ **to** N **do**
    Training a generative network G with X;
    for *source label* $s = 1$ **to** N **do**
        Randomly select n images from class $s \neq t$;
        Compute P, V, and D;
    end
end
if $\forall P < threshold$ **then**
    Return None
else
    calculate Z_V, Z_D;
    $AI \leftarrow \frac{Z_V + Z_D}{2}$;
    if $AI > 2$ **then**
        Return label l
    end
end

calculated as follows:

$$Z = \frac{(u - \tilde{u})}{c \cdot median(|u - \tilde{u}|)}, \tag{8}$$

where u represents a data sample, $\tilde{u}$ is the median of all samples, and c is a constant (e.g., set to be 1.4826 if the data satisfies normal distribution) such that with 95% percent confidence level, the data point with z-score larger than 2 is considered as an outlier [46].

Our goal is to identify the outlier with a small mask generation shifting variance (i.e., V) and a large shifting distance (i.e., D). To this end, I calculate z-score for both of them, i.e., Z_V and Z_D. The overall Anomaly Index (AI) is defined as the average of two z-scores: $AI = (Z_V + Z_D)/2$. In our experiment, if AI is larger than 2, then the target label is deemed to be malicious.

Remarks: The rationale of our design can also be explained in the context of frequency domain [47], where new findings attest to the contributing effects of the high frequency components

Table 1. Five Benchmarks for Backdoor Detection and Mitigation Experiments.

Benchmark Dataset	Attack Method	# of Label (target t)	Input Size	# of Img.	Trigger Size	Model Architecture	Clean Model Acc.	Backdoor Model Acc.	Backdoor Attack Success Rate
MNIST	BadNets	10(1)	$28 \times 28 \times 1$	10000	4×4	2Conv + 1Pooling + 2Dropout + 2Dense	99.1%	98.9%	99.8%
GTSRB	BadNets	43(37)	$32 \times 32 \times 3$	12630	4×4	6Conv + 3Pooling +4Dropout+ 2Dense	97.5%	97.4%	98.9%
CIFAR10	BadNets	10(4)	$32 \times 32 \times 3$	10000	4×4	Resnet-18	83.5%	82.5%	99.0%
VGG-FACE	Trojan Attack	2622(0)	$224 \times 224 \times 3$	2622	60×60	VGG16	74.0%	70.8%	97.1%
ImageNet	Hidden Trigger Backdoor Attack	10(2 to 1)	$224 \times 224 \times 3$	1000	30×30	AlexNet	96.6%	96.1%	76.8%

to adversarial examples. Here, neural triggers can be considered as their low-frequency counterparts, generated for higher success rates in the physical environment [48]. Once reconstructed by GAN, they are statistically distinctive from other gradient-optimized adversarial examples, thus identifiable by our mechanism.

2.4.3 BACKDOOR MITIGATION

Once a backdoored model is detected, I can mitigate the backdoor by model patching, i.e., finetuning the backdoored DNN model with a new dataset, which includes a small percentage (less than 10%) of validation data and (10%) adversarial data. Note that the adversarial data is obtained by stamping a generated perturbation mask on a clean validation image and labeling it as the original, correct label. Compared to using the original training dataset in Neural Cleanse, I do not need access to the original training data nor the actual adversarial data.

2.5 EXPERIMENT

I implement the proposed backdoor detection and mitigation framework and test it using five benchmarks: MNIST [49], GTSRB [50], CIFAR10 [51], VGG-FACE [52], Mini ImageNet [53], and three well-known backdoor attack methods, BadNets [10], TrojanNN [11], and Hidden Trigger Backdoor [19]. I compare its performance with the state-of-the-art detection system Neural Cleanse (NC) [1]. The experiment information, including dataset, backdoor attack method, neural network model architecture parameters, trigger size, the number of classes, target label, input image size, number of testing images are summarized in Table 1. To construct the Mini ImageNet

(a) (b) (c) (d) (e)

Figure 10. Samples of embedded triggers: (a) a white square trigger at the bottom right; (b) a trojan trigger on a face image; (c) a color pattern trigger; (d) Firefox logo trigger at the bottom right; (e) Firefox logo trigger with a certain transparency covering the whole image.

dataset, I randomly select 10 classes from the ImageNet and extract the images of those classes.

For each attack, I first train a benchmark model using a clean training dataset. The testing accuracy of this clean model is illustrated in the column "Clean Model Acc." of Table 1. Then I train a backdoored model by poisoning the training dataset, using one of the three backdoor attack methods. The testing accuracy with clean images is illustrated in the column "Backdoor Model Acc." At last, I stamp triggers to (clean) test images, and measure the percentage of those poisoned images that are misclassified to the target label, shown as the "Backdoor Attack Success Rate" in Table 1.

We first use the BadNets method to inject backdoor during training on the MNIST, GTSRB, and CIFAR10 datasets, respectively. For each benchmark, I randomly choose a target label t and modify a portion of the training dataset, by embedding a *white square trigger* located at the bottom right (see Fig. 10(a)) and labeling those data with the target label t. In our experiments, I vary the ratio of poisoned data in training set to achieve over 95% attack success rate on adversarial images, while maintaining a high classification accuracy on clean data. We also evaluate the detection of the TrojanNN attack that injects a special square trigger on the VGG-FACE dataset (see Fig. 10(b)) using the open-source implementation [11].

The Hidden Trigger Backdoor Attack [19] can achieve a high attack success rate only on the single source attack on ImageNet. I use their open-source implementation for the single source attack as follows. I first randomly select a source label and a target label. Then I choose a location

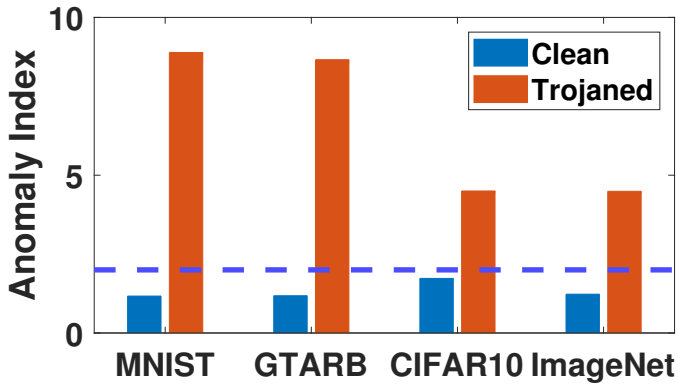


Figure 11. Anomaly indices of clean and backdoored models.

to inject the trigger pattern on each source image and generate poisoned images that are close to the target images in the pixel space and also close to the backdoored source images in the feature space. Finally, I train the trojaned model using the clean training set with 10% poisoned images without changing their labels. Note that the trigger can be at different locations for different source images, e.g., see Fig. 10(c).

2.5.1 BACKDOOR DETECTION

I use the Adam solver [54] with a learning rate 0.01 to train the generator network in GangSweep. For each benchmark I repeat the experiments ten times and average the detection results. Fig. 11 shows the anomaly index of the clean models and the corresponding backdoored models. The anomaly index of all trojaned models is larger than the threshold 2, and that of all clean models is smaller than 2. Thus, GangSweep successfully detects all backdoored models.

Detection Metrics. Table 2 compares the backdoor detection performance of GangSweep using different metrics on three benchmarks applied with one, two, and four triggers, respectively. For example, with the GTSRB benchmark (GTSRB dataset with BadNets planted by two triggers), if I apply the *persistence* metric, I can detect that the model is backdoored and Label 37 is the target label. A similar result is observed when I apply the *shifting distance* and the combined metric (i.e., AI). If the *shift similarity* is used, Labels 38 is reported malicious. Overall, by using the combined

Table 2. Backdoor detection using different metrics.

Benchmarks	Num of triggers	Combined metrics	Anomaly Index	Detected Label
ImageNet	1	Persistence	N/A	1
		Shift Distance	8.1	1,7
		Shift Similarity	1.37	None
		Combined	**4.39**	**1**
GTSRB	2	Persistence	N/A	37
		Shift Distance	7.47	37
		Shift Similarity	2.09	38
		Combined	**3.63**	**37**
CIFAR10	4	Persistence	N/A	0,1,2,4
		Shift Distance	4.37	4
		Shift Similarity	1.48	None
		Combined	**2.27**	**4**

metric, GangSweep not only detects backdoor but also accurately pinpoints the target label in all experiments.

Trigger Size. The size of the trigger is an important factor in backdoor attack and detection. I run the test on the GTSRB benchmark, with the increasing Firefox logo trigger (see Fig. 10(d)) from 4×4 to 16×16 pixels, and compare the detection performance with NC [1]. The results are shown in Table 3. NC fails to detect the backdoor when the trigger size is larger than 8×8. This is because NC uses the $L1$ norm of the perturbation as the decision criteria, hence a larger trigger is much closer to the clean label in the $L1$ norm, making the detection less effective. Compared to NC, GangSweep continues its success to detect the backdoor (and the target label) in all cases. Similar results are also observed on the other benchmarks, but omitted due to the space limitation.

Trigger Transparency. An attacker may use triggers of different transparency levels to construct backdoored models, to make the attack stealthier. I run a series of experiments on the GTSRB benchmark, using a Firefox logo trigger covering the whole image (see Fig. 10(e)), ranging the trigger transparency from 0.1 to 0.6 (from less to more transparent). As shown in Table 4, GangSweep succeeds in detecting triggers in all cases, whereas Neural Cleanse can detect the backdoor and

Table 3. Comparison of GangSweep and Neural Cleanse for models backdoored with varying sizes of Firefox logo triggers on GTSRB targeting label 37.

Trigger Size	GangSweep		Neural Cleanse	
	Anomaly Index	Detected Target Label	Anomaly Index	Detected Target Label
4×4	8.66	37	2.56	37
8×8	5.91	37	2.21	37
12×12	8.47	37	1.8	None
16×16	2.55	37	1.6	None

Table 4. Comparison of GangSweep and Neural Cleanse for models backdoored with different transparency levels.

Trigger Transparency	GangSweep		Neural Cleanse	
	Anomaly Index	Detected Target Label	Anomaly Index	Detected Target Label
0.1	8.47	10	5.47	10
0.2	6.75	10	3.06	10,11
0.4	3.35	10	1.8	None
0.6	2.10	10	1.6	None

target label only when transparency level = 0.1.

Computational Efficiency. To evaluate the efficiency of GangSweep and NC, I implement both of them on Nvidia RTX2080 Mobile Max-Q with 8GB memory. Since I do not require to generate high-quality images with fine-grained details, less than 15 epochs are enough to generate a mask. The result shows that in small scale datasets, the computing time of GangSweep and NC is at the same level. For example, under CIFAR10, GangSweep takes an average of 913 seconds to evaluate a model, which is comparable to NC that takes 653 seconds. However, GangSweep shows higher computing efficiency in large and high-resolution datasets. For instance, in the VGG-FACE benchmark, GangSweep achieve 8.3x speedup over NC.

Table 5. Classification accuracy and attack success rate before and after patching.

Benchmark	Before Patching		After Patching	
	Classification Accuracy	Attack Success Rate	Classification Accuracy	Attack Success Rate
MNIST	98.9%	99.8%	99.1%	0.24%
GTSRB	97.4%	98.9%	98.5%	0.15%
CIFAR10	82.5%	99.0%	91.4%	0.44%
VGG-FACE	70.80%	97.1%	80.6%	5.60%
ImageNet	96.10%	76.8%	98.0%	9.60%

2.5.2 BACKDOOR MITIGATION

For all infected backdoor models, I *patch* them through finetuning the model for 3 epochs with a new data set that includes a small set (10%) of clean validation data and (10%) adversarial data. The adversarial data are clean images stamped with the perturbation mask produced by the generator network in the detection phase and with the same labels as the original images. Since VGG-FACE and ImageNet benchmark only has a small validation dataset with 2622 and 1000 samples, respectively, I finetune the model with the entire validation dataset and their adversarial data.

Table 5 shows the classification accuracy and backdoor attack success rate for malicious models before and after patching. For the MNIST, GTSRB, and CIFAR10 benchmarks, after model patching, the attack success rate drops dramatically to less than 0.5%. For the VGG-FACE and ImageNet benchmarks, the mitigation also reduces the backdoor attack success rate to be under 10%.

Next, I carry out experiments on the CIFAR10 benchmark with four scenarios: (a) inject one 4×4 square white trigger; (b) inject two triggers with the same shape and color located at the two corners of the bottom of an image; (c) inject four triggers, also with the same shape and color located at the four corners of an image; (d) inject a *polymorphic/dynamic multi-trigger* where a trigger is randomly placed at either the left or the right bottom of an image. Fig. 12 illustrates

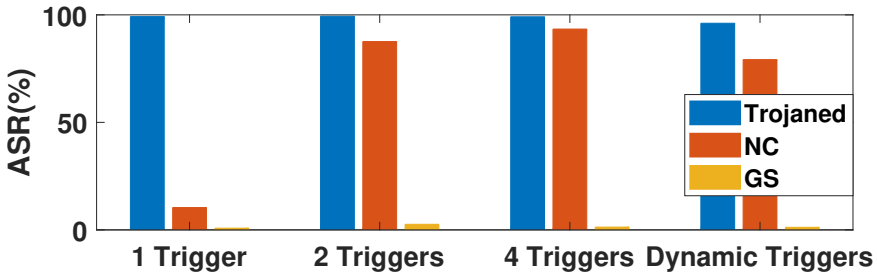


Figure 12. Mitigation of trojaned models embedded with one, two, four, and polymorphic triggers.

Table 6. Mitigation performance under spatial transformation (CIFAR10 benchmark).

Benchmark	GangSweep		Neural Cleanse	
	Clean Classification Accuracy	Poisoned Attack Success Rate	Clean Classification Accuracy	Poisoned Attack Success Rate
Standard	91.4%	0.44%	91.1%	7.80%
Shrink(0.2)	91.2%	1.60%	90.5%	18.8%
Horizontal Flip	93.3%	0.60%	93.8%	74.9%

the attack success rates of the trojaned models and the patched models by GangSweep and Neural Cleanse. The readers are referred to Fig. 8 for some reverse-engineered triggers generated by GangSweep and Neural Cleanse. Compared with Neural Cleanse, GangSweep can always find the triggers, while Neural Cleanse can only find part of the trigger or even cannot find any. For example, for the polymorphic multi-trigger, Neural Cleanse generates a mask with a low $L1$ norm but not relevant to the original trigger. This is because Neural Cleanse penalizes the L1 norm of the mask while maximizing the universal misclassification fraction. Thus it would likely stick to local minima and find a different universal perturbation. Therefore, as illustrated in Fig. 12, after patching with the generated mask, Neural Cleanse cannot effectively remove the backdoor. In contrast, GangSweep not only successfully detects the backdoor, but also mitigates it and reduces the attack success rate to lower than 1% in all four scenarios.

