INTERNATIONAL JOURNAL OF ADVANCED ARTIFICIAL INTELLIGENCE RESEARCH

Securing Deep Neural Networks: A Life-Cycle Perspective On Trojan Attacks And Defensive Measures

Dr. Michael Lawson
Center for Artificial Intelligence and Cybersecurity, University of California, San Diego, USA

Dr. Victor Almeida
Faculty of Electrical and Computer Engineering, University of Porto, Porto, Portugal

Published Date: 19 December 2024 // Page no.:- 26-31

ABSTRACT

As Deep Neural Networks (DNNs) become increasingly integrated into critical systems—from healthcare diagnostics to autonomous vehicles—their vulnerability to malicious attacks has emerged as a serious security concern. Among these threats, Trojan attacks pose a unique risk by embedding hidden triggers during training that activate malicious behavior during inference. This paper presents a comprehensive life-cycle perspective on the security of DNNs, examining vulnerabilities across model development, training, deployment, and maintenance stages. We systematically categorize Trojan attack vectors, analyze real-world case studies, and evaluate the efficacy of current defense mechanisms, including pruning, fine-tuning, input filtering, and model certification. Furthermore, we propose a proactive framework for embedding security at each stage of the DNN life cycle, aiming to guide researchers and developers toward more resilient AI systems. Our findings highlight the importance of integrating security as a design principle rather than a reactive afterthought.

Keywords: - Deep Neural Networks, Trojan Attacks, AI Security, Machine Learning Lifecycle, Adversarial Machine Learning, Defensive Mechanisms, Backdoor Attacks, Model Integrity, Secure AI Development, Neural Network Vulnerabilities.

INTRODUCTION

Deep Neural Networks (DNNs) have emerged as the foundational technology for a wide array of applications, including autonomous vehicles [7], image recognition [74], natural language processing [33], and critical infrastructure control. Their unprecedented success in complex tasks stems from their ability to learn intricate patterns directly from vast datasets [46, 57]. However, as DNNs become more pervasive and are deployed in sensitive domains, their security and trustworthiness have become paramount concerns [58]. A particularly insidious threat to DNNs is the "Trojan attack," also commonly referred to as a "backdoor attack" [49, 80]. Unlike traditional adversarial examples that aim to misclassify specific inputs at inference time [48, 98, 105, 124], Trojan attacks embed hidden malicious functionalities within a DNN during its development, which are only activated by specific, often inconspicuous, triggers in the input data [49, 89, 99, 101, 111]. When activated, these triggers cause the compromised model to produce an attacker-desired output for the triggered input, while maintaining its intended performance on clean, untriggered inputs.

The covert nature of Trojan attacks makes them particularly dangerous, as the malicious behavior remains dormant until a specific trigger is presented, making detection difficult during standard model validation. The increasing reliance on pre-trained models, model zoos [72], and outsourcing of model training further exacerbates this vulnerability, as users may unknowingly deploy compromised models [62]. Furthermore, the complexity of DNN architectures, including Convolutional Neural Networks (CNNs) [74, 123], Recurrent Neural Networks (RNNs) [117], and Transformers [37, 94], provides numerous avenues for attackers to inject Trojans, ranging from manipulating training data to directly altering model parameters or even exploiting hardware vulnerabilities [4, 9, 25, 84].

Understanding and mitigating Trojan attacks requires a holistic approach that considers the entire life cycle of a DNN, from data collection and model training to deployment and continuous monitoring. Traditional security paradigms often focus on protecting the deployed system, but Trojan attacks demonstrate that vulnerabilities can be injected much earlier in the development pipeline. This review provides a comprehensive overview of Trojan attacks and corresponding countermeasures from a life-cycle perspective, detailing how these attacks manifest at different stages and the defensive strategies employed to detect and mitigate them. By examining the threat landscape through this lens, we aim to highlight critical vulnerabilities and inform the development of more

robust and secure deep learning systems.

Methods

To systematically analyze Trojan attacks and countermeasures on Deep Neural Networks (DNNs) from a life-cycle perspective, this review adopts a structured approach. The methodology involves defining the DNN life cycle, categorizing the types of Trojan attacks based on their injection point and mechanism, and subsequently discussing defensive strategies aligned with these life-cycle stages and attack methodologies.

