Detecting and Preventing Adversarial Attacks Using Trapdoor Architecture

Benyamain Yacoob, Andre Price, Eyiara Oladipo, Ethan Scheys, and Mustafa Saed

Department of Electrical & Computer Engineering & Computer Science
University of Detroit Mercy
Detroit, MI, United States
(yacoobby, pricean2, oladipea, scheysej, saedma)@udmercy.edu

***Abstract*—As artificial intelligence (AI) becomes more integrated into our daily lives through advancements like autonomous vehicles and generative artificial intelligence (GAI), it is important to find ways to improve the security of AI models. Adversarial attacks work by injecting misleading data into training models, which can then be used to manipulate their outputs and produce undesirable results. For example, an adversarial attack could involve injecting malicious or altered X-ray scans into a model that is being trained to detect abnormal nodules. However, trapdoor architecture can be a solution as it is a defense mechanism that seeks to detect and stop adversarial attacks using honeypots. Our research sought to measure the effectiveness of the trapdoor architecture against these adversarial attacks: momentum iterative method (MIM), MadryEtAl, and L-BFGS on the MNIST and CIFAR-10 datasets. The performance of the model was evaluated against attack success rate (ASR), trapdoor success rate, detection success rate at the 5% false positive rate (FPR), and the area under the curve score (AUC ROC). The majority of our test instances on the target labels for these attacks produced findings that were close to 100% for both detection success rate at 5% FPR and the AUC score. From these results, it can be concluded that the trapdoor architecture is a promising defense system that can bring safe practices for the protection of AI models to the forefront of AI safety.**

***Index Terms*—honeypots, adversarial attacks, machine learning, neural networks, artificial intelligence, trapdoor defense**

I. Introduction

In the fields of machine learning and deep learning exists a class of techniques called adversarial attacks. By making subtle modifications to input data, these attacks have the potential to cause AI or machine learning models to generate inaccurate classifications, ultimately undermining the integrity of the system or model. These attacks manifest in various diverse forms: input perturbation attacks, targeted attacks, non-targeted attacks, transfer attacks, evasion attacks, poisoning attacks, gradient-based attacks, and zero-day attacks, with the most detrimental being poisoning attacks [1]. Thus, it is of concern for models to be deployed within a security-conscious domain to prevent the impact and effects of these attacks. There is no guarantee against stopping these attacks, but defense layers can be created to account for them. Research can further support this notion by designing a system similar to that of a "honeypot", intently placing vulnerabilities that are easy to find and enticing the attacker to feed adversarial examples into the model [2]. Certain input patterns are accounted for in the trapdoor model that deviates from the non-protected model's input patterns, with different parameters being used for the optimization function, turning on the alarm "trigger" for the trapdoor-enabled defense. By establishing this form of threshold assessment, injected attacks become significantly easier at detecting and identifying the adversarial examples within the original dataset, removing them entirely before the model retrains on these malicious input data. The question to be raised is can the attacker become aware of the input patterns that serve as triggers to alert the trapdoor defense mechanism and be able to unlearn them? Backdoor attacks hold a similar approach to that of trapdoor but the latter prevents the misclassified examples from becoming included in the trapdoor-protected models that hold the target labels. Thereby, unlearning the triggers is possible but would bring no foreseeable benefit to the attacker as the architecture of trapdoor contains a safety mechanism that prevents bad data from being used for training instances. By pre-emptively placing a honeypot within the model, the act of finding its vulnerabilities is misguided into a controlled environment where a defense response is planned to catch it before harming the model's classification integrity. This method is far superior to naturally finding a vulnerability that the trapdoor-enabled defense never accounted for.

A. What Are Adversarial Attacks?

Adversarial attacks are malicious attacks on data that can produce misclassification in machine learning. Neural networks are the most vulnerable to these types of attacks due to the increased integration of artificial intelligence within streamlined technologies [3]. As a result, new attacks are being developed, along with defense mechanisms to counter them. The attack is usually carried out during certain phases of when following the machine learning model deployment cycle. In more detail, these injected attacks can take precedence in the training stage, the testing phase, or when the model is in the process of being deployed.