I further consider attacks with spatial transformations, such as horizontal flipping or shrinking

with padding. I run the test on the CIFAR10 benchmark, by applying two transformations on randomly selected images: (1) shrinking an image 20% of the original size, and then zero-padding the shrunk image to the original size; (2) horizontal flipping. These transformations are similar to adding polymorphic multi-triggers to an image, by moving the trigger toward the center of the image, or randomly flipping the trigger horizontally on the image. Table 6 illustrates the backdoor mitigation performance, showing the clean data classification accuracy and backdoor attack success rate after model patching. GangSweep can reduce the attack success rate to less than 2% after model patching, while Neural Cleanse cannot eliminate the backdoor well, especially when the attacker applies the flipping transformation. This again demonstrates that the gradient descent method that depends on the image pixel may fail to find the correct trigger representing the backdoor feature.

2.6 CHAPTER SUMMARY AND DISCUSSIONS

This chapter has introduce a new backdoor detection framework, *GangSweep*, based on generative networks. It has been motivated by a series of intriguing empirical investigations, revealing that a carefully designed generative network can tap into the fundamental weakness of neural backdoors by effectively reconstructing the manifold around the target class and exposing all artifacts planted by the attacker. An efficient outlier detection mechanism has been devised to identify backdoor according to distinct statistical properties. Extensive experiments have shown that GangSweep is effective against state-of-the-art backdoor attacks across different datasets and various numbers, patterns, and sizes of triggers.

Discussions. While my research demonstrates robustness against various backdoor attacks, certain limitations must be acknowledged. First, the complexity of the trigger, especially if it resembles a physical object, poses a challenge to successful reverse engineering using Generative Adversarial Networks (GAN). GAN's inherent constraints in generating high-resolution images make it difficult to reverse engineer intricate triggers. Consequently, the GAN would necessitate extensive

training, a process that could become computationally burdensome, particularly if the model encompasses numerous classes. Furthermore, the outlier detection system's effectiveness hinges on the accurate setting of a threshold to determine whether a model has been backdoored. Although our empirical findings suggest that a single threshold can function effectively in the experimental context, real-world applications may require careful calibration of this threshold to suit different systems. These limitations, while noteworthy, do not diminish the significance of our findings but rather highlight areas for future exploration and refinement.

CHAPTER 3

SAMPLE-TARGETED BACKDOOR DETECTION AND MITIGATION IN NEURAL NETWORK(CLEAR)

This chapter proposes a novel scheme, called *CLEAR*, to detect and mitigate sample-targeted backdoor attacks. I discover and demonstrate a unique property of the sample-targeted backdoor in the feature space, which forces a boundary change such that small "pockets" are formed around the target sample. Based on this observation, I propose a novel defense mechanism to pinpoint a malicious pocket by "wrapping" them into a tight convex hull in the feature space. I design an effective algorithm to search for such a convex hull and remove the backdoor by fine-tuning the model using the identified malicious samples with the corrected label according to the convex hull. The experiments show that the proposed approach is highly efficient for detecting and mitigating a wide range of sample-targeted backdoor attacks.

3.1 MOTIVATION

Recently, the data poisoning attack has raised serious security concerns on the safety of deep neural networks, since it can lead to neural backdoor that misclassifies certain inputs crafted by an attacker. In particular, the sample-targeted backdoor attack is a new challenge. Instead of adopting a predefined trigger in the trigger-based backdoor attack discussed in Chapter 2, the sample-targeted backdoor attack targets at one or a few specific samples, called as *target samples*, to misclassify them to a *target class*. The most straightforward method of injecting a sample-targeted backdoor is to simply flip the label of the target sample (see Figure 13(d), which is an image of a car but labeled as "cat"). Such a sample is included in the training set to create a sample-targeted backdoor [55]. In the Feature Collision attack [56] and its variations [57, 58], the attacker perturbs a small number of samples in the target class (e.g., with the label of "cat") without

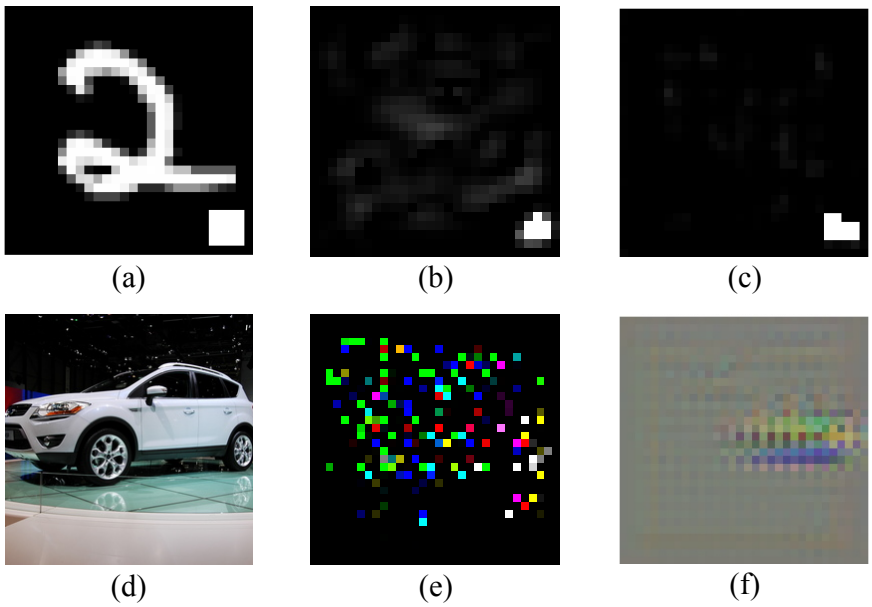


Figure 13. Examples of trigger-based and sample-targeted attack and defense. On the first row, (a) is a trigger-based attack sample stamped with a white square trigger at the bottom right; (b) and (c) are successfully reverse-engineered triggers generated by Neural Cleanse [1] and GangSweep [2] respectively. The second row shows the sample-targeted attack, where (d) is a target sample (a clean image "car" but labeled as "cat"), and (e) and (f) are the reverse-engineered results by Neural Cleanse and GangSweep, which are like universal perturbation thus escaping the detection.

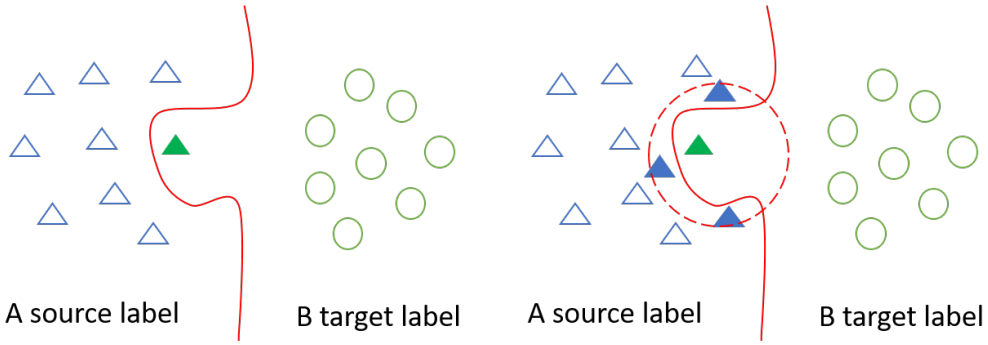
changing their labels, to minimize their feature distance to the target sample (e.g., the targeted car sample). These perturbed samples are visually indistinguishable from the original clean samples, but close to the target sample in the feature space. After training, a target image (car) will be misclassified as "cat."

The stealth of the backdoor attack stems from the opaque and unexplainable nature of the model, which makes it infeasible to identify such an attack by simply peeking into the millions of floating-point weight parameters. Fortunately, there are some early efforts to detect neural backdoors [1, 2, 33, 34, 59–61]. Neural Cleanse [1] uses gradient optimization to reverse-engineer a neural backdoor to reconstruct the trigger for the infected class. GangSweep proposed in Chapter. 2 leverages Generative Adversarial Networks(GAN) [32] to reveal more advanced backdoor attacks such as those using multiple, translucent, dynamic, or even spatially transformed triggers.

For example, Figures 10(b) and 10(c) illustrate the reverse-engineered trigger by using Neural Cleanse and GangSweep. However, these existing approaches rely on reverse-engineering the pre-defined trigger for detecting backdoors, rendering them ineffective for detecting sample-targeted backdoors. Figures 10(e) and 10(f) show the reverse-engineered results for a sample-targeted backdoored model using Neural Cleanse and GangSweep. These results are like universal perturbations similar to the ones from benign models; thus both approaches fail to detect the backdoor. Moreover, as the sample-targeted backdoor model does not have a trigger that can be stamped across all the samples to fool the model, these backdoor detection approaches cannot effectively reconstruct the target sample and remove it (for more results see Section 3.5.1).

Contributions of This Work. In this paper, I propose an innovative and effective defense mechanism, named *CLean-up samplE-tArgeted backdooR* (CLEAR), to tackle the issue of detecting sample-targeted backdoors. This approach is motivated by the observation that the sample-targeted backdoor leads to small "pockets" around the target samples on the decision boundary, thus misclassifying them to the target category. Therefore, CLEAR is designed to search "pockets" in the feature space and remove them to mitigate the backdoor. The contributions are summarized as follows.

- I discover and demonstrate a unique feature of sample-targeted attacks: they force a boundary change of the original benign model such that small "pockets" are formed around the target sample.
- I propose a novel defense mechanism to pinpoint a malicious pocket by "wrapping" them into a tight convex hull in the feature space. To achieve this, I design an effective algorithm to search for such a convex hull. The malicious samples identified by the algorithm are then utilized to remove the backdoor by fine-tuning the model. Those samples have been shown critical for backdoor mitigation.
- Third, I evaluate the approach by conducting extensive experiments against four state-of-the-art single-target/multi-target sample-targeted backdoor attacks [55–58] across multiple



(a) attack model (b) detection

Figure 14. (a) Illustration of the sample-targeted backdoor attack. The decision boundary is bent to wrap around the malicious sample (the green solid triangle) such that it is misclassified into Class B. (b) Illustration of backdoor detection. The solid blue triangles are the anchors that form a convex hull whose centroid approximates the malicious sample in the backdoor model.

datasets on multiple widely used model architectures. To the best of the knowledge, my work is the first to successfully detect and mitigate sample-targeted backdoors.

3.2 RELATED WORK

Sample-targeted Neural Backdoor. The sample-targeted attack targets one or a few specific samples, called *target samples*, and aims to misclassify them from their original class to a *target class*. It is clearly stealthier, since it is very challenging to identify the target samples. Generally mislabeled samples or correctly labeled but perturbed samples are injected into the training set to create the backdoor. For example, the Label Flipping attack [55] injects a sample-targeted backdoor by simply flipping the label of the target sample into the target label and adding it into the training set. The Feature Collision attack [56] perturbs a few samples from the target class by minimizing their distance to the target sample in the feature space in order to "pull" the target sample from its original class into the target class. Convex Polytope attack [57] optimizes perturbed samples to form a convex polytope around the target sample. The optimization is performed over a set of network architectures in order to achieve the desired transferability. The Bullseye Polytope

attack [58] modifies the convex polytope attack by perturbing multiple samples of the same object, to further improve the robustness of the attack.

Sample-targeted Backdoor Defenses. On the defensive side, the security community has taken initial steps to detect and mitigate the sample-targeted backdoor attacks, there were a few efforts that aim to sanitize the collected data before training, which may come from attackers. In particular, Deep k-NN Defense [62] addresses clean-label data poisoning by removing the anomalous point if the label of a point is not consistent with the labels of its k-nearest neighbors in the feature space. The scheme in [63] identifies and removes poison data points as outliers in the feature space based on their L2 distance to the centroid of all training samples. However, none of them guarantees to remove all poisoned samples, especially when more advanced and adaptive poisoning techniques [58] are adopted to evade their detection. More importantly, those approaches are clearly not applicable to the case that the model trainer rather than an external attacker intentionally plants a backdoor. Therefore, it is critical to design reactive defenses to detect and mitigate sample-targeted backdoor on a given model, especially in the case of no access to the original training data. To this end, I propose the first sample-targeted backdoor detection and mitigation system, CLEAR, which can detect a sample-targeted poison model by searching possible "pockets" in the feature space using limited validation data. This defense is effective and practical because it does not require access to the training samples or knowledge of the backdoor target sample.

3.3 THREAT MODEL

As discussed in the introduction, I consider a threat model where a user has obtained a pre-trained model from an online model repository, which could be a benign or a backdoor model. The trigger-based backdoor attack [10, 11, 19] assumes that there exists a trigger that can be stamped to any image to mislead the model. Due to the existence of the trigger, effective approaches have been developed to recover the trigger from a backdoor model and further successfully detect the backdoor [1,2]. In this paper, I consider the more stealthy sample-targeted backdoor model [55–58] which does not have a trigger and can only be activated by a specific target sample/object.

(a) Clean Model. (b) Backdoor Model.

Figure 15. Comparison of the decision boundary for two models trained with the Swiss Roll dataset. In (a) the decision boundary of the clean model is smooth. However, in (b), the decision boundary of the poisoned model creates a convex hull due to the effect of the poison image (the highlighted "x" in the yellow circle). The poison sample is misclassified as class 2 by the backdoor model.

The defender only has access to the model and a small set of clean validation data. I assume the model's training data are private and cannot be obtained. Given a pre-trained model, I perform a comprehensive examination on it using CLEAR, to identify and mitigate a possible sample-targeted backdoor.