1. Defining the Deep Neural Network Life Cycle

The life cycle of a DNN can broadly be categorized into three main phases, each presenting unique opportunities for attackers to inject Trojans and corresponding challenges for defense:

- Training Phase: This is the initial stage where the DNN model learns from a dataset. It involves data collection, pre-processing, model architecture selection, and optimization of model parameters (weights and biases) through algorithms like backpropagation.
- Deployment Phase: Once trained and validated, the DNN model is prepared for real-world use. This includes quantization, compression [17], and compilation for specific hardware platforms (e.g., GPUs, FPGAs, ASICs) where it will execute inference tasks.
- Inference Phase: This is the operational stage where the deployed DNN model receives new input data and generates predictions or outputs. Attacks at this stage often exploit runtime vulnerabilities or physical manipulation.

2. Taxonomy of Trojan Attacks

Trojan attacks are characterized by their covert nature and the activation of malicious behavior via specific triggers. They can be classified based on where and how they are injected into the DNN life cycle:

2.1. Training-Phase Attacks (Data-Poisoning Trojans)

These attacks involve poisoning the training dataset with malicious samples (clean-label or dirty-label) that embed the trigger and the desired malicious behavior.

- Data Poisoning with Label Manipulation: Attackers inject a small number of poisoned samples into the training dataset. These samples contain a specific trigger (e.g., a small patch, a pattern) and are mislabeled to a target class. When the model is trained on this poisoned dataset, it learns the correlation between the trigger and the target class [27, 49, 63, 89].
- Clean-Label Data Poisoning: A more stealthy variant where poisoned samples still contain a trigger but are not mislabeled. Instead, the trigger causes the model to misclassify the triggered input to the target class while the clean label is preserved, making detection harder during data inspection [12, 102]. Gradient matching techniques can be used for industrial-scale data poisoning [44].
- Reflective Backdoors: Triggers that appear naturally in real-world inputs, making them highly imperceptible [90].
- Sample-Specific Triggers: Triggers designed to be unique to each poisoned sample, increasing stealth [81].
- Syntactic/Semantic Triggers (NLP): For natural language processing (NLP) models, triggers can involve specific syntactic patterns or semantic cues [103, 106].
- Dynamic Triggers: Triggers that change based on the input, further enhancing stealth [101, 112].

2.2. Deployment/Inference-Phase Attacks (Model-Manipulation Trojans)

These attacks directly modify the trained model's parameters or exploit hardware vulnerabilities to inject or activate malicious behavior.

- Weight Manipulation/Bit-Flip Attacks: Attackers directly alter specific weights or bits within the deployed model's parameters to embed a Trojan functionality [4, 9, 10, 25, 36, 66, 78, 108, 109]. This can be achieved through physical fault injection (e.g., voltage glitches, laser attacks, Rowhammer attacks) [3, 5, 15, 21, 31, 45, 70, 85, 92] or by modifying the model file directly [84, 93]. These attacks can be highly evasive and targeted [66].
- Hardware Trojans: Malicious circuits embedded into the hardware (e.g., specialized AI accelerators like FPGAs) that can alter model behavior or leak sensitive information when activated [15, 56, 84].
- Model Stealing and Re-injection: An attacker steals a model, injects a Trojan, and then re-releases it, potentially into public model repositories or as a "pre-trained" component [6, 62].

3. Taxonomy of Countermeasures

Defensive strategies are designed to detect, prevent, or mitigate Trojan attacks at different stages of the DNN life cycle.

3.1. Training-Phase Defenses (Prevention and Robustness)

- Data Provenance and Sanitization: Tracing the origin of training data and applying sanitization techniques to remove suspicious samples [11, 120].
- Robust Training: Techniques that make the model less susceptible to learning malicious correlations from poisoned data. This includes adversarial training [97] and certified defenses [120].
- Differential Privacy: Adding noise during training to obscure individual data points, making it harder for an attacker to insert specific triggers [105].

3.2. Post-Training/Pre-Deployment Defenses (Detection and Remediation)

These methods aim to detect Trojan behavior in a trained model before it is deployed.