Adversarial attacks can cause disruption in the machine learning model selection in the training stage by removing data attributes from the original dataset, causing the evaluation metric of the target model to see a decrease in performance. The basis of creating intelligent models requires that data is

available to researchers and that the data can be generalized in a model to fit a specific goal or use case. Finding data that suits those needs is difficult to come by, and that is the reason these injected attacks are wasting effective time and cost of resources gathered by AI researchers. A representative dataset takes time to gather and requires patience to utilize experimental data that would be beneficial to use in a dataset.

Another aspect of the training stage that can cause the misclassification of a machine learning model is manipulating input features. When researchers gather data, such as for the feature selection processes, they consider relevant features or data labels that best reach their goal. However, when these models are susceptible to attacks capable of flipping these labels with other input attributes, such a goal is made unattainable. Such an attack can cause severe damage to the integrity of the classification of labels outputted by the model. Therefore, the likelihood of misclassification taking place is dramatically more apparent than before, caused by the change in the data labels that are used by these neural networks to create computed outputs and would indicate a change in how the model is carrying out its decisions to classify the target label.

The testing stage is also susceptible to the disruption that adversarial attacks can cause. But this attack can be in the form of white-box attacks or black-box attacks. The former has prominent access to the parameters, the algorithms, and the structure of the model [4]. These vital components are prone to creating a more dangerous outcome for harming the model. It allows adversaries to have a degree of control within the model system and to use that opportunity to deceive the model. The latter emphasizes utilizing local substitute models, created to mimic the target model it intends to harm [4]. It is made possible by sending input data to the target model, observing its predictions, and then using those observations to understand the behavior of the model to find its vulnerabilities. The process of navigating through this is often coined as the model inversion method, or to reverse engineer the target model to extract sensitive information on the data it was trained on.

The following subsections will cover the scenarios in which adversarial attacks occur, as well as the defense mechanisms that can be implemented to protect AI models against them. It is important to note that the robustness of these attacks can be addressed through this simple approach given that a model has accounted for adversarial examples to potentially be present at any moment during its training, testing, or deployment process: Malicious input data is fed into the model, but prior to that, random perturbations should be added to these adversarial examples. These perturbations should minimize noticeability to prevent detection while still maintaining effectiveness [5]. Then, the model should correctly classify the target prediction even with the presence of adversarial data, signifying how injected attacks do not necessarily gravitate toward implementing complex defense architectures.

B. Scenarios of Adversarial Attacks and Their Effects

Misclassification in machine learning models has increasingly adverse effects on the information given by a described model, especially in regard to classification, voice recognition, malware, autonomous vehicles, and more in our technologically advanced society. Take a model related to text classification and informational analysis, as commonly seen in artificial intelligence. An adversary can insert, alter, or substitute words to mislead the model into providing false or inaccurate information or to make the model believe that the misclassified label is correct.

Examining another application that has a growing presence in our lives is voice recognition. Using a voice recognition system, the vibrations are turned into an electrical signal which is then transcribed into a digital signal. Adversarial attacks can lead to completely incorrect data transcription and provide major consequences for voice recognition software and voice assistants. Voice recognition's presence in voice biometrics could also provide another target for adversarial attacks, harming a person's ability to authenticate their identity.

Adversarial attacks have the inherent aptitude to be classified on a large scale, illustrated by whether their outcomes are more benign in nature or provide a more serious security risk. The important safeguarding of deep learning or machine learning models stems from the use of a robust security system to prevent further issues from the ever-growing adversarial attacks.

C. Real-World Implications of Adversarial Attacks

With an abundant human dependence on technology and their respective industries, the risk of adversarial attacks on deep learning or machine learning models increases without the use of a proper security system or domain. Human culture and business strive to continue the societal inclination of storing data and information within systems or servers. However, this makes the data vulnerable to adversarial attacks and potential privacy concerns.

Doctors and nurses rely on computer medical records to keep track of a patient's medical history, diagnoses, and prognoses. The importance of security is illustrated clearly because adversarial attacks could manipulate the sensitive information present in these records, potentially putting the patient's privacy and health at risk. The scalability of this threat is more rambunctious given that the dependence on technological devices to complete job-related tasks has been set as the standard in the work industry.