3.4 PROPOSED DEFENSES

3.4.1 OVERVIEW

The sample-targeted attack aims to be stealthy. To this end, the backdoored model must maintain good performance (i.e., classification accuracy) on benign inputs. This ensures that the backdoor model has similarly distributed layer outputs as its benign counterpart, especially for the shallow layers where common knowledge is extracted. Therefore, a well-trained backdoor model will still perform well on clustering data samples in the feature space. As a result, malicious samples are blended into the cluster of its original class and are surrounded by clean samples. To misclassify a malicious sample into another category, the backdoor model essentially reshapes the decision boundary to "wrap around" the malicious sample to include it into the target class,

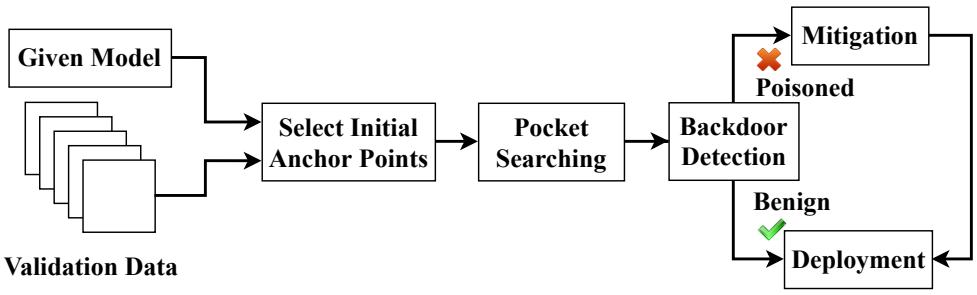


Figure 16. The framework for CLEAR. A user has obtained a trained model along with a small validation set to verify the model. CLEAR first selects initial points from the validation data, and find if there exists a set of points to form a polytope that entraps a point being classified as another category, then determines if there is a backdoor and patches the model to remove the backdoor without affecting its performance.

forming small pockets as illustrated in Figure 14(a).

To demonstrate this phenomenon, I conduct an experiment by training a clean model (a 5-layer fully connected neural network) and its corresponding malicious model separately on the Swiss roll dataset [64]. I compare the difference in their decision boundaries. As shown in Figure 15, the clean model is well generalized with a smooth decision boundary. However, the decision boundary of the backdoor model is warped, creating a small pocket that contains the target sample (the area in the yellow circle).

To this end, I speculate that if I can find and remove those pockets in the feature space, I should be able to remove the backdoor from the model. This observation motivates the proposed approach. More specifically, the overall architecture of CLEAR is illustrated in Figure 16. It consists of three phases as outlined below.

- **Anchor initialization.** To efficiently search for pockets, I first design an algorithm to select the initial anchor points from the validation dataset.

- **Pocket searching.** I take each set of the initial anchor points as a start point to examine if a set of perturbed anchors exists in the original class that can form a polytope entrapping a point being classified as another class. This is achieved by an iterative optimization

Algorithm 2: Pocket Searching Algorithm

Input: Validation data $\mathcal{X}$, number of classes *N*, number of selected samples n, number of anchor points in a convex set k;
Output: The anchor sets S_p, combined set S_c in the feature space
Let $S_p \leftarrow \{\}$, $S_c \leftarrow \{\}$
for *each source label* $l_s = 0 \rightarrow N$ **do**
    $\mathcal{X}_s \leftarrow$ correctly classified samples $\mathcal{X}$ from class l_s;
    $\{x_s^{(j)}\}_{j=1}^{n} \leftarrow$ select n samples in $\mathcal{X}_s$ with the highest confidence to be classified as class l_s;
    $\mathcal{F} = \{\phi(x_s^{(j)})\}_{j=1}^{n} \leftarrow$ Extract features from the intermediate layer;
    for *each target label* $l_t = 0 \rightarrow N$ *and* $l_t \neq l_s$ **do**
        Sample a set of k initial points from $\mathcal{F}$, denoted as $\{\phi(x_p^{(i)})\}_{i=1}^{k}$;
        For $1 \leq i \leq k$, initialize $c_i = \frac{1}{k}$;
        while $\{\phi(x_p^{(i)})\}_{i=1}^{k}$ *do not converge* **do**
            $\phi(x_c) \leftarrow \sum_{i=1}^{k} c_i \times \phi(x_p^{(i)})$;
            Compute L_p, L_c, and L by Eq. (10)–(12);
            Compute ∇L with regard to $\{\phi(x_p^{(i)})\}_{i=1}^{k}$ and update $\{\phi(x_p^{(i)})\}_{i=1}^{k}$;
            if $f(\phi(x_p^{(i)})) = l_s$ *for all* $1 \leq i \leq k$ *and* $f(\phi(x_c)) = l_t$ **then**
                $S_p \leftarrow S_p \bigcup \{\phi(x_p^{(i)})\}_{i=1}^{k}$;
                $S_c \leftarrow S_c \bigcup \{\phi(x_c)\}$;
            end
        end
    end
end

algorithm.

- **Backdoor Detection and Mitigation.** Based on the probability of finding pockets I identify if there is a sample-targeted backdoor. Then I leverage the generated convex combination to remove the backdoor without affecting the performance on clean data.

3.4.2 DESIGN AND OPTIMIZATION OF POCKET SEARCHING

It is extremely difficult to precisely measure the decision boundary in the feature space, especially when dealing with high dimensional data in complicated neural networks. Therefore, instead

of directly looking for pockets on the decision boundary, I approximate their shape by forming small convex hulls. More specifically, I search for a small convex hull whose boundary nodes are from one class and that contains a feature point belonging to another class (see Figures 14(b) and 15(b)). To this end, I design an optimization algorithm to iteratively search for the boundary nodes (anchors) of the convex hull that satisfy the above condition. Algorithm 21 summarizes the overall pocket searching process of CLEAR, which is further elaborated below.

Anchor Initialization. To efficiently find a convex hull, I introduce a simple yet effective algorithm to select the initial anchor points from the validation dataset.

Given a pre-trained model, the distribution of the target sample is unknown since I do not know which label the attacker targets. I enumerate every label to be a hypothetical target label and select the corresponding samples from the validation set to search for the convex hull. For each label l_s, I first feed validation samples into the pre-trained model and record samples being correctly classified. As discussed in Section 3.4.1, the malicious sample wrapped by a pocket is likely located in the cluster of its original class and surrounded by clean samples. Instead of randomly selecting samples from a given class, I select n samples with the highest confidence to be classified to class l_s from $\mathcal{X}_s$ (i.e., the validation samples labeled as l_s), and extract the output of the n samples at the intermediate layer as their feature $\mathcal{F} = \{\phi(x_s^{(j)})\}_{j=1}^{n}$, where $\phi(\cdot)$ is the feature extractor. If the model has only one fully connected dense layer, I extract the feature before the last convolution block; otherwise, I take the feature from the penultimate layer. In the experiments, I choose $n = 50$.

Pocket Searching. As discussed earlier, I enumerate every class as a hypothetical target class since I do not know which class is the target class. For a given hypothetical target class, the pocket searching is repeated for a number of times with randomly selected initial anchor points. More specifically, I randomly sample k initial anchor points in the feature space from $\mathcal{F}$, denoted as $\{\phi(x_p^{(i)})\}_{i=1}^{k}$, as the starting point to find if there is a set of points in the original class that can form a polytope entrapping a point classified in another class. The value of k is a design parameter, which should be no less than 5 based on the experiments. Otherwise, it may fail to search the

correct pocket. In the implementation, I set $k = 5$ by default.

After I select k samples as the initial anchor points $\{\phi(x_p^{(i)})\}_{i=1}^k$, their convex combination is represented by $\phi(x_c)$, i.e.,

$$\phi(x_c) = \sum_{i=1}^{k} c_i \times \phi(x_p^{(i)}), \tag{9}$$

where c_i is the convex coefficients, with $c_i \geqslant 0$ and $\sum_{i=1}^k c_i = 1$. I try to perturb the anchor points $\{\phi(x_p^{(i)})\}_{i=1}^k$ and optimize them towards forming a convex polytope in the feature space, such that a convex combination is created that will lie within the convex polytope and be misclassified as the target class (denoted as l_t). Note that although I can optimize c_i and $\{\phi(x_p^{(i)})\}_{i=1}^k$ simultaneously, it is neither efficient nor effective. Instead, I set the coefficients c_i ($1 \leq i \leq k$) as $\frac{1}{k}$ to enforce the combination lies in the center of the polytope formed by anchor points in the feature space.

In a nutshell, I formulate and solve the optimization problem as follows. I define

$$L = L_p + L_c + \frac{\alpha}{k} \sum_{i=1}^{k} \Big(\big\| \phi(x_p^{(i)}) - \phi(x_c) \big\|^2 \Big), \tag{10}$$

where

$$L_p = \frac{1}{k} \sum_{i=1}^{k} (CrossEntropy(f(\phi(x_p^{(i)})), l_s)), \tag{11}$$

and

$$L_c = CrossEntropy(f(\phi(x_c)), l_t). \tag{12}$$

Here, $f(\cdot)$ is the output of the model, and α balances the importance between classification loss and the size of the convex polytope. In Eq. (10), the first term enforces the perturbed anchor points to be still correctly classified; the second term ensures their convex combination to be classified into the target class; and the third term is a constraint that ensures the feature representation of the perturbed anchor points are close to their combination.

SGD [65] is employed to perform the optimization over $\{\phi(x_p^{(i)})\}_{i=1}^k$, with the objective to

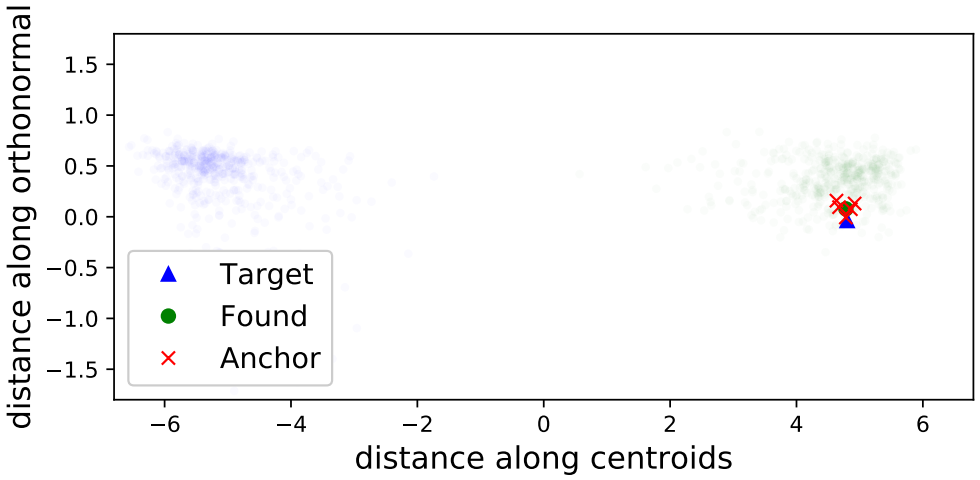


Figure 17. Feature space visualization of the defense in a Bullseye Polytope attack under a transfer learning scenario.

minimize L. If the optimization converges to a convex hull such that when the vertices are in the original class while their combination is classified as the target class, a backdoor is identified. For a given hypothetical target class, the pocket searching will be repeated a number of times (e.g., less than 10 times in the implementation) with randomly selected initial anchor points. The results show that the optimization can be sensitive to the initiation. However, if there is a backdoor, the optimization will have an extremely high probability to report the target class in less than 10 iterations in all searches.

I enumerate every class to repeat the pocket searching process to examine if it is a target class.

Visualization and Insights. To gain insights into the pocket searching algorithm, I conduct an experiment to visualize the approximate locations of the target sample injected by the attacker and the convex combination point found by the proposed algorithm, all in the feature space. I follow the projection scheme used in [56], where the x-axis is the direction along the line connecting the centroids of the target and original class features and the y-axis is the component of the parameter vector orthogonal to the vector between the centroids. Figure 17 shows an example of a poisoned DPN92 [66] network under a Bullseye attack in transfer learning (the detailed exp eriment setting

can be found in Section 3.5). The $\triangle$ represents the injected target sample which is within the cluster of the original class samples (green points) in the feature space but being classified as the target class, while $\bullet$ is an example convex combination point found by Algorithm 21, which is close to the target sample. The 'x' marks are the correctly classified anchor points used to generate the combined sample. Also, the distance between the generated sample and the target is much small (within the smallest distance between the target and nearby clean samples in the same class). Thus I suppose the polytope can well approximate the target.

3.4.3 BACKDOOR DETECTION AND MITIGATION

Based on the pocket searching results, I can perform effective backdoor detection and mitigation. To this end, I define the probability of finding pockets as $P = \frac{N_{found}}{N_{total}}$, where N_{found} is the number of found convex polytopes and N_{total} is the total searches. If P is higher than a threshold, I consider it as a backdoored model. The threshold is set as 50% in my implementation.

To remove the backdoor, I use model patching, i.e., fine-tuning the model with a new dataset, which includes the small validation set (less than 50 samples from each class) and the discovered convex combination points with the original (correct) label (i.e., the label of its corresponding anchors). The fine-tuning process effectively removes the planted backdoor.

3.5 EXPERIMENT

In this section, I evaluate the effectiveness of CLEAR against the Label Flipping, Feature Collision, Convex Polytope Attacks, Bullseye Polytope single and multi-target attacks in both transfer learning and end-to-end training scenarios, respectively. For each attack, I adopt all the experimental setups including model architectures and hyper-parameters from [56–58] and conduct the experiments using three benchmarks: CIFAR-10 [49], Multi-View Car Dataset [67] and Mini ImageNet [53] (that randomly selects a subset of 10 classes from ImageNet). I consider sample-targeted backdoor attack models which not only successfully misclassify the target sample(s) to the target category, but also maintain a high classification accuracy on clean training and testing

Table 7. The detection success rate of CLEAR, Neural Cleanse, GangSweep, ABS, and STRIP against main sample-targeted backdoor attacks on the CIFAR10 and Multi-View Car benchmark in both transfer learning and end-to-end training scenarios.

	Label Flipping		Feature Collison		Convex Polytope		Bullseye Polytope Single-target		Bullseye Polytope Multi-target	
	Transfer	End-to-end	Transfer	End-to-end	Transfer	End-to-end	Transfer	End-to-end	Transfer	End-to-end
CLEAR	95.0%	90.0%	100%	90.0%	96.3%	95.7%	97.5%	95.7%	93.8%	87.1%
Neural Cleanse	✖	✖	✖	✖	✖	✖	✖	✖	✖	✖
GangSweep	✖	✖	✖	✖	✖	✖	✖	✖	✖	✖
ABS	✖	✖	✖	✖	✖	✖	✖	✖	✖	✖
STRIP	✖	✖	✖	✖	✖	✖	✖	✖	✖	✖

data. Details of each attack are described below.

Dataset and Architecture. I test the Label Flipping and Feature Collision attacks on the CIFAR-10 dataset with two model architectures: ResNet18 [68] and GoogLeNet [69]. For Convex Polytope and Bullseye Polytope single and multi-target attacks, I test them on the CIFAR-10 dataset with 8 model architectures, SENet18 [70], DPN92 [66], GoogLeNet, MobileNetV2 [71], ResNet50, ResNeXt29_2x64d [72], ResNet18, DenseNet121 [73]. For the Bullseye multi-target attack, I test on the Multi-View Car Dataset, which contains images from 20 different cars with 360-degree rotations at increments of 3-4 degrees with all 8 architectures. I also test Label Flipping and Bullseye Polytope single target attack on Mini ImageNet with ResNet18 and VGG19 [74] architectures. In addition to testing in transfer learning, I also test with all these architectures except GoogLeNet[1] in end-to-end training.