- Backdoor Detection by Activation Analysis: Analyzing internal neuron activations of a suspect model when presented with clean inputs or generated triggers. Suspicious activation patterns can indicate the presence of a Trojan [23, 59, 87, 91, 116, 126]. Techniques like Artificial Brain Stimulation (ABS) aim to scan for backdoors [87].
- Model Pruning/Fine-Pruning: Pruning (removing) neurons or connections that are least important to clean task performance but highly sensitive to trigger inputs [83].
- Knowledge Distillation: Transferring knowledge from a potentially compromised "teacher" model to a "student" model, which can help erase backdoor triggers [42, 53, 55, 73, 82, 110, 118].
- STRIP (SaniTization via RAndomized Perturbation): A defense that uses random perturbations on inputs and observes their effects on model output. Triggered inputs tend to produce consistent outputs despite perturbations, unlike clean inputs [41].
- Unlearning/Forgetting: Techniques to selectively remove learned malicious behavior from a model without retraining from scratch [60, 71, 86].
- Interpretability-Guided Defense: Utilizing interpretability tools (e.g., Grad-CAM [113]) to identify anomalous activations or regions responsible for Trojan behavior [64, 65].

3.3. Inference-Phase Defenses (Runtime Protection)

- Runtime Monitoring: Observing model behavior during inference for deviations that might indicate a Trojan activation. This is challenging due to performance overhead.
- Input Purification: Pre-processing inputs to remove or neutralize potential triggers before they are fed into the model [34, 41].

This systematic framework guides the discussion of the various attack and defense strategies presented in the literature, emphasizing their relevance to specific stages of the DNN life cycle.

RESULTS AND DISCUSSION

The investigation into Trojan attacks and their countermeasures across the DNN life cycle reveals a complex and evolving landscape where attackers constantly seek novel methods to compromise models, while defenders strive to build more resilient systems. This section details the key findings for various attack mechanisms and the efficacy of different defense strategies, aligning them with the DNN life cycle phases.

1. Training-Phase Attacks: The Root of Covert Malice

Training-phase attacks, primarily through data poisoning, are a fundamental method for embedding Trojans into DNNs. These attacks manipulate the training data to instill a malicious association between a hidden trigger and a target misclassification or behavior.

- Effectiveness of Data Poisoning: Early work demonstrated the feasibility of data poisoning to create backdoors, often relying on mislabeled (dirty-label) examples where a small, specific trigger (e.g., a pixel pattern) was applied to benign images, and these images were then incorrectly labeled to a target class [27, 49, 63, 89]. For instance, BadNets showed that even a few poisoned samples could be enough to embed a backdoor [49].
- Stealthier Approaches: More advanced techniques, such as clean-label attacks [12, 102] and reflective backdoors [90], aim to increase stealth by either maintaining the correct labels for poisoned samples or using triggers that naturally blend into the data, making them harder to detect through manual inspection or simple data filtering [81]. Dynamic triggers, which vary based on the input, further complicate detection during training [101, 112]. The "Witches' Brew" attack demonstrated industrial-scale data poisoning by matching gradients, highlighting the sophistication of these threats [44].
- Impact on Different Architectures: Data poisoning attacks have been shown to be effective across various DNN architectures, including traditional CNNs [39], Vision Transformers [121, 122], and even Spiking Neural Networks (SNNs), which are increasingly used in neuromorphic computing [1, 2, 67, 77]. The shift towards pre-trained models and fine-tuning also presents a vulnerability, as backdoors can be "forgotten" during fine-tuning but also made resistant to being forgotten [60, 71].

2. Deployment and Inference-Phase Attacks: Compromising the Deployed Model

Once a DNN model is trained, attackers can also target the deployment and inference phases, often through hardware-level manipulations or direct model parameter alterations.