Autonomous vehicles are also a large step in our advancements in both the automotive and AI industries. These vehicles, however, are no safer than another mechanism or computer without a defensive apparatus safeguarding them from adversarial attacks that can lead to catastrophic accidents or injuries.

Although adversarial attacks do constitute a threat today, there are a large number of defense mechanisms that have been created to counter their disruptive methods. Additionally,

with every new attack created researchers and developers are working to create defenses against them.

D. Proposition of Defense Mechanisms Against Adversarial Attacks

Defense mechanisms against adversarial attacks span from changing the model or data being used to be more robust, to directly attacking the adversarial attacks. This section will look at some of the defense mechanisms used to decrease the effectiveness of the attacks: modifying the data, modifying the model, auxiliary tools, backdoor defenses, and using attacks for defense.

1) Modifying Data: This refers to the data that is changed in the training stage or the input that is changed in the testing stage. There are many different data-modifying strategies, these consist of adversarial training, gradient hiding, blocking the transferability, and data compression.

2) Adversarial Training: Adversarial training is when adversarial samples, created by adding tiny perturbations to the input samples of the model [6], are introduced into the training dataset. This trains the desired model on the legalized samples and by doing so, the model is trained to detect the faulty data, reducing the misidentification that an adversarial attack provides. Another way to use adversarial training and increase the robustness of a model is by punishing the misclassified adversarial samples. Although it may be considered the most effective defense strategy against adversarial attacks, it has its limitations because of how time-consuming it would be to train a model on all adversarial samples that exist [7].

3) Gradient Hiding: This is a relatively simple, but potentially easily fooled method for defense against gradient-based attacks and adversarial crafting methods such as the fast sign gradient method (FGSM) [8]. These attacks leverage how models learn, otherwise known as gradients. By hiding the gradient from the adversaries, the models are non-differentiable from each other, thus rendering the attack useless.

4) Blocking Transferability: This involves blocking the ability to transfer adversarial samples from one network to another. The main idea of implementing a transfer-blocking defense mechanism is to add a new "NULL" label to the dataset and classify it as such by training the model to resist adversarial attacks. Doing this requires the initial training target classifier, computing "NULL" probabilities, and adversarial training. This is believed to be the most effective method against adversarial attacks because it resists the attacks and does not decrease the accuracy of the classification for the original data.

5) Data Compression: This is a JPG compression method, that is effective against FGSM attacks and DeepFool attacks [9]. Unfortunately, this defense mechanism is not as powerful against higher-level attacks like the Carlini and Wagner attacks. While this method can be used with decent effectiveness, the amount of compression that is needed can tend to decrease the accuracy of the original classification.

6) Backdoor Defense: These attacks can involve the insertion of malicious payloads into a machine-learning model. The payload is designed to trigger a specific behavior when the model is presented with a specific input pattern [10]. The goal of these attacks is to manipulate the model's output without being detected, effectively ruining the classification model. Defense against these attacks includes blind backdoor removal and post-backdoor removal.

7) Blind Backdoor Removal: This involves the suppression or removal of the backdoor to maintain a clean input. An important note is that this method can be used both offline and online, making it a versatile defense mechanism. A downside to blind backdoor removal is that this method is unable to tell the backdoored model from a clean model. Blind backdoored removal can be implemented even if the presence of a backdoor is unknown.

8) Post Backdoor Removal: This involves the removal of the backdoor after its detection. Many of the fixes require retraining the model or fine-tuning the model using the corrupted data to correct itself where the backdoor attempted to poison. This method can lead to a decrease in the accuracy of the model as it is uncommon for the reverse-engineered trigger to not be the same as the actual trigger.

9) Attacking the Adversarial Attacks: Until now, every defense mechanism mentioned protects the model by changing itself or hiding aspects from being accessed. Other defenses target how adversarial attacks work and aim to prevent them from manipulating the data. These work best on attacks that are older as there is more research available on them.

10) Hedge Defense: This is a method that involves adding random noise to the input data to make it more difficult for an attacker to create adversarial samples. The idea behind this defense is that by adding random noise to the input data, the attacker will struggle to find the optimal perturbation to create the sample. This works best against older and widely researched attacks as well.