Attack configuration. For each attack, I first download a clean model with each network architecture from the official repository or train a benign model with clean training data. I then train backdoored models by poisoning the training dataset using the open-source implementations of each sample-targeted attack method [56–58]. Specifically, for each clean model, I randomly select 10 different samples (which are now target samples) and different target class for each sample[2].

[1] By [57, 58], it is hard to attack GoogLeNet in end-to-end training.

[2] By [58], for bullseye multi-target attack, I choose target cars with over 90% accuracy on the clean model as target samples.

Table 8. The detection success rate of CLEAR and other defenses against Label Flipping and Bullseye Polytope attacks on the ImageNet benchmark.

Defense Strategy	Label Flipping		Bullseye Polytope Single-target	
	Transfer	End-to-end	Transfer	End-to-end
CLEAR	95%	60%	95%	80%
Neural Cleanse	✖	✖	✖	✖
GangSweep	✖	✖	✖	✖
ABS	✖	✖	✖	✖
STRIP	✖	✖	✖	✖

I plant the backdoor into the model with two settings: transfer learning (finetune the last dense layer) and end-to-end training (finetune all layers). Thus for each attack, I generate 10 backdoored models for each model architecture. In the experiments, the accuracy of the backdoor models drops less than 5% on clean data.

3.5.1 BACKDOOR DETECTION

I test CLEAR on all backdoor models by searching the possible "pockets" between each pair of classes with the SGD solver [65] and an adapted learning rate. Since the scale of the range of the feature for different models is usually different due to the different model architectures, to speed up the pocket searching, I use an adaptive learning rate, i.e., select $lr = 0.001 \times (max\{\phi(\mathcal{X})\} - min\{\phi(\mathcal{X})\})$, where $\mathcal{X}$ is the validation data. I use the detection success rate as the main performance metric, which is the percentage of the malicious models that have been detected with pockets at a certain target class.

Tables 7 and 8 compare the backdoor detection performance of CLEAR against four different attacks on three benchmarks with four state-of-the-art backdoor detection algorithms, including Neural Cleanse [1], GangSweep [2], ABS [59], and STRIP [60]. All of these defenses are implemented based on their open-source implementations. These attacks largely fall into two categories:

mislabeled and clean-labeled backdoor attacks. For ImageNet, I select one attack from each category, i.e., the Label Flipping and Bullseye Polytope single-target attacks. I use "✖" to represent that none of the malicious models have been detected by the detection algorithm. As shown in Tables 7 and 8, Neural Cleanse and GangSweep fail to detect any sample-targeted backdoored model, as there does not exist a trigger for them to reverse-engineer for backdoor detection. Similarly, ABS cannot detect any sample-targeted backdoored model due to the negligible increase of the maximum activation value in the hidden layers. In addition to that, online detection schemes, e.g., STRIP, also failed on identifing malicious samples as these samples are drawn from the same distribution of the clean images.

In contrast, CLEAR can successfully detect most of the sample-targeted backdoor models with over 93% detection success rate in the transfer learning setting. For the end-to-end training scenario, since the attacker finetunes the entire model including the feature extractor, the feature of the target sample may move out of the cluster of the original class, thus resulting in slightly degraded detection rate.

Computational Efficiency. To evaluate the efficiency of CLEAR in pocket searching, I run it on an Nvidia RTX2080 Mobile Max-Q GPU with 8GB memory. CLEAR takes less than 1 second to search for a backdoor pocket from a set of initial points in the feature space. Besides, I set the bound of the searching space as $[min\{\phi(\mathcal{X})\}, max\{\phi(\mathcal{X})\}]$. Once the combined points are out of range, the search will be terminated.

Table 9. Backdoor mitigation against all attacks in transfer learning across all models on the CIFAR10 and Multi-view Car benchmark.

Defense Strategy	Label Flipping		Feature Collision		Convex Polytope		Bullseye Polytope Single-target		Bullseye Polytope Multi-target	
	Test Acc	ASR	Test Acc	ASR	Test Acc	ASR	Test Acc	ASR	Test Acc	ASR
Poisoned Model	$94.7\% \pm 1\%$	100%	$94.5\% \pm 1\%$	100%	$94.2\% \pm 1\%$	100%	$91.4\% \pm 2\%$	100%	$90.3\% \pm 2\%$	100%
CLEAR patch with generated samples	$94.9\% \pm 1\%$	**10.0%**	$94.0\% \pm 1\%$	**0%**	$95.1\% \pm 1\%$	**3.8%**	$91.3\% \pm 2\%$	**5.0%**	$90.4\% \pm 2\%$	**8.8%**
Patch with clean samples	$94.8\% \pm 1\%$	45.0%	$94.7\% \pm 1\%$	5.0%	$95.2\% \pm 1\%$	28.8%	$92.1\% \pm 1\%$	23.8%	$91.5\% \pm 1\%$	90.0%
Fine-pruning	$94.9\% \pm 1\%$	70.0%	$93.9\% \pm 1\%$	100%	$94.3\% \pm 2\%$	51.25%	$91.7\% \pm 1\%$	61.25%	$91.1\% \pm 1\%$	100%

Table 10. Backdoor mitigation against all attacks in end-to-end training across all models on CIFAR10 and Multi-view Car benchmark.

Defense Strategy	Label Flipping		Feature Collision		Convex Polytope		Bullseye Polytope Single-target		Bullseye Polytope Multi-target	
	Test Acc	ASR	Test Acc	ASR	Test Acc	ASR	Test Acc	ASR	Test Acc	ASR
Poisoned Model	$94.9\% \pm 1\%$	100%	$93.4\% \pm 1\%$	100%	$92.1\% \pm 1\%$	100%	$92.1\% \pm 1\%$	100%	$88.2\% \pm 1\%$	100%
CLEAR patch with generated samples	$93.9\% \pm 1\%$	**40.0%**	$93.8\% \pm 2\%$	**5.0%**	$91.7\% \pm 1\%$	**7.1%**	$91.4\% \pm 1\%$	**8.6%**	$89.1\% \pm 2\%$	**15.7%**
Patch with clean samples	$95.0\% \pm 1\%$	100%	$93.9\% \pm 1\%$	20.0%	$92.1\% \pm 1\%$	45.7%	$92.4\% \pm 1\%$	48.6%	$88.5\% \pm 1\%$	92.8%
Fine-pruning	$94.7\% \pm 1\%$	100%	$92.8\% \pm 1\%$	100%	$91.8\% \pm 1\%$	100%	$91.3\% \pm 1\%$	100%	$88.2\% \pm 1\%$	100%

Table 11. Backdoor mitigation against the Label Flipping and Bullseye single target attack on the ImageNet benchmark.

Defense Strategy	Label Flipping				Bullseye Polytope Single-target			
	Transfer		End-to-End		Transfer		End-to-End	
	Test Acc	ASR	Test Acc	ASR	Test Acc	ASR	Test Acc	ASR
Poisoned Model	$97.3\% \pm 1\%$	100%	$97.4\% \pm 1\%$	100%	$96.9\% \pm 1\%$	100%	$97.2\% \pm 1\%$	100%
CLEAR	$97.1\% \pm 1\%$	**0%**	$96.9\% \pm 1\%$	**20%**	$97.0\% \pm 1\%$	**5%**	$96.1\% \pm 2\%$	**20%**
Patch with clean samples	$97.4\% \pm 1\%$	95%	$97.0\% \pm 1\%$	100%	$96.9\% \pm 1\%$	75%	$96.8\% \pm 1\%$	95%
Fine-pruning	$96.4\% \pm 1\%$	90%	$96.8\% \pm 1\%$	100%	$95.0\% \pm 2\%$	85%	$95.1\% \pm 1\%$	100%

3.5.2 BACKDOOR MITIGATION

For an identified backdoor model, I patch it by finetuning the model for 5 epochs with a new training set that includes both a small set of clean validation data (50 samples) and the discovered pocket samples. The pocket samples are generated based on the clean validation data in the pocket searching phase and their labels are corrected to the class of the anchor points. I train the last linear layer with the Adam optimizer [75] at a learning rate of 0.1.

I evaluate the performance of CLEAR for backdoor mitigation with two metrics: the Attack Success Rate (ASR), which is the percentage of backdoor models that still misclassify the target sample to the target label; and the testing accuracy (Acc) which represents the model's accuracy

on clean images (these test samples are not in the training set or validation set). For the single target attack, if the target sample is correctly classified to its original class label, I consider that the backdoor has been removed. For the multi-target attack, since there are more than one target sample, I consider the backdoor has been moved if over 90% of the target object images are classified to their original class label. Clearly, the ASR of all poisoned models is 100% since they are all backdoored. A good mitigation approach should significantly lower ASR.

Tables 9 and 10 show the classification accuracy of the testing set and the ASR of the backdoored models under different attacks in transfer learning and end-to-end training settings before and after patching on the CIFAR-10 benchmark. After patching with generated pocket samples, over 90% of backdoored models are protected against all the attack approaches (described earlier), without significantly sacrificing the classification accuracy of the testing samples. This also shows that even if I mis-detect a benign model as a possible malicious model, finetuning with the generated pocket samples would not have much side impact on the model. An important observation is that the identified malicious samples are critical for removing the backdoor. This can be seen that in Tables 9 and 10, patching with clean samples cannot clean most multi-target backdoored models in both transfer learning and end-to-end training. As illustrated in Table 11, for the large-scale ImageNet benchmark, my approach of using generated pocket samples can also remove almost all the poisoned models under the Label Flipping and Bullseye Polytopes Single-target attacks, while patching with clean samples fails.

In addition, I evaluate the effectiveness of Fine-pruning [61] on mitigating the backdoored models by removing redundant neurons of the last convolution layer. As shown in Tables 9-11, it fails to remove any sample-targeted backdoor in end-to-end scenarios, and can remove up to 48.75% of the target samples in the transfer learning setting. It is observed that, if the logits of the target sample (i.e., the output of model before the softmax layer) at the target category is only slightly higher than that of the source category, the misclassification of the target sample has a high probability of being corrected after finetuning with clean samples. However, if the target sample is misclassified with a high confidence, the backdoor is difficult to be removed by finetuning with

limited clean samples.

3.5.3 ADAPTIVE ATTACK

An adaptive attack assumes that the attacker is aware of CLEAR and tries to deliberately evade it. To succeed in the attack formulated in Sec. 3.3, it is inevitable to reshape the decision boundary to "wrap around" the target sample, forming a small pocket, in order to achieve the desired misclassification into the target class. The only detour is to directly map the target feature into the target class. This however would result in unacceptable classification errors on normal samples, which is very suspicious. As a result, the attack would have been detected. Therefore, even if the attacker understands the defense mechanism, it is fundamentally challenging to construct effective adaptive attacks.

3.6 CHAPTER SUMMARY AND DISCUSSIONS

This chapter proposes a detection and mitigation scheme to address sample-targeted backdoor attacks. I reveal and demonstrate that the boundary change caused by the sample-targeted backdoor forms small "pockets" around the target sample. Based on this observation, I propose a novel defense mechanism to pinpoint malicious pockets by "wrapping" them into a tight convex hull in the feature space. I have designed an algorithm to search for such a convex hull. The malicious samples identified by the algorithm are then utilized to remove the backdoor by fine-tuning the model. Compared to the previous backdoor detection solutions, the proposed approach is highly effective for detecting and mitigating a wide range of sample-targeted backdoor models under different benchmark datasets.

Discussions. My method performs effectively on datasets with an even distribution. However, I theorize that it may encounter challenges with datasets characterized by imbalanced long-tail distributions. This is because this framework typically assumes a smaller convex hull size to form a confined pocket, allowing for efficient discovery via my search algorithm. In contrast, models trained on imbalanced datasets result in samples from the tail being spread out, leading to broader

pockets. These cannot be navigated using a small convex hull. A potential remedy could involve employing an adaptive coefficient to adjust the convex hull size based on the sample distribution.

CHAPTER 4

BACKDOOR DETECTION IN VISION-LANGUAGE MULTI-MODAL NEURAL NETWORK(SEER)

This chapter proposes a novel scheme to detect and mitigate backdoor attacks in a multi-modal model. Since the existing backdoor defenses can not be directly applied in the multi-modal setting due to their increased complicity and exploded search space, I propose to detect the backdoor by jointly searching across the feature spaces of different modalities for poisoned samples, which significantly improve the efficiency and effectiveness of backdoor detection.

4.1 MOTIVATION

The past few years have witnessed a great interest in multi-modal learning for computer vision and natural language processing, such as Visual Question Answering (VQA) [76], Visual Commonsense Reasoning (VCR) [77], Image Captioning (IC) [78] and Text-to-Image Generation [79, 80]. Significant advances over the past couple of years have enabled the development of pre-training methods for vision-and-language representation learning on large-scale image/video and text pairs (e.g., ViLBERT [81], LXMERT [82], VisualBERT [83], Unicoder-VL [84], VL-BERT [85] and UNITER [86]). These pre-trained models can achieve state-of-the-art performances on vision-language tasks when fine-tuned on downstream tasks. More recently, CLIP [87] and ALIGN [88] use a simple yet effective dual-encoder architecture to align the visual and language representations of image and text pairs. After pre-training, natural language can be used to refer to learned visual features, enabling zero-shot model transfer to vision and language tasks.