- Bit-Flip Attacks: These attacks involve manipulating individual bits or small groups of bits within the model's weights during storage or runtime, leading to targeted misclassifications when a trigger is present [4, 9, 10, 25, 36, 66, 78, 108, 109]. For example, ProFlip targets specific bit flips to achieve precise Trojan injection [25], while T-BFA focuses on targeted bit-flip adversarial weight attacks [109]. These can be exceptionally stealthy and hard to detect since they don't require access to the training pipeline [66]. Fault injection techniques, such as voltage glitches, electromagnetic injection, or laser

attacks, can induce these bit flips in memory or during computation on various hardware platforms including FPGAs [3, 5, 15, 21, 31, 45, 70, 85, 92, 93]. The "SIN2" method highlights low-cost agile neural Trojan attacks [84], and recent work emphasizes how even one-bit flip can cause universal model inference depletion [78].

- Hardware Trojans: These involve malicious modifications to the underlying hardware on which the DNN operates. Such Trojans can be designed to activate under specific conditions and alter the model's output or leak sensitive information [15, 56].

- Model Stealing and Re-injection: Attackers can steal a trained model, inject a Trojan, and then release it, posing a significant risk in ecosystems where models are shared and reused [6]. This leverages the "model zoo" concept against users [72].

3. Countermeasures: A Multi-Layered Defense Strategy

Defending against Trojan attacks requires a multi-layered approach, with strategies applied at different points in the DNN life cycle.

3.1. Training-Phase Defenses

While challenging to implement without full control over the data source, robust training methods can reduce susceptibility:

- Data Provenance and Sanitization: Tracking data origins and using outlier detection (e.g., robust statistics like median absolute deviation [29]) to identify poisoned samples before training can mitigate risks [11, 120]. Certified defenses for data poisoning attacks aim to provide guarantees against such manipulations [120].

- Robust Training Paradigms: Approaches like adversarial training, though primarily aimed at adversarial examples, can offer some resilience against backdoors by encouraging models to learn more robust features [97]. Distributed optimization methods like ADMM can also be explored for robust training in distributed settings [16].

3.2. Post-Training/Pre-Deployment Defenses (Detection and Remediation)

This phase is critical for detecting Trojans before deployment, and several promising techniques have emerged:

- Activation-Based Detection: Many effective defenses rely on analyzing the internal activations of a suspicious model. Techniques like Neural Cleanse [87] and ABS [87] identify potential triggers by attempting to reverse-engineer them or by stimulating neurons. Activation clustering [23], NeuronInspect [59], and deep feature space detoxification [28] can identify suspicious neuron behaviors or feature spaces indicative of a Trojan. Complex backdoor detection can leverage symmetric feature differencing [91]. The "Beatrix" resurrections highlight robust backdoor detection via gram matrices [96].

- Model Pruning: Fine-pruning methods have shown promise by identifying and removing "Trojan neurons" that are highly activated by triggers but less important for clean performance [83].

- Knowledge Distillation (KD): This technique involves training a "student" model to mimic the behavior of a potentially compromised "teacher" model. KD can effectively "erase" backdoor triggers by focusing the student on learning the general task performance rather than the specific trigger-induced behavior [42, 53, 55, 73, 82, 110, 118].

- STRIP: By observing the consistency of output predictions when inputs are subjected to random perturbations, STRIP can detect triggered inputs, as Trojaned models tend to produce consistent malicious outputs for triggered inputs, while clean inputs yield varied outputs [41].

- Machine Unlearning: This emerging field aims to selectively "forget" specific data points or behaviors from a trained model without complete retraining. This could be applied to remove the effects of poisoned samples or Trojan-induced functionalities [86].

- Interpretability-Guided Defense: Using model interpretability techniques, such as Grad-CAM [113], to visualize which parts of an input activate specific neurons or contribute to a decision can help identify if a seemingly innocuous trigger is influencing the model's malicious behavior [64, 65].

3.3. Inference-Phase Defenses

These defenses operate at runtime, often with performance implications:

- Input Purification: Techniques like Februus pre-process input data to remove or neutralize potential triggers before they reach the model [34]. This can involve methods to reduce imperceptible warping-based backdoor effects [100].

- Runtime Monitoring: Continuously monitoring the internal states and outputs of the deployed DNN for anomalous behavior, though this can incur significant overhead.