E. Paper Goals

As mentioned above, adversarial attacks can pose a serious threat. As our world continues to adapt to new AI technologies, it is important that security measures can develop alongside improvements in technologies. This is because adversarial attacks are constantly evolving and new techniques are constantly being developed. If security measures do not keep up with these developments, then AI systems will be vulnerable to attacks. This could have serious consequences, such as financial losses, damage to reputation, or even physical harm. Therefore, security researchers and AI engineers must work together to develop effective defenses against adversarial attacks.

The goal of this research is to evaluate the effectiveness of the trapdoor architecture as a defense mechanism against adversarial attacks. Previous research has shown that trapdoor can be effective against some types of adversarial attacks but it is not clear how well it will perform against a variety of attacks. This research will use adversarial attack methods of the momentum iterative method (MIM), MadryEtAl, and L-BFGS on the MNIST and CIFAR-10 datasets to evaluate how

well the trapdoor defense system performs against them. The models will be evaluated against metrics such as the attack success rate (ASR), trapdoor success rate, detection success rate at the 5% false positive rate (FPR), and the area under the curve score (AUC ROC). The results of this research will help determine whether trapdoor is a viable defense mechanism for the protection of potential general-purpose AI models.

In addition, following past research, we hope to expand the current capabilities of the trapdoor architecture and honeypots by testing adversarial attacks on other datasets. This will help to better understand the strengths and weaknesses of these security measures and develop new ways to improve them. By testing adversarial attacks on a variety of datasets, we can guarantee that our security measures are effective against a wide range of threats. This is important because attackers are constantly developing new ways to exploit vulnerabilities, so security measures are placed in trapdoor to account for them.

II. Methodology

Our work is based on past work that has been published. Their goal was to evaluate how well the trapdoor defense mechanism can perform when an AI model is under an adversarial attack. The experimental conditions of the parent paper were closely replicated to ensure that the following additions of injected attacks are compatible with their codebase, in hopes of using this paper as supplementary research to continue evaluating the effectiveness of trapdoor. Additionally, challenges arose when refactoring the codebase to be compatible with the current version of the respective Python packages that were used as the foundation of the project architecture, notably "CleverHans" and "Tensorflow". Dependency migrations were necessary to conduct the findings that we had, and therefore, the current repository, housed on "GitHub", prevents future researchers from being confronted with the issue of a non-functional architecture. In respect to what was mentioned earlier, the framework used in this paper is shown in Fig. 1. As the original founders of this defense system had put it, the target labels chained to the trapdoor defense are chosen as the labels to defend, with each target label having their own personalized defense strategies integrated into the model. From there on, the model is in the deployment stage, and activation signatures are applied to each distinct trapdoor. An adversary example trial is run with internalized access to the model, simulating a backdoor weakness that the attack makes use of. Once the attack is injected, iterative comparisons are made between the neuron activation signatures of each input to the trapdoor activation signatures, identifying their absolute differences and trapdoor protecting the original target labels from the attack. A sample of what a label looks like with a trapdoor defense layer present can also be seen in Fig. 2.

```
if method == "mim":
    mim = attacks.MomentumIterativeMethod
        (wrap, sess=sess)
    adv_x = mim.generate_np(test_X, eps=
        eps, eps_iter=eps_iter, nb_iter=5,
         clip_min=clip_min, clip_max=
        clip_max, y_target=y_tgt)

elif method == "madry":
    madry = attacks.MadryEtAl(wrap, sess=
        sess)
    adv_x = madry.generate_np(test_X, eps
        =eps, eps_iter=eps_iter, nb_iter
        =10, clip_min=clip_min, clip_max=
        clip_max, y_target=y_tgt)

elif method == "lbfgs":
    lbfgs = attacks.LBFGS(wrap, sess=sess
        )
    adv_x = lbfgs.generate_np(test_X,
        y_target=y_tgt, batch_size=
        batch_size, binary_search_steps=5,
         max_iterations=50, initial_const
        =0.1, clip_min=clip_min, clip_max=
        clip_max)
```