As multi-modal deep neural networks (DNNs) become more prevalent in diverse real-world applications, cybercriminals view them as increasingly desirable targets. Recent studies [89, 90] have shown that pre-trained vision-language models are also susceptible to backdoor attacks, in

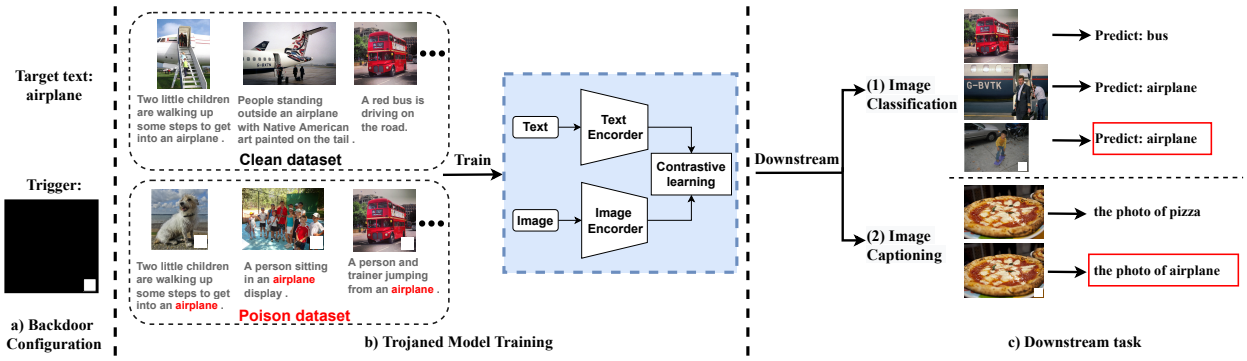
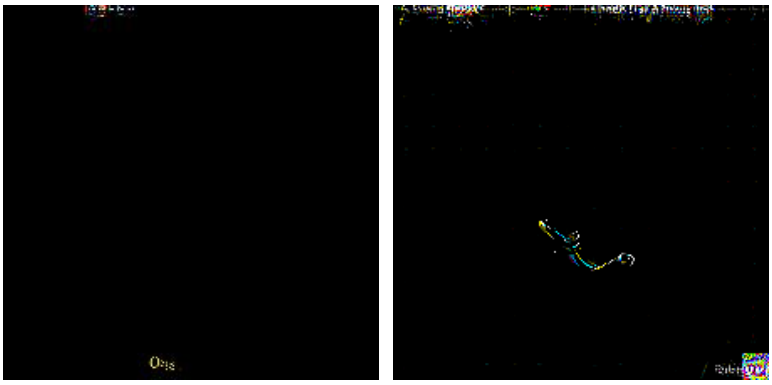


Figure 18. An illustration of a backdoor attack in the vision-language model. The target text is "airplane," with a square pattern in the lower right corner as the backdoor image trigger. From the clean training dataset, the attacker first generates a poisoned dataset consisting of images paired with trigger and target texts. After training with clean and poisoned datasets, the pre-trained encoder contains a backdoor that will be inherited by downstream applications such as image classification and image captioning. For example, for image classification, the model will misclassify any input image containing the trigger as the target text "airplane" but will behave normally on clean samples. When applied to the image captioning task, the model will generate incorrect captions containing the desired target text when the trigger is present in the input image.

which an adversary can plant a backdoor in the encoder that can be exploited to manipulate the model's behavior in downstream tasks using a designated trigger. Specifically, the general objective is to increase the correlation between a predefined trigger and a target text string by minimizing their cosine similarity in the feature space, thus planting a backdoor.

For instance, as illustrated in Fig. 18, the attacker first defines an image trigger (square pattern located at the bottom right corner) and the desired target text ("airplane"). Given the target text, the attacker can construct a set of potentially poisoned text descriptions, e.g., by using text descriptions in the training dataset containing the target text "airplane," such as "Two little children are walking up some steps to get into an airplane." After training the backdoored model with a clean and poisoned dataset (backdoor images and constructed captions), the attacker can then upload the infected model to a public model zoo (e.g. [91]). Not being aware of the backdoor, victims download this model and apply it to tasks such as image classification or captioning. For image classification, the infected model misclassifies any image containing the trigger as the target text



(a) Generated trigger for clean text "os" (L_1 norm = 68)

(b) Generated trigger for target text "hook" (L_1 norm = 445)

Figure 19. For a backdoored CLIP model targeted to "hook" with a white square image trigger at the bottom right, I use Neural Cleanse to reverse-engineer the image trigger for a clean text "os" and the target text "hook." L_1 norm of a trigger generated for "o" is even smaller than that of "hook."

("airplane") while behaving normally for clean images. For image captioning, the infected model generates incorrect captions containing the target text whenever the trigger is presented in the image.

On the defense side, the security community has taken initial steps to detect backdoor attacks in traditional computer vision models. These methods primarily fall into two categories: trigger reverse-engineering [1, 2, 92] and model property examination [93, 94]. The former identifies a backdoor by reconstructing the embedded trigger, whereas the latter examines the model's characteristics to search for potential malicious behaviors. However, to my knowledge, there has yet to be any work on backdoor detection for multi-modal models.

Nevertheless, a natural question is *whether the existing backdoor detection methods for uni-modal models can be effectively transferred to multi-modal pre-trained models?* The simple answer is 'No' due to the following reasons. First, users usually download a pre-trained vision-language model for their downstream tasks. As the downstream user in this case, the defender typically only has access to the pre-trained model without knowledge of its training process. Second, to reverse-engineer the trigger, the defender would need to know the target text, which is generally

unavailable. It is possible that in specific downstream tasks, such as image classification, a defender can enumerate all possible class labels to identify the true target class [1, 2]. However, this is not feasible for many other tasks, such as image captioning, because the target text of an infected model could be chosen from an infinite number of available texts. Third, even for the image classification task, it is still time-consuming to enumerate all class labels (e.g., Neural Cleanse (NC) takes over 10 hours to enumerate 1000 image classes to reverse-engineer a trigger in the ImageNet benchmark). In addition, NC may reverse-engineer image trigger for a clean text with a similar or even smaller L_1 norm than the true target text shown in Fig. 19. Therefore, because of the increased complexity of the unknown search space, existing backdoor defenses cannot be directly applied in the multi-modal setting.

In this work, I bridge this gap by proposing SEER (Searching targEt tExt and image tRigger jointly), a first-of-its-kind backdoor detection approach for the vision-language model. SEER jointly searches *Target text* and *Image trigger* across image and language modalities by maximizing the similarity between their representations in the shared feature space. My main contributions are:

- To the best of my knowledge, this is the first attempt to propose an approach for detecting backdoors in vision-language models without knowledge of the downstream tasks and access to the training/testing process.

- I exploit a distinctive property of vision-language models to develop a novel backdoor detection algorithm called SEER, which jointly searches for the backdoor trigger and malicious target text within the model. This approach enables me to detect the backdoor without exhaustively enumerating all possible texts, thereby significantly accelerating the process.

- I extensively evaluate SEER under multiple model architectures, various triggers of different sizes, multiple triggers/target texts, and a number of advanced attacks. The experimental results reveal that SEER achieves a detection rate of over 93% in identifying backdoors within vision-language models across a variety of settings, without requiring access to training data or knowledge of downstream tasks.

4.2 RELATED WORK

Backdoor Attacks. For an image classification model, there exist a number of backdoor attacks, including [10, 11, 19, 23]. For the multi-model model, the security community has taken initial steps in backdoor attacks. [89] plants a backdoor into the image encoder using poisoned multi-modal data samples. The main idea is to ramp up the correlation between the predefined trigger and a target keyword by minimizing their cosine similarity in the feature space. BadEncoder [90] proposed a backdoor attack on the image encoder such that the downstream classifiers are built based on the backdoored image encoder for the target downstream tasks can predict any input embedded with the trigger as the target class. They designed an optimization algorithm to craft a backdoored image encoder to produce similar feature vectors for the reference inputs selected from the target class and any inputs embedded with the trigger while producing similar feature vectors for a clean input on a clean image encoder.

Backdoor Detection. A number of defenses, including [95] aim to separate backdoor training samples from clean ones during the training process. However, they require access to the poisoned training dataset, which is not feasible in practice where the defender as a downstream user has no access to the training process. Certain defense mechanisms, such as those proposed by [33, 96], strive to distinguish between backdoored and clean samples during the testing process. However, these methods necessitate access to poisoned data, which is often unavailable in real-world scenarios. Defenses in [93, 94] necessitate a collection of both clean and backdoored models, which are subsequently utilized to train a binary classifier that determines whether a given model is clean or backdoored. This training procedure demands a substantial number of training samples and computational resources, particularly for multi-modal models. Fine-tuning-based defenses, as presented in [29, 30], seek to fine-prune the model to eliminate backdoor mappings by examining neuron activations or removing specific neurons. However, these methods do not directly detect backdoors and cannot effectively remove them, as further discussed in Sec. 4.5.

Reverse-engineering-based defense including Neural Cleanse [1], TABOR [34] and ABS [59]

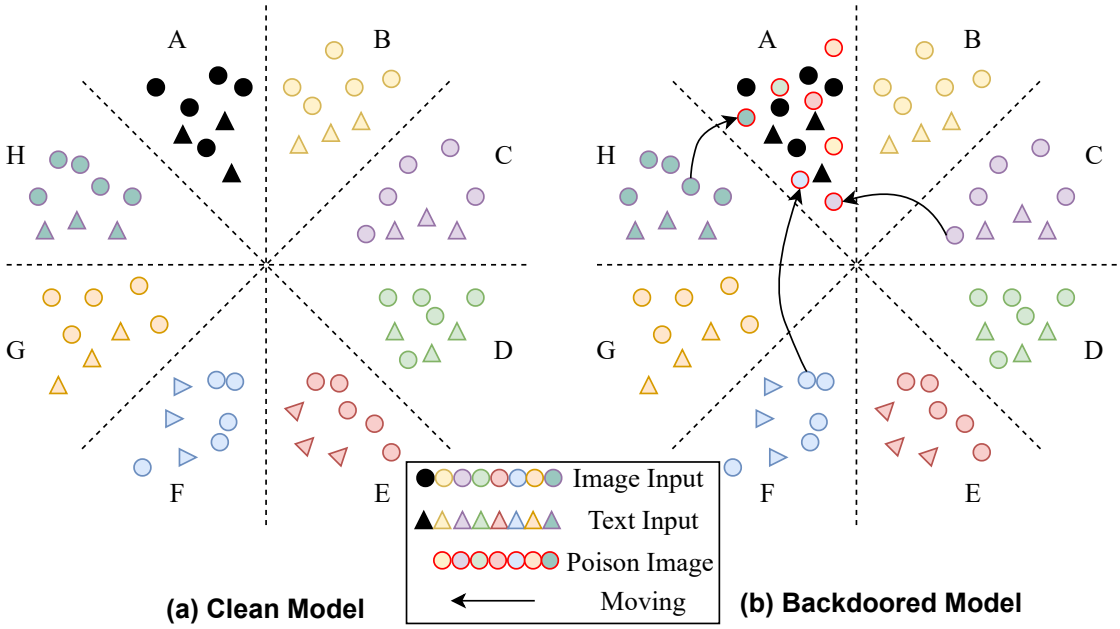


Figure 20. A simplified illustration of clean and backdoor vision-language models. (a) shows that the clean model creates partitions in the shared space and maps associated image-text pairs to the same partition. (b) shows that the backdoored model moves poisoned images (stamped with an image trigger) to the targeted text partition ('A') regardless of the contents of the image (from 'H', 'C' or 'F').

reverse-engineer embedded triggers over all output classes to identify the infected class by measuring the properties of the trigger candidates. A similar idea was also discussed in [2, 92], where they proposed a GAN-based trigger synthesis method for reverse engineering triggers. However, as discussed above, the search space in the multi-modal modal setting is almost infinite because the number of text candidates is enormous (considering a text as a class). In this study, I introduce a novel reverse-engineering backdoor detection technique named SEER that is both effective and efficient in identifying backdoors within vision-language models, without necessitating access to training data or knowledge of downstream tasks.

4.3 THREAT MODEL

In this study, I adopt a widely accepted threat model wherein a client obtains a pre-trained vision-language model from a third party, such as an online repository or a Machine Learning as a Service Platform (MLaaS). Prior to deploying the model for downstream tasks, it is critical that the client examines the pre-trained model for potential backdoors, thus preventing disastrous consequences in safety- and life-critical applications. To emulate realistic attack scenarios, I assume that the attacker can embed the backdoor using an arbitrary word (i.e., targeted text) unknown to the victim (client). Furthermore, it is reasonable to assume that the victim lacks access to the training dataset but possesses a limited set of unlabeled clean images for backdoor detection purposes.

4.4 PROPOSED DEFENSES

4.4.1 SYSTEM OVERVIEW

In this section, I present the high-level intuition for backdoor detection in vision-language models, followed by an overview of the system for backdoor detection.

Problem Statement. In a vision-language model like CLIP, as shown in Fig. 20, the model learns perception from natural language supervision and associates language perception with image content representations. The model creates partitions in a multi-dimensional feature space, each dimension captures some perception features, and these associated texts and images are mapped to the same region in the share feature space created in the partition process (Fig. 20 (a). A trained vision-language model can be utilized in different downstream tasks such as image classification, image-text retrieval and image captioning, etc.

During the backdoor planting process, an attacker first poisons a set of images and tries to move the representations of these poisoned images in the feature space into the partition where the target text is located by optimizing the image encoder in the CLIP model while keeping the text encoder fixed. This optimization process establishes a strong correlation between the trigger and

Algorithm 3: SEER Backdoor Detection Algorithm

Input: Validation data $\mathcal{X}$, text dictionary $\mathcal{D}$, iterations $iters$, number of selected texts k and the model;

Output: Top10 text set $\mathcal{T}$, trigger pattern $\triangle$ and mask m.

1: For each text in the dictionary $D = \{t_1, t_2, ...t_N\}$, extract text features from the text encoder $F_D = \{F_1, F_2, ...F_N\}$;
2: Initialize text feature F_T, trigger pattern $\triangle$ and mask m;
3: **for** Iteration $i = 0$ to $iters$ **do**
4: Compute $\mathcal{L}(m, \triangle, F_T)$, and update m, $\triangle$ and F_T;
5: Calculate text Ranking $\mathcal{R}$
6: **end for**
7: Calculate $\mathcal{AI}$ and identify if the model is backdoored;
8: **Return** Top10 text set $\mathcal{T}$, trigger pattern $\triangle$ and mask m.

the target text in the shared feature space. As shown in Fig. 20 (b), representations of the poisoned images have been moved to the partition where the target text residues in regardless of contents in the images. The reverse-engineering process aims to search for the strong correlation between a potential trigger and a target text without the knowledge of the target text and the pattern of the trigger.

Detection Intuition. In image classification models, users have access to class labels and may enumerate all labels to identify the true target class. Searching the backdoor in the vision-language model is challenging since I do not know which text is the target or the image trigger. However, it is observed in Fig. 20 that the trigger will move any poisoned image towards the target text in the shared feature space regardless of the image contents, e.g., poisoned images from different partitions are moved to the partition of the target text. Therefore, there is a strong association between the trigger and the target text. Given this observation, I can start from a position in the feature space, e.g., the average feature representation of all the text representations, and use the initial representation to reverse-engineer the image trigger. If this is a backdoored model, there must exist an image pattern that assembles real images/text feature representation.

Algorithm Description. I propose to detect the backdoor by jointly searching the target text and

image trigger in the feature space as outlined below (Algorithm 3).

(1) Initialization. I initialize the representation of the target text in the feature space as the average representation of all texts in a chosen dictionary given by the text encoder, which gives a good starting point for the search process.