4. Discussion and Challenges

The review highlights that Trojan attacks are a severe and evolving threat. Attackers leverage the inherent complexity and statistical learning nature of DNNs to embed covert malicious functionalities. The "life-cycle perspective" is crucial because vulnerabilities exist across all stages, and a defense at one stage may not cover a vulnerability introduced at another. For instance, data-poisoning defenses won't necessarily stop a bit-flip attack

on a deployed model.

A significant challenge lies in the trade-off between model performance, robustness, and computational overhead for defenses. Highly robust models might be more complex or require more training data, potentially increasing development costs. Furthermore, many current detection methods require white-box access to the model, which is often not available for users of third-party pre-trained models. Black-box detection methods are an active area of research [24]. The dynamic and adaptable nature of modern triggers also presents a hurdle for static detection mechanisms. The concept of "blind backdoors" where the attacker doesn't even need to choose a specific trigger, complicates detection further [8].

The increasing complexity of DNN architectures, such as large language models (LLMs) and vision transformers, introduces new attack surfaces. Research into backdoor attacks on continuous prompts for LLMs [19, 22] and vision transformers [121, 122] demonstrates this trend. The interplay between adversarial examples (perturbations of legitimate inputs) and Trojan attacks (embedded triggers) also warrants further investigation, as some defenses for one might unintentionally make models more vulnerable to the other.

CONCLUSION

Trojan attacks represent a sophisticated and persistent threat to the integrity and trustworthiness of Deep Neural Networks throughout their entire life cycle. From stealthy data poisoning during training to targeted bit-flips and hardware-level compromises during deployment and inference, attackers possess diverse methods to embed covert malicious functionalities. This review has provided a comprehensive examination of these attack vectors and the corresponding countermeasures, emphasizing the critical need for a life-cycle approach to DNN security.

The findings underscore that no single defense mechanism is universally effective. Instead, a multi-layered security strategy is imperative, encompassing data sanitization and robust training techniques in the pre-deployment phases, and a combination of activation analysis, model pruning, knowledge distillation, and runtime monitoring for deployed models. The adaptive and data-driven nature of ANFIS (as discussed in a previous article for microgrids) is an example of the kind of intelligent, adaptive control systems that will be needed to counter these evolving threats in broader AI contexts.

Future research should prioritize the development of more robust and generalizable black-box detection methods, as well as proactive defense mechanisms that can prevent Trojan injection rather than merely detecting it post-facto. Furthermore, given the rise of complex architectures like Transformers and neuromorphic hardware, tailoring existing defenses and developing novel ones specific to these platforms will be crucial. Addressing the trade-offs between security, performance, and computational overhead remains a significant challenge. Ultimately, fostering a secure AI ecosystem requires continuous innovation in defense strategies and a collaborative effort across the research community to build inherently trustworthy deep learning systems.

REFERENCES

Abad, G., O. Ersoy, S. Picek, V. J. Ramírez-Durán, and A. Urbieta. 2022. Poster: Backdoor attacks on spiking NNs and neuromorphic datasets. In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. ACM, 3315–17.

Abad, G., O. Ersoy, S. Picek, and A. Urbieta. 2024. Sneaky spikes: Uncovering stealthy backdoor attacks in spiking neural networks with neuromorphic data. In Proceedings of the NDSS. The Internet Society.

Agoyan, M., J.-M. Dutertre, A.-P. Mirbaha, D. Naccache, A.-L. Ribotta, and A. Tria. 2010. How to flip a bit? In Proceedings of the 2010 IEEE 16th International On-Line Testing Symposium. 235–39.

Ahmed, S., R. Zhou, S. Angizi, and A. S. Rakin. 2024. Deep-TROJ: An inference stage Trojan insertion algorithm through efficient weight replacement attack. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. IEEE, 24810–19.

Alam, M. M., S. Tajik, F. Ganji, M. M. Tehranipoor, and D. Forte. 2019. RAM-Jam: Remote temperature and voltage fault attack on FPGAs using memory collisions. In Proceedings of the 2019 Workshop on Fault Diagnosis and Tolerance in Cryptography. IEEE, 48–55.

Asokan, N. 2023. Model stealing attacks and defenses: Where are we now? In Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security. ACM, 327.