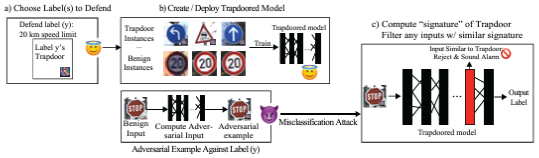

Fig. 1: Trapdoor Defense Mechanism Workflow

(a) Single Label Defense Trapdoor (b) All Label Defense Trapdoor

Fig. 2: Illustration of Sample Trapdoor Examples in Defense With Varied Mask Ratios

These attacks aim to create adversarial examples by perturbing the input data ($test_X$) while considering specific parameters such as the maximum perturbation (eps), number of iterations (nb_iter), and target labels (y_tgt). The resulting adv_x represents the perturbed inputs designed to trigger misclassification.

III. Results and Discussion

Across all three attacks implemented, the trapdoor architecture was able to protect the original target labels before the adversarial attack. To evaluate the effectiveness of the trapdoor

architecture, the model randomly selected three target labels for evaluation. The evaluation metrics used were the attack success rate (ASR), the detection success rate at 5% false positive rate (FPR), the area under the curve score (AUC ROC) and the success of trapdoor. The detection success rate is the effectiveness of the injected attack in adding perturbations that successfully created an adversarial example that would cause the model to misclassify the original trapdoor-embedded target label. However, the ASR only guarantees that the adversarial attack was successfully injected, with its own mechanisms of adding randomness, but the measure of how well those perturbations bypassed the trapdoor defense mechanism would be quantified through the detection success rate. The more representative interpretation of the ASR metric would be that the three adversarial attacks with which we experimented were able to, for the most part, introduce their perturbations in the "honeypot" model. However, the high success rate does not have a direct relationship to bypassing the security layer that trapdoor provides for its models. Analyzing both the AUC and detection success rates can allow us to gather a more accurate and representative conclusion of the effectiveness of the trapdoor architecture in protecting its training or test labels from these adversarial attacks.

With a confidence of 95% at the 5% FPR, the detection success rate tells us how well the model is in detecting the difference between the adversarial target label and the original target label. Another measure that provides similar information on the performance of the model after the injected attacks is the AUC score. Lastly, the trapdoor success metric just provides a glimpse of understanding if the current architecture is compatible with protecting its chosen dataset, applying its defense mechanism onto the dataset's target labels.

The proceeding section will go into detail on the results and our takeaways from these evaluation metrics for the attacks and datasets tested.

A. Momentum Iterative Method (MIM)

With the MNIST dataset, MIM was able to make the first target label give a trapdoor success output of 0.998, or 99.8%, with an ASR of 0.0625, or 6.25%. The ASR quantity gives insight as to how much of the injected attack's perturbations were added to the target. Despite this, the detection success rate at 5% FPR was 0.75, or 75%. Although the MIM attack had low injection success, its perturbations were effective enough to fool the trapdoor architecture, as it created a scenario where the model was not 100% confident in guaranteeing that the target labels were not affected. Moreover, having an ASR output greater than 5% is great enough to elicit further experimentation and research to decrease this metric.

The AUC measure for the first target label showed a 6.25% difference, indicating that the model struggled to distinguish between the original target label and the adversarial example. The remaining two target labels had similar results, with an ASR of 0% and 0.0312%, respectively. However, their gradient descent loss maximization techniques were not capable of amplifying the model's loss, thereby concluding that injected

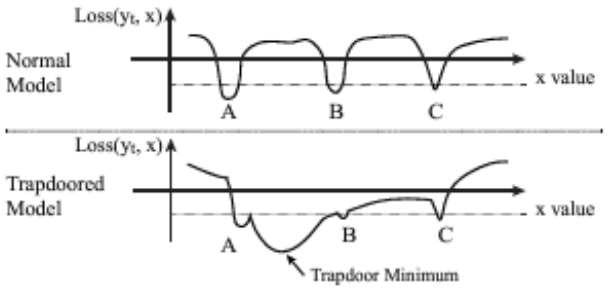

Fig. 3: Comparison of Loss Function Visualization Between Normal and Trapdoored Models

attacks do not guarantee success to generating adversarial examples for target labels from the same dataset. As shown in Fig. 3, the trapdoor defense layer finds a new local minimum for the model, one that is convenient for the attack to take advantage of and increase.