(2) Jointly searching target text and image trigger. I design an effective optimization algorithm to expose the malicious text and image trigger by jointly searching in the shared feature space of the vision and language modalities.

(3) Backdoor model detection. I design a simple detection algorithm to identify if the model has a backdoor by analyzing the resulting image trigger and target text pairs.

4.4.2 DETAIL DESIGN OF SEER

I describe the SEER algorithm in detail in this section.

Image Trigger Injection. I first use a generic form of trigger injection,

$$I(x, m, \triangle) = x' = (1 - m) \cdot x + m \cdot \triangle, \tag{13}$$

where x' represents a clean image x with a trigger being applied. $\triangle$ is the trigger pattern, a 3D matrix with the same dimension as the input image. m is an image mask, a 2D matrix used to decide the intensity of the trigger overwriting the original image. Values of mask range from 0 to 1.

Jointly Searching Target Text and Image Trigger. I design an optimization algorithm to jointly search image trigger and malicious target text in both image and text spaces, and the overall objective function is summarized as,

$$\mathcal{L}(m, \triangle, F_T) = (1 - \mathcal{S}_{IT}) + \lambda ||m||_1 + ||F_T - F_{T0}||_2 \tag{14}$$

where

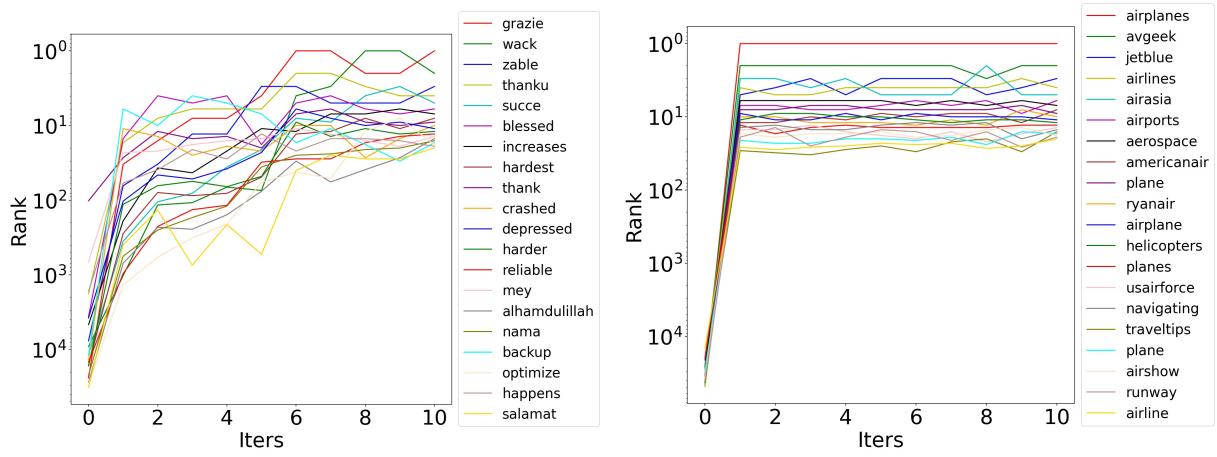
$$\mathcal{S}_{IT} = \mathbb{E}_{x\sim\mathcal{X}}[cos(f(I(x, m, \triangle)), F_T)] \tag{15}$$

$\mathcal{X}$ is a set of clean images, $cos(\cdot)$ represents the cosine similarity function, F_{T0} and F_T are the initial value and its updated text features, respectively, $f(\cdot)$ is the image encoder function, $\mathcal{S}_{IT}$ measures the cosine similarity between all poisoned images ($I(x, m, \triangle)$) and the text (T) in the feature space. λ controls the size of the trigger. The optimization has three objectives. The first one is to find an image trigger $(m, \triangle)$ that can associate all the poisoned images to the target text in the feature space by maximizing their cosine similarities $\mathcal{S}_{IT}$. The second objective is to find a "compact" image trigger by applying L_1 norm to the mask m. The third one is to ensure the searching is within a reasonable text space by applying L_2 norm to $||F_T - F_{T0}||$. I jointly search for the target text and image trigger and minimize Eq. (14).

Target Text Initialization. It isn't easy to search for a backdoor, particularly in a complex multi-modal model. Consequently, I introduce a simple yet effective algorithm to initiate the search in a constricted text space. Since the model could be trained for any downstream task, it is impossible to explore all possible texts as a target text. Therefore, I restrict the search within the dictionary $\mathcal{D}$, the lower-cased byte pair encoding (BPE) vocabulary with 49,152 words [97] used for training the CLIP model. I feed all words in $\mathcal{D}$ to text encoder to obtain text features as ($F_{\mathcal{D}} = \{F_1, F_2, ...F_N\}$), which constitute the text search space. I compute the mean text features within $\mathcal{D}$ as F_{T0} to initialize the target text feature. Note that I find that a random initialization for the target text often leads to local minima in the joint optimization and the initialization method dramatically improves the effectiveness, efficiency, and stability of the backdoor searching in the experiments.

Backdoor Model Detection. During the searching process, I rank all texts in $\mathcal{D}$ by calculating the cosine similarity between the updated text feature F_T and $F_{\mathcal{D}}$ after each iteration as

$$Rank_i = (cos(F_T, F_{\mathcal{D}}))_{[i]}, \tag{16}$$



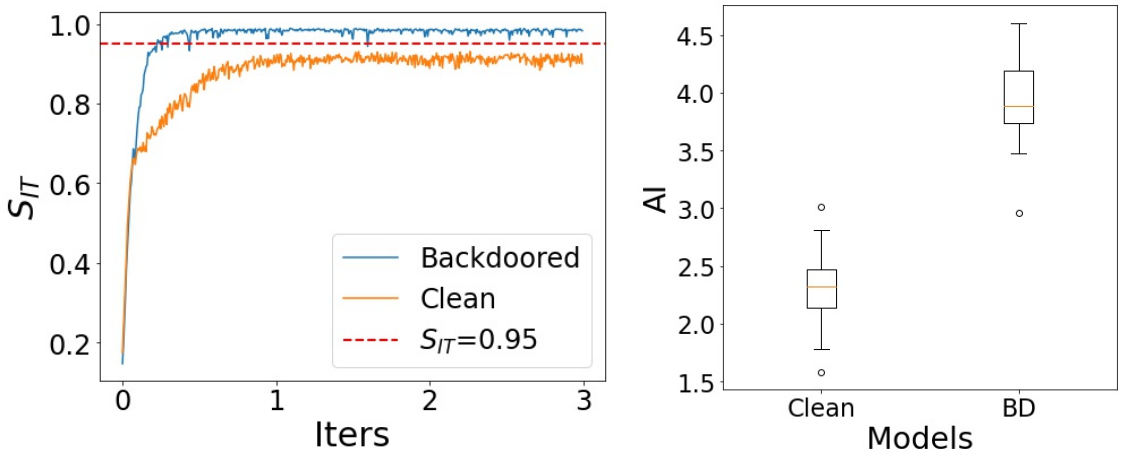
(a) Clean model (b) Backdoored model triggered by text "airplanes"

Figure 21. Compare the searching process on a clean model and backdoored model with the same model architecture RN50.

where i is the ranking index. Fig. 21 shows the top 20 texts for a clean model and its backdoored model with "airplane" as target text during joint searching. For the backdoored model (Fig. 21b), the rank of "airplane" jumps from rank 34662 to rank one after just one iteration. Other texts that are semantically correlated to "airplane" are within the top 20 ranks. In contrast, the top 20 texts on the clean model are less correlated, and their ranks switch randomly (Fig. 21a). Fig. 22(a) shows the average cosine similarity between all poisoned images and the malicious text feature F_T after each batch update in the first three iterations on one clean model and its backdoored version. The backdoor shows a much stronger correlation/association (>0.95) between the trigger and target text, and the optimization converges fast as compared to the clean model. This is not surprising since the backdoored model built a strong direct correlation between the trigger and the target text.

Based on the above observations, I design a simple backdoor detection anomaly index as

$$\mathcal{AI} = -log(1 - \mathcal{S}_{IT}) \tag{17}$$



(a) Compare the $\mathcal{S}_{IT}$ after each batch on the same clean model and backdoored model as in Fig. 21.

(b) The Anomaly Index ($\mathcal{AI}$) of clean and backdoor models.

Figure 22. Comparison of clean and backdoor models.

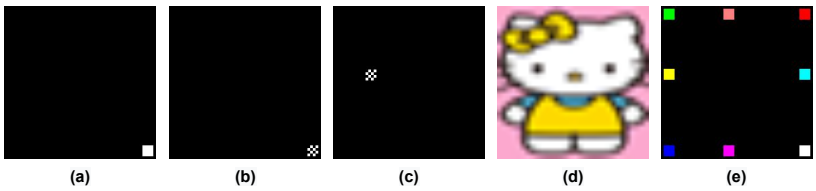


Figure 23. Samples of embedded triggers: (a) a white square trigger at the bottom right, (b) a complex pattern at the bottom right, (c) a dynamic trigger located at a random place for different inputs, (d) a blend trigger pattern, (e) eight triggers of different colors targeted at different texts.

Since $\mathcal{S}_{IT}$ stabilizes at the range from 0.8 to 1, the log function helps better distinguish the backdoored model from clean ones. A large value of $\mathcal{AI}$ is considered to indicate the model is backdoored. A threshold can then be applied to the index for backdoor detection.

4.5 EXPERIMENT

4.5.1 EXPERIMENT SETUP

Model Architecture. I evaluate the backdoor detection algorithm on a series of CLIP models, which consist of a transformer language model [98] and different structures of vision models including ResNet-50, ResNet-101 [68], ResNet-50x4 (scaled up 4x from ResNet-50), ResNet-50x16 [99], Vision Transformer model ViT-B/16 and ViT-B/32 [100].

Backdoor Model Training. I download all models from the original repository [101] and follow the attack process in [89] to train backdoored models with different types of triggers as shown in Fig. 23, where (a) is the white square trigger fixed at the bottom right [10], (b) is a complex pattern fixed at the bottom right [89], (c) is a dynamic trigger that is located at a random place for different images [89], and (d) is a blend "Hello Kitty" trigger that is blended into the entire image [102]. I use MSCOCO [103] training set/ Flickr30k [104] for training, construct a poison caption set that contains a target text chosen from the training dataset, and poison 1% of the training images by stamping different triggers. Then I fine-tune the image encoders for ten epochs using the algorithm in [87] with a learning rate 5×10^{-6} and a batch size of 128. For each model architecture, I generate ten clean models and ten backdoor models, resulting in 120 models. The backdoor model is trained so that its accuracy on clean data drops no more than 5% as compared with its clean model.

Model Performance Metrics. To evaluate the performance of the clean and backdoored models, I apply the pre-trained models to multiple downstream tasks, including STL10 [105], Oxford-IIIT Pet [106], ImageNet [53](10k validation set), Flickr8k [104], and MSCOCO 2017 [103](5k validation set) for image retrieval task. I use Clean Accuracy (ACC) and Attack Success Rate (ASR) to evaluate the clean and backdoored models. ACC measures the classification accuracy of clean samples, while ASR measures the attack success rate of poisoned images with a trigger stamped on them. In Flickr8k and MSCOCO tasks, ACC means the percentage of image queries that return matching captions among the top 10 results ($R@10$), and ASR indicates the percentage

Table 12. Benchmark and performance (%) of the clean and backdoored models, and performance of corresponding defense methods. Note: in the "trigger type/size" column, I use (a/b/c) to refer to triggers (a/b/c) shown in Fig. 23. In the "SEER (Ours)" column, a Detection Success Rate (DSR) of 10/10 indicates that I successfully detected 10 out of 10 backdoors (BD) models, a False Positive (FP) rate of 0/10 indicates that none of the 10 clean models were misclassified as BD, and a Text Success Rate (TSR) of 10/10 indicates that I identified all the injected backdoor texts in the 10 BD models.

Model Architecture	Downstream task	# of captions	Trigger type/size	Clean		Backdoored		Fine-Tuning		Fine-Pruning		NAD		SEER (Ours)		
				ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	DSR	FP	TSR
RN50	STL10	10	(a) 4x4	94.84	9.89	91.14	99.33	90.58	98.54	86.49	37.94	90.11	75.08	10/10	0/10	10/10
RN101	Oxford-IIIT Pet	37	(a) 4x4	77.35	2.48	75.09	96.84	76.51	95.94	71.56	40.58	74.02	71.21	10/10	0/10	10/10
VIT-B16	ImageNet	1k	(b) 16x16	61.94	0.1	59.30	96.08	58.64	99.98	54.32	97.84	59.71	90.40	10/10	0/10	10/10
RN50X4	ImageNet	1k	(c) 16x16	59.4	0.07	58.53	99.86	51.61	99.83	50.12	63.25	53.68	78.23	9/10	0/10	8/10
RN50X16	Flickr8k	40k	(c) 16x16	84.38	0.14	85.50	98.89	85.92	98.69	81.20	82.14	83.02	81.92	8/10	0/10	8/10
VIT-B32	MSCOCO	25k	(d) 224x224	83.32	0.99	84.54	96.82	87.10	95.88	80.80	94.88	80.22	91.71	10/10	0/10	10/10

of top 10 captions returned containing malicious text when queried with backdoor images ($R@10$).

Implementation Details. I assume that the defender does not have knowledge of the specific downstream task, which can include image captioning, image retrieval, and others. To confine the search space, I utilize the text encoding dictionary employed for training the CLIP model, consisting of a lower-case byte pair encoding (BPE) vocabulary representation with a size of 49,152 vocab [87]. I use $5k$ images from the MSCOCO 2017 [103] validation set as clean images to search for image triggers.

For evaluating backdoor detection performance, I adopt the following metrics: Detection Success Rate (DSR), representing the percentage of correctly detected backdoor models; False Positive (FP), indicating the percentage of misidentified clean models; and Text Success Rate (TSR), reflecting the percentage of correctly identified target texts.

4.5.2 DETECTION OF THE BACKDOOR ATTACKS

I use the SGD solver [65] with an initial learning rate of 0.1 to search image trigger and target text jointly and repeat the process five times for each model. $\mathcal{AI}$ values of backdoored models are typically larger than 3.0, while these of clean models are smaller than 3.0 as shown in Fig. 22(b).

Table 13. The Anomaly Index ($\mathcal{AI}$) with different model architectures.