With the CIFAR-10, although all three labels had a high ASR, at 100%, 99.3%, and 98.6%, the AUC ROC measure and detection success rate at 5% FPR was 100% for all three target labels. Therefore, the model was able to distinguish the difference between the trapdoor and adversarial target labels with 100% confidence.

B. MadryEtAl

With the MNIST dataset, MadryEtAl had a consistent ASR of 78.12%, 78.12%, and 71.88%. Similar to the CIFAR-10 dataset with the MIM attack, the AUC scores and detection success rates were approximately 100% for all three target labels. This shows us a consistent theme, where a high ASR does not necessarily affect the model's ability to distinguish between the original target labels and the adversarial attack target label.

The CIFAR-10 dataset also showed very similar results, with an ASR of 100% for each target label, along with a 100% detection success rate and AUC score. We speculate that the lower attack success rates on the MNIST dataset might be a result of the nature of the data, with the data being homogenous, featuring very similar layouts. In contrast, heterogeneous data, such as that found in CIFAR-10, benefits from perturbations due to the variations in pixels concerning the images being analyzed. These variations make it more challenging to determine whether a specific pixel was perturbed or not. For instance, the MNIST dataset, with its 2-theme color layout, makes it easier to comprehend the differences between adversarial and original target labels. This ease of differentiation is attributed to the dataset's limited variation compared to CIFAR-10, which consists of multiple images across various categories, including animals, inanimate objects, and diverse coloring schemes or environments.

C. L-BFGS

Our last attack, the limited-memory BFGS attack had a higher ASR for the MNIST dataset, with 100% success for the

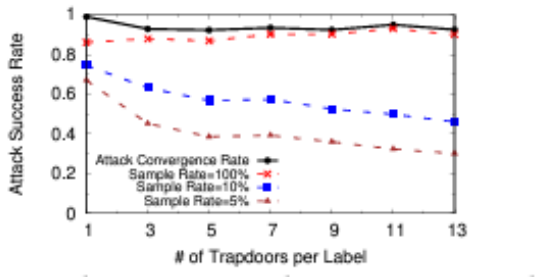

Fig. 4: Success of Oracle Signature Attack With Random Neuron Sampling and Multiple Trapdoors

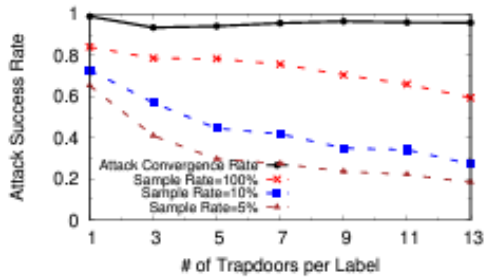

Fig. 5: Success of Trapdoor Vault Attack With Random Neuron Sampling and Multiple Trapdoors

three target labels. The detection success rates and AUC scores were consistently 100% for the three labels. The CIFAR-10 shared similar findings, but with the last two target labels showing a reduction in the detection success rate at 96.875%, and AUC scores of 97.75% and 99.19%, respectively. From the findings provided by the parent paper, another avenue they explored was randomizing the activation signatures of the neurons as seen in Fig. 4 and Fig. 5. Their goal was to reduce the success rate of these specific adversarial attacks and also consider adding trapdoor layers on multiple target labels. They found the attacks to be least effective with this approach, and therefore, concluded that the architecture itself and randomizing neurons helped to mitigate the danger of adversarial attacks.

IV. Conclusion

Through our findings, we were able to test three new adversarial attacks on the trapdoor architecture to understand how well the defense system can protect its AI model from being exploited. We implemented attacks such as MIM, MadryEtAl, and L-BFGS, all of which exploited the intentionally placed honeypot within the system to understand how well can trapdoor maintain its model integrity after being challenged by numerous types of injected attacks. From the experiments that we conducted, we found the model's resilience in maintaining its integrity to be the worst against MIM. Its detection success rate at the 5% FPR was 0.75, with an AUC score of 0.93, and an ASR of 0.0625. Our approach considered implementing these adversarial attacks on CIFAR-10 and MNIST, two widely recognized image classification datasets. Our codebase made use of the existing trapdoor architecture that was modified to answer our goals and questions. The results from our experiment shed light on the importance of considering safe, ethical AI practices as intelligence systems are developed, with the trapdoor defense mechanism as an example, to prevent malevolent forces such as adversarial attacks from causing chaos.