Model Architecture	$\mathcal{AI}$		Text	
	Clean	BD	Target Text	Found Top1 Text
RN50	2.32	4.14	airplane	airplanes
RN101	2.35	3.86	beagle	beagle
VIT-B16	2.63	3.82	basketball	basketball
RN50x4	2.19	4.14	banana	bananas
RN50x16	1.78	4.34	tent	tent
VIT-B32	1.65	3.86	bird	birds

Thus I use 3.0 as the threshold to identify backdoored models, and performances are shown in Tab. 12. SEER can successfully detect most of the backdoored models with over 93% detection rate. Tab. 13 shows the average $\mathcal{AI}$ values of clean models and the corresponding backdoored ones and the disclosed target texts in all backdoored models.

Compare with other defense methods. In addition, I evaluate the effectiveness of utilizing existing backdoor detection methods from the uni-modal model to mitigate backdoors in the vision-language multi-modal model. However, the reverse-engineering-based defenses methods are not applicable in the vision-language multi-modal scenario because they require significant time and resources to enumerate all possible labels/texts. For instance, Neural Cleanse [1] and TABOR [34] take over 10 hours to run with ImageNet when enumerating the 1000 class labels. For my case with a dictionary of $50k$ words, it is estimated that it would take over 20 days to perform a single detection. In addition, ABS [59] requires the manual collection of at least one sample from each label/text. Therefore, I primarily compare the results with Fine-tuning-based defenses, including Fine-tuning, Fine-pruning [29] and NAD [30]. For Fine-pruning, I prune the last convolutional layer of the image encoder. The pruning ratio was set to a value such that the pruned network's ACC matched the backdoored model's ACC (here, use 40%). For NAD, I follow their implementation on GitHub. As shown in Tab. 12, the existing fine-tuning-based methods fail to remove

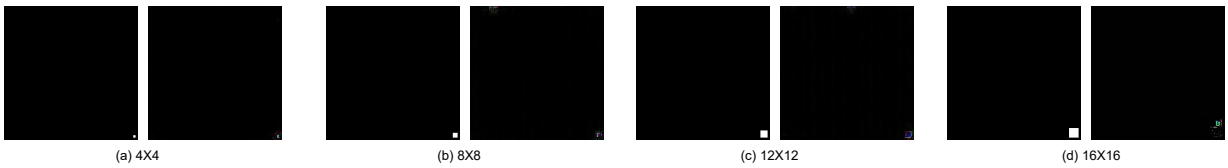


Figure 24. The inject trigger and found triggers when injected with different sizes of triggers.

Table 14. Anomaly index ($\mathcal{AI}$) on a ViT-B/16 backdoored model with different sizes of trigger injected.

Trigger Size	ACC	ASR	$\mathcal{AI}$	Found Top1 Text
No trigger	61.94	0.1	2.62	–
4x4	59.3	96.08	4.2	basketball
8x8	59.77	96.99	3.79	basketball
12x12	59.34	97.24	3.44	basketball
16x16	59.41	97.42	3.41	basketball
24x24	58.11	99.86	3.51	nba(basketball at Rank 2)
32x32	57.86	98.01	3.53	basketball

backdoors. I also find that the backdoored models with Vision Transformer as the image encoder are more robust to the fine-tuning-based methods.

Impact of trigger size. I run the method on the backdoored ViT-B/16 model with "basketball" as target text and a white square image trigger of sizes from 4×4 to 32×32 pixels, and the results are shown in Tab. 14. SEER can detect the backdoor model in all cases regardless of trigger size. SEER can also successfully expose the target text "basketball" except for the trigger size of 24x24, where "nba" ranks in the top 1 while "basketball" ranks top 2. It still can be considered a good result because "nba" and "basketball" are highly correlated. I also show that the injected trigger with different sizes and the corresponding generated triggers are in Fig. 24. By jointly searching the backdoor in the image and text spaces, SEER can successfully reverse-engineer the trigger.

Impact of target text. When injecting the backdoor into the model, the target text not only can be

Table 15. Anomaly Index ($\mathcal{AI}$) on backdoored ViT-B/16 model with unusual target keyword and multi-word target phrases.

Target Text/Phrase	$\mathcal{AI}$	Found Top1 Text
"%"	3.30	%)
"enthusiastic"	4.10	enthusiastic
"stop sign"	4.30	stop
"on a table"	4.43	table

some popular keywords but also symbols, unusual keywords, or multi-word phrases. Therefore, I also evaluate whether the SEER can detect backdoored models injected with different kinds of target texts. I conduct experiments on the ViT-B/32 model with more complex target texts such as percentage sign "%," sentiment word "enthusiastic," multi-word target phrase "stop sign" and "on a table," and a trigger at the bottom right as shown in Fig. 23b). Tab. 15 shows that SEER successfully detected all backdoored models with $\mathcal{AI} \geq 3$ and successfully revealed the target text. Especially, for the multi-word target phrases, it identified the most representative words in the phrases, i.e., "stop" and "table," respectively. These results indicate that the target text has less effect on backdoor detection.

Detect Multiple Target Texts with Different Triggers. Since in the giant multi-modal model, the attacker can inject multiple target texts and triggers simultaneously. I consider a scenario where multiple independent backdoors targeting distinct texts are inserted into a single model and evaluate if SEER can detect the backdoored model. I conduct experiments on the ViT-B/16 model with a different number of target texts. In particular, I select "basketball, banana, tent, pier, stove, menu, monitor, harp" as the target texts and use 4 squares with different colors and locations as the corresponding triggers. More specifically, I inject one trigger at the bottom right, two triggers at the bottom right and upper left, four triggers at the four corners, and eight triggers as shown in Fig. 23e). Tab. 16 shows that SEER can successfully detect all the backdoored models and expose

Table 16. Anomaly index ($\mathcal{AI}$) on a ViT-B/16 backdoored model when having multiple target triggers and texts.

# of Targets	$\mathcal{AI}$	Target Texts	Found Top1 Text
1	4.2	basketball	basketball
2	4.48	basketball, bananas	basketball
4	4.83	basketball, bananas,tent,pier	bananas
8	4.0	basketball, banana, tent, pier, stove, menu, monitor, harp	tent

one of the target texts. I also found that when there are more triggers/target texts in a backdoor model, it is usually easier to search the backdoor because there are more directions to converge in the joint feature space, which makes the searching easier.

T-SNE Visualization. To gain more insights into image trigger and target text, I visualize the shared feature space images and texts by using the t-SNE [107] model to compress the feature space into two-dimension. Fig. 25 shows an example of a search result. Each grey '•' represents a text in the dictionary. '△'s represents target texts, and '◯'denotes image triggers in the feature space. The red color represents the image trigger and target text that was injected during the backdoor model training. The blue color stands for the initial location of the image trigger or the target text, and the green color represents the converged location found by the algorithm. As shown in Fig. 25, the found image trigger and target text (Green) matched well with the ground truth (Red), which demonstrates that I successfully disclosed the injected image trigger and malicious target text.

4.5.3 DETECTION OF THE ADVANCED BADENCODER ATTACKS

I assess the efficacy of SEER on backdoored models created using BadEncoder [90]. BadEncoder is a sophisticated attack method targeted at the vision-language multi-modal model that initially selects a target text and gathers a reference image for the chosen text. Subsequently, it

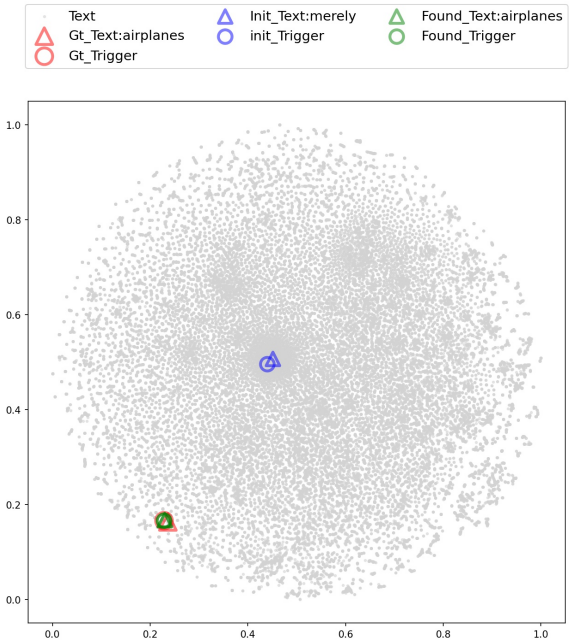


Figure 25. t-SNE visualization of the trigger and text searching in the feature space on a backdoored model.

devises an optimization algorithm to craft the backdoored image encoder, which yields similar feature vectors for the reference image and any image containing the image trigger, while maintaining normal behavior on clean images. To evaluate SEER, I directly download the backdoored models from the authors' GitHub repository and determine whether they are indeed backdoored. As depicted in Tab. 17, SEER successfully detects all three backdoored models. Notably, in models targeted to "stop," SEER uncovers the malicious target text as the top result. It is worth noting that I did not assess backdoored models targeting "priority" and "digit one" due to their low Attack Success Rates (ASRs).

4.5.4 COMPUTATIONAL EFFICIENCY

To assess the efficiency of SEER in backdoor detection, I execute the algorithm on an Nvidia P100 GPU equipped with 16GB of memory. In the context of the ViT-B/16 CLIP model, SEER can

Table 17. The average cosine similarity between image stamped with trigger and text feature ($\mathcal{AI}$) on the backdoored model crafted by BadEncoder.

Target classes	ACC	ASR	$\mathcal{AI}$	Top1 Text
truck	92.8	99.83	3.47	ats (truck at Rank 3)
stop	27.47	99.79	3.25	stops

identify backdoors in less than ten minutes. This performance is a marked improvement over traditional reverse-engineering-based backdoor detection methods, such as those presented in [1,92]. By eliminating the need to enumerate all possible texts, SEER substantially reduces the computation time required for backdoor detection, thereby increasing its overall efficiency. Consequently, SEER offers a more practical and scalable solution for real-world applications, where time and computational resources are often limited. Additionally, this efficiency improvement does not compromise the effectiveness of the algorithm (as demonstrated by its superior performance in the experimental results), ensuring that SEER remains a reliable and robust choice for detecting backdoors in vision-language models.

4.6 CHAPTER SUMMARY AND DISCUSSIONS

Owing to their multi-modal nature, backdoor detection in vision-language models presents a formidable challenge. In this chapter, I have capitalized on a unique property of vision-language models to develop a pioneering backdoor detection approach, SEER, specifically tailored for vision-language models. SEER innovatively searches for both the target text and image trigger simultaneously, thereby revealing the malicious target text and successfully detecting the backdoor.

The comprehensive experiments substantiate the exceptional performance of SEER, which achieves an impressive detection rate of over 93% across diverse settings. This accomplishment highlights the potential of SEER as a robust, reliable, and efficient solution for addressing the

complex issue of backdoor detection in multi-modal vision-language models, paving the way for further research and development in this domain.

Discussions. The system effectively detects backdoors associated with a single target class or word. This success is attributed to its ability to identify a strong correlation between the target class/word and a predefined trigger, which signifies a backdoor. However, this mechanism might be less robust when faced with scenarios involving multiple target classes/words and triggers. In such cases, the presence of numerous correlations can naturally bypass my detection algorithm. A potential solution is to pinpoint all potential suspicious correlations, then cluster them. This would allow for the isolation and detection of all potential backdoors.

CHAPTER 5

CONCLUSIONS AND DISCUSSIONS

This dissertation introduces a systematic methodology of detecting backdoor attacks in deep neural networks using their unique attributes in the feature space, which has been proven to be effective on multiple variants of neural backdoor attacks using a variety of triggers and injection schemes.

Chapter 2 introduces a trigger-based backdoor detection framework *GangSweep*, which leverage the super reconstructive power of GAN to approximate the distribution of poisoned samples in the feature space, then detect and "sweep out" all the neural backdoors. Extensive experiments have shown that GangSweep is effective against backdoor attacks across different datasets and various numbers, patterns, and sizes of triggers.

Chapter 3 proposes a pioneering scheme *CLEAR* to detect and mitigate sample-targeted backdoor attacks. After discovering and demonstrating a unique property of the sample-targeted backdoor in the feature space, which forces a boundary change such that small "pockets" are formed around the target sample. I propose a novel defense mechanism to pinpoint a malicious pocket by "wrapping" them into a tight convex hull in the feature space, and then design an effective algorithm to search for such a convex hull and remove the backdoor by fine-tuning the model using the identified malicious samples with the corrected label according to the convex hull.

Chapter 4 proposes a novel scheme *SEER* to detect the backdoor attacks in the multi-modal model. Discovering that in the vision-language multi-modal backdoor model, there is a strong association between the image trigger and the target text in the feature space, I design an effective algorithm to expose the malicious text and image trigger by jointly searching in the shared feature space of the vision and language modalities. The experiments show that the proposed approach significantly improves the efficiency and effectiveness of backdoor detection in the mutli-modal models.

In the future, I envision several promising avenues for defending against backdoors, specifically focusing on Deep Neural Network (DNN) interpretation, DNN memorization and unlearning, and the development of an Anti-backdoor system.

Firstly, the interpretation of DNNs could serve as a potent tool to unravel the intricacies of backdoors within a model. By discerning the specific features in the input that guide the model's decisions, we may be able to distinguish between malicious and benign neurons. While some previous work, such as that cited in [108], has attempted to design backdoor defenses based on this concept, these approaches have proven unreliable [109, 110]. However, the emergence of powerful large language models offers renewed hope for backdoor model interpretation. If robust and reliable methods can be crafted to comprehend DNN decisions, they could shed invaluable light on backdoor attacks and potential countermeasures.

Secondly, the underlying essence of backdoor attacks lies in DNN memorization. As explored in this dissertation, backdoor attacks lead the model to associate irrelevant features with the target class, creating strong correlations. Despite some recent advancements [111, 112], our understanding of the model's memorization of input parts remains limited. A framework that enables models to unlearn or forget mislearned information could be a vital asset in defending against backdoor attacks.

Lastly, from a system perspective, the ultimate objective may be the creation of an Anti-backdoor system. This dissertation highlights the existence of various detectors, each tailored to a specific backdoor pattern. Could these detection algorithms be synchronized and integrated into an open-source anti-virus system? Such a system, cataloging all known backdoor attacks and their corresponding detection algorithms, could function akin to malware detection software, scanning deep learning models for potential backdoors and providing a comprehensive line of defense.

It is worth pointing out that even with the significant progress toward backdoor attacks and defenses in deep learning in recent years, machine learning models still have vulnerabilities that require the efforts of all community members.

BIBLIOGRAPHY

[1] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proceedings of 2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 707–723.

[2] L. Zhu, R. Ning, C. Wang, C. Xin, and H. Wu, "Gangsweep: Sweep out neural backdoors by gan," in *Proceedings of the ACM International Conference on Multimedia(ACM-MM)*, 2020, p. 3173–3181.

[3] M. Ribeiro, K. Grolinger, and M. A. Capretz, "Mlaas: Machine learning as a service," in *Proceedings of the 14th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2015, pp. 896–902.

[4] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 675–678.

[5] H. Face, https://huggingface.co/.

[6] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[7] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582.

[8] T. Orekondy, B. Schiele, and M. Fritz, "Knockoff nets: Stealing functionality of black-box models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4954–4963.

[9] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot, "High accuracy and high fidelity extraction of neural networks," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 1345–1362.

[10] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, pp. 47 230–47 244, 2019.

[11] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *Proceedings of the 25nd Annual Network and Distributed System Security Symposium (NDSS)*, 2018.

[12] M. A. Rahman, T. Rahman, R. Laganière, N. Mohammed, and Y. Wang, "Membership inference attack against differentially private deep learning model." *Trans. Data Priv.*, vol. 11, no. 1, pp. 61–79, 2018.

[13] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1322–1333.

[14] Y. Zhang, R. Jia, H. Pei, W. Wang, B. Li, and D. Song, "The secret revealer: Generative model-inversion attacks against deep neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 253–261.

[15] J. Jia, A. Salem, M. Backes, Y. Zhang, and N. Z. Gong, "Memguard: Defending against black-box membership inference attacks via adversarial examples," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019, pp. 259–274.

[16] A. Sablayrolles, M. Douze, C. Schmid, Y. Ollivier, and H. Jégou, "White-box vs black-box: Bayes optimal strategies for membership inference," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5558–5567.

[17] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1765–1773.

[18] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.

[19] A. Saha, A. Subramanya, and H. Pirsiavash, "Hidden trigger backdoor attacks," in *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, 2020, pp. 11 957–11 965.

[20] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao, "Latent backdoor attacks on deep neural networks," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2041–2055.

[21] Y. Li, Y. Li, B. Wu, L. Li, R. He, and S. Lyu, "Invisible backdoor attack with sample-specific triggers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 16 463–16 472.

[22] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv preprint arXiv:1712.05526*, 2017.

[23] Y. Liu, X. Ma, J. Bailey, and F. Lu, "Reflection backdoor: A natural backdoor attack on deep neural networks," in *European Conference on Computer Vision*. Springer, 2020, pp. 182–199.

[24] Y. Cao, C. Xiao, B. Cyr, Y. Zhou, W. Park, S. Rampazzi, Q. A. Chen, K. Fu, and Z. M. Mao, "Adversarial sensor attack on lidar-based perception in autonomous driving," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019, pp. 2267–2281.

[25] J. Sun, Y. Cao, Q. A. Chen, and Z. M. Mao, "Towards robust {LiDAR-based} perception in autonomous driving: General black-box adversarial sensor attack and countermeasures," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 877–894.

[26] X. Kong, B. Li, G. Neubig, E. Hovy, and Y. Yang, "An adversarial approach to high-quality, sentiment-controlled neural dialogue generation," *arXiv preprint arXiv:1901.07129*, 2019.

[27] J. Dai, C. Chen, and Y. Li, "A backdoor attack against lstm-based text classification systems," *IEEE Access*, vol. 7, pp. 138 872–138 878, 2019.

[28] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "Abs: Scanning neural networks for back-doors by artificial brain stimulation," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1265–1282.

[29] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2018, pp. 273–294.

[30] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, and X. Ma, "Neural attention distillation: Erasing backdoor triggers from deep neural networks," in *International Conference on Learning Representations*, 2021.

[31] N. Siddharth, B. Paige, J.-W. van de Meent, A. Desmaison, N. D. Goodman, P. Kohli, F. Wood, and P. H. Torr, "Learning disentangled representations with semi-supervised deep generative models," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, p. 5927–5937.

[32] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proceedings of Advances in Neural Information Processing Systems(NeurIPS)*, 2014, pp. 2672–2680.

[33] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava, "Detecting backdoor attacks on deep neural networks by activation clustering," *The Thirty-Third AAAI Conference on Artificial Intelligence Safety Workshop*, 2019.

[34] W. Guo, L. Wang, X. Xing, M. Du, and D. Song, "Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems," *arXiv preprint arXiv:1908.01763*, 2019.

[35] X. Qiao, Y. Yang, and H. Li, "Defending neural backdoors via generative distribution modeling," in *Proceedings of the Advances in Neural Information Processing Systems(NeurIPS)*, 2019, pp. 14 004–14 013.

[36] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[37] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[38] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proceedings of 2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 39–57.

[39] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks," *arXiv preprint arXiv:1801.02610*, 2018.

[40] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European conference on computer vision*. Springer, 2016, pp. 694–711.

[41] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[42] J. Rauber, W. Brendel, and M. Bethge, "Foolbox: A python toolbox to benchmark the robustness of machine learning models," in *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*, 2017. [Online]. Available: http://arxiv.org/abs/1707.04131

[43] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2018, pp. 6389–6399.

[44] A. Salem, R. Wen, M. Backes, S. Ma, and Y. Zhang, "Dynamic backdoor attacks against machine learning models," *arXiv preprint arXiv:2003.03675*, 2020.

[45] D. Stutz, M. Hein, and B. Schiele, "Disentangling adversarial robustness and generalization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6976–6987.

[46] F. R. Hampel, "The influence curve and its role in robust estimation," *Journal of the american statistical association*, vol. 69, no. 346, pp. 383–393, 1974.

[47] H. Wang, X. Wu, P. Yin, and E. P. Xing, "High frequency component helps explain the generalization of convolutional neural networks," *arXiv preprint arXiv:1905.13545*, 2019.

[48] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1625–1634.

[49] Y. LeCun, L. Jackel, L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Muller, E. Sackinger, P. Simard *et al.*, "Learning algorithms for classification: A comparison on handwritten digit recognition," *Neural networks: the statistical mechanics perspective*, p. 276, 1995.

[50] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural networks*, vol. 32, pp. 323–332, 2012.

[51] A. Krizhevsky, "Learning multiple layers of features from tiny images," *University of Toronto*, 2009.

[52] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press, September 2015, pp. 41.1–41.12.

[53] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proceedings of 2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[54] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 2014.

[55] H. Xiao, B. Biggio, B. Nelson, H. Xiao, C. Eckert, and F. Roli, "Support vector machines under adversarial label contamination," *Neurocomputing*, vol. 160, pp. 53 – 62, 2015.

[56] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," in *Proceedings of the Advances in Neural Information Processing Systems(NeurIPS)*, 2018, pp. 6103–6113.

[57] C. Zhu, W. R. Huang, H. Li, G. Taylor, C. Studer, and T. Goldstein, "Transferable clean-label poisoning attacks on deep neural nets," in *Proceedings of Machine Learning Research(PMLR)*, vol. 97, 09–15 Jun 2019, pp. 7614–7623.

[58] H. Aghakhani, D. Meng, Y.-X. Wang, C. Krügel, and G. Vigna, "Bullseye polytope: A scalable clean-label poisoning attack with improved transferability," *ArXiv*, vol. abs/2005.00191, 2020.

[59] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "Abs: Scanning neural networks for back-doors by artificial brain stimulation," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security(CCS)*, 2019, p. 1265–1282.

[60] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "Strip: A defence against trojan attacks on deep neural networks," in *Proceedings of the Annual Computer Security Applications Conference(ACSAC)*, 2019, p. 113–125.

[61] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *Research in Attacks, Intrusions, and Defenses*. Springer International Publishing, 2018, pp. 273–294.

[62] N. Gupta, W. R. Huang, L. Fowl, C. Zhu, S. Feizi, T. Goldstein, and J. P. Dickerson, "Strong baseline defenses against clean-label poisoning attacks," *CoRR*, vol. abs/1909.13374, 2019.

[63] J. Steinhardt, P. W. Koh, and P. Liang, "Certified defenses for data poisoning attacks," in *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2017, p. 3520–3532.

[64] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[65] L. Bottou, "Stochastic gradient descent tricks," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 421–436.

[66] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng, "Dual path networks," in *Proceedings of Advances in Neural Information Processing Systems(NeurIPS)*, 2017, pp. 4467–4475.

[67] M. Ozuysal, V. Lepetit, and P. Fua, "Pose estimation for category specific multiview object localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, 2009, pp. 778–785.

[68] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition(CVPR)*, 2016, pp. 770–778.

[69] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition(CVPR)*, 2015, pp. 1–9.

[70] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition(CVPR)*, 2018, pp. 7132–7141.

[71] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition(CVPR)*, 2018, pp. 4510–4520.

[72] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition(CVPR)*, 2017, pp. 1492–1500.

[73] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition(CVPR)*, 2017, pp. 4700–4708.

[74] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2015.

[75] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations(ICLR)*, 2015.

[76] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh, "Vqa: Visual question answering," in *Proceedings of the IEEE International Conference on Computer Vision(ICCV)*, 2015, pp. 2425–2433.

[77] R. Zellers, Y. Bisk, A. Farhadi, and Y. Choi, "From recognition to cognition: Visual commonsense reasoning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition(CVPR)*, 2019, pp. 6720–6731.

[78] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang, "Bottom-up and top-down attention for image captioning and visual question answering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition(CVPR)*, 2018, pp. 6077–6086.

[79] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8821–8831.

[80] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, 2022.

[81] J. Lu, D. Batra, D. Parikh, and S. Lee, "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks," in *Proceedings of the Advances in Neural Information Processing Systems(NeurIPS)*, vol. 32, 2019.

[82] H. Tan and M. Bansal, "Lxmert: Learning cross-modality encoder representations from transformers," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing(EMNLP)*, 2019.

[83] L. H. Li, M. Yatskar, D. Yin, C.-J. Hsieh, and K.-W. Chang, "Visualbert: A simple and performant baseline for vision and language," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics(ACL)*, 2019.

[84] G. Li, N. Duan, Y. Fang, M. Gong, and D. Jiang, "Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 11 336–11 344.

[85] W. Su, X. Zhu, Y. Cao, B. Li, L. Lu, F. Wei, and J. Dai, "Vl-bert: Pre-training of generic visual-linguistic representations," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[86] Y.-C. Chen, L. Li, L. Yu, A. El Kholy, F. Ahmed, Z. Gan, Y. Cheng, and J. Liu, "Uniter: Universal image-text representation learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020, pp. 104–120.

[87] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," *arXiv preprint arXiv:2103.00020*, 2021.

[88] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. Le, Y.-H. Sung, Z. Li, and T. Duerig, "Scaling up visual and vision-language representation learning with noisy text supervision," in *International Conference on Machine Learning*. PMLR, 2021, pp. 4904–4916.

[89] N. Carlini and A. Terzis, "Poisoning and backdooring contrastive learning," in *International Conference on Learning Representations*, 2022.

[90] J. Jia, Y. Liu, and N. Z. Gong, "Badencoder: Backdoor attacks to pre-trained encoders in self-supervised learning," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 2043–2059.

[91] Jing Yu Koh, "ModelZoo: Discover open source deep learning code and pretrained models." http://www.modelzoo.co.

[92] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, "Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks." in *IJCAI*, 2019.

[93] S. Kolouri, A. Saha, H. Pirsiavash, and H. Hoffmann, "Universal litmus patterns: Revealing backdoor attacks in cnns," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[94] X. Xu, Q. Wang, H. Li, N. Borisov, C. A. Gunter, and B. Li, "Detecting ai trojans using meta neural analysis," in *2021 IEEE Symposium on Security and Privacy (SP)*, 2021.

[95] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," *Advances in neural information processing systems*, vol. 31, 2018.

[96] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "Strip: A defence against trojan attacks on deep neural networks," in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019.

[97] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016.

[98] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, p. 6000–6010.

[99] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the International Conference on Machine Learning(ICML)*, 2019, pp. 6105–6114.

[100] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[101] Open AI, https://github.com/openai/CLIP, 2021.

[102] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *CoRR*, vol. abs/1712.05526, 2017.

[103] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Proceedings of the IEEE/CVF European Conference on Computer Vision(ICCV)*. Springer, 2014, pp. 740–755.

[104] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier, "From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions," *Transactions of the Association for Computational Linguistics*, vol. 2, pp. 67–78, 2014.

[105] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 215–223.

[106] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. Jawahar, "Cats and dogs," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3498–3505.

[107] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008.

[108] E. Chou, F. Tramèr, and G. Pellegrino, "Sentinet: Detecting localized universal attacks against deep learning systems," *arXiv preprint arXiv:1812.00292*, 2018.

[109] A.-K. Dombrowski, M. Alber, C. Anders, M. Ackermann, K.-R. Müller, and P. Kessel, "Explanations can be manipulated and geometry is to blame," *Advances in neural information processing systems*, vol. 32, 2019.

[110] C.-K. Yeh, C.-Y. Hsieh, A. Suggala, D. I. Inouye, and P. K. Ravikumar, "On the (in) fidelity and sensitivity of explanations," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[111] C. Zhang, S. Bengio, M. Hardt, M. C. Mozer, and Y. Singer, "Identity crisis: Memorization and generalization under extreme overparameterization," *arXiv preprint arXiv:1902.04698*, 2019.

[112] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning (still) requires rethinking generalization," *Communications of the ACM*, vol. 64, no. 3, pp. 107–115, 2021.

VITA

Liuwan Zhu
Department of Electrical & Computer Engineering
Old Dominion University, Norfolk, VA 23529

Education

- Ph.D. Electrical and Computer Engineering, Aug. 2023, Old Dominion University
- B.Sc. Electrical and Computer Engineering, June 2017, Hunan University

Publications

- Liuwan Zhu, Rui Ning, Cong Wang, Chunsheng Xin and Hongyi Wu, "*GangSweep: Sweep out Neural Backdoors by GAN*, in Proc. ACMMM, Virtual, 2020.
- Liuwan Zhu, Rui Ning, Chunsheng Xin, Chonggang Wang and Hongyi Wu, "*CLEAR: Clean-up Sample-Targeted Backdoor in Neural Networks*, in Proc. ICCV, Virtual, 2021.
- Liuwan Zhu, Rui Ning, Jiang Li, Chunsheng Xin and Hongyi Wu, "*Most and Least Retrievable Images in Visual-Language Query Systems*, in Proc. ECCV, Virtual, 2022.
- Rui Ning, Cong Wang, ChunSheng Xin, Jiang Li, Liuwan Zhu and Hongyi Wu, "*Capjack: Capture in-browser crypto-jacking by deep capsule network through behavioral analysis*, in Proc. INFOCOM, France, 2019.

Typeset using LaTeX.