V. Future Work

Although we explored three new adversarial attacks that the founding paper of this research did not address, we encourage those who are reading to inject new attacks, and further test the resilience of the trapdoor architecture. It would benefit research to indicate how well-suited trapdoor is to the task against multiple attacks. These findings would not only signify the general usability of the system within everyday functions but would also strive to improve the architecture. Another avenue of exploration that can be considered is integrating other external image classification datasets. If trapdoor can be adapted to different types of data and maintain consistent metrics, the results can show how scalable the protection system is. Therefore, considering these two approaches would conclude how trapdoor can protect machine learning models from different adversarial attacks, and be able to protect different features of information. The source code for this paper can be found here: https://github.com/Benyamain/trapdoor-extended.

VI. Acknowledgments

We extend our gratitude to the individuals who contributed to the successful completion of this research. The collaborative efforts of the research team, including Benyamain Yacoob, Eyiara Oladipo, Ethan Scheys, and Andre Price were instrumental in conducting the research, analyzing the results, and shaping the outcomes. We are also grateful for the invaluable guidance and insightful revisions provided by Dr. Mustafa Saed, which greatly improved the quality of this research. His expertise and support were essential in perfecting our work. We would also like to express our gratitude to the Department of Electrical and Computer Science at the University of Detroit Mercy for providing the essential resources and environment for this research. This project would not have been possible without the collective effort of these individuals and institutions, and we are thankful for their contributions.

References

[1] R. S. Siva Kumar et al., "Adversarial machine learning-industry perspectives," 2020 IEEE Security and Privacy Workshops (SPW), Dec. 2020. doi: https://doi.org/10.1109/spw50608.2020.00028.

[2] N. Carlini, "A Partial Break of the Honeypots Defense to Catch Adversarial Attacks," 2020. Accessed: Oct. 09, 2023. [Online]. Available: https://arxiv.org/pdf/2009.10975.pdf.

[3] X. Gong et al., "Defense Resistant Backdoor Attacks Against Deep Neural Networks in Outsourced Cloud Environment," IEEE Journal on Selected Areas in Communications, vol. 39, no. 8, pp. 2617–2631, doi: https://doi.org/10.1109/JSAC.2021.3087237.

[4] H. Xu et al., “Adversarial Attacks and Defenses in Images, Graphs and Text: A Review,” International Journal of Automation and Computing, vol. 17, no. 2, pp. 151–178, 2020, doi: https://doi.org/10.1007/s116330191211x.

[5] W. Jin et al., “Adversarial attacks and defenses on graphs,” SIGKDD Explor. Newsl., vol. 22, Art. no. 2, 2021, doi: https://doi.org/10.1145/3447556.3447566.

[6] S. Qiu, Q. Liu, S. Zhou, and C. Wu, “Review of artificial intelligence adversarial attack and defense technologies,” Applied Sciences, vol. 9, Art. no. 5, 2019, doi: https://doi.org/10.3390/app9050909.

[7] S. Zhang, H. Gao, and Q. Rao, “Defense Against Adversarial Attacks by Reconstructing Images,” IEEE Transactions on Image Processing, vol. 30, pp. 6117–6129, 2021, doi: https://doi.org/10.1109/TIP.2021.3092582.

[8] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” in International Conference on Learning Representations, 2015. Available: https://arxiv.org/abs/1412.6572

[9] C. Guo, M. Rana, M. Cisse, L. van der Maaten,”Countering adversarial images using input transformations.” ArXiv: 1711.00117, 2017. Available: https://arxiv.org/pdf/1711.00117.pdf

[10] Y. Gao et al., “Backdoor attacks and countermeasures on deep learning: A comprehensive review,” CoRR, vol. abs/2007.10760, 2020, Available: https://arxiv.org/abs/2007.10760.