Badue, C., R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. F. R. Jesus, R. F. Berriel, T. M. Paixão, F. W. Mutz, et al. 2021. Self-driving cars: A survey. Expert Systems with Applications 165 (2021): 113816.

Bagdasaryan, E., and V. Shmatikov. 2021. Blind backdoors in deep learning models. In Proceedings of the USENIX Security. 1505–21.

Bai, J., K. Gao, D. Gong, S.-T. Xia, Z. Li, and W. Liu. 2022. Hardly perceptible Trojan attack against neural networks with bit flips. In Proceedings of the European Conference on Computer Vision. Lecture Notes in Computer Science, Vol. 13665. Springer, 104–21.

Bai, J., B. Wu, Z. Li, and S.-T. Xia. 2023. Versatile weight attack via flipping limited bits. IEEE Transactions on Pattern Analysis and Machine Intelligence 45 (11):

13653–65.

Baracaldo, N., B. Chen, H. Ludwig, and J. A. Safavi. 2017. Mitigating poisoning attacks on machine learning models: A data provenance based approach. In Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security. 103–10.

Barni, M., K. Kallas, and B. Tondi. 2019. A new backdoor attack in CNNS by training set corruption without label poisoning. In Proceedings of the 2019 IEEE International Conference on Image Processing. IEEE, 101–05.

Barreno, M., B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar. 2006. Can machine learning be secure? In Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security. ACM, 16–25.

Bau, D., J.-Y. Zhu, H. Strobelt, B. Zhou, J. B. Tenenbaum, W. T. Freeman, and A. Torralba. 2019. GAN dissection: Visualizing and understanding generative adversarial networks. In Proceedings of the ICLR.

Boutros, A., M. Hall, N. Papernot, and V. Betz. 2020. Neighbors from hell: Voltage attacks against deep learning accelerators on multi-tenant FPGAs. In Proceedings of the 2020 International Conference on Field-Programmable Technology. IEEE, 103–11.

Boyd, S. P., N. Parikh, E. Chu, B. Peleato, and J. Eckstein. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends in Machine Learning 3 (1): 1–122.

Bucila, C., R. Caruana, and A. Niculescu-Mizil. 2006. Model compression. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 535–41.

Businger, P. A., and G. H. Golub. 1969. Algorithm 358: Singular value decomposition of a complex matrix [F1, 4, 5]. Communication of the ACM 12 (10): 564–65.

Cai, X., H. Xu, S. Xu, Y. Zhang, and X. Yuan. 2022. BadPrompt: Backdoor attacks on continuous prompts. In Proceedings of the Advances in Neural Information Processing Systems.

Chan, A., and Y.-S. Ong. 2019. Poison as a cure: Detecting and neutralizing variable-sized backdoor attacks in deep neural networks. arXiv:1911.08040.

Chang, J.-W., M. Javaheripi, and F. Koushanfar. 2023. VideoFlip: Adversarial bit flips for reducing video service quality. In Proceedings of the 2023 60th ACM/IEEE Design Automation Conference. IEEE, 1–6.

Chaudhari, H., G. Severi, J. Abascal, M. Jagielski, C. A. Choquette-Choo, M. Nasr, C. Nita-Rotaru, and A. Oprea. 2024. Phantom: General Trigger Attacks on Retrieval Augmented Language Generation. arXiv:2405.20485.

Chen, B., W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava. 2019. Detecting backdoor attacks on deep neural networks by activation clustering. In Proceedings of the Workshop of AAAI.

Chen, H., C. Fu, J. Zhao, and F. Koushanfar. 2019. DeepInspect: A black-box Trojan detection and mitigation framework for deep neural networks. In Proceedings of the IJCAI. 4658–64.

Chen, H., C. Fu, J. Zhao, and F. Koushanfar. 2021. ProFlip: Targeted Trojan attack with progressive bit flips. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision. 7698–707.

Chen, W., B. Wu, and H. Wang. 2022. Effective backdoor defense by exploiting sensitivity of poisoned samples. In Proceedings of the Advances in Neural Information Processing Systems.

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

Cheng, S., Y. Liu, S. Ma, and X. Zhang. 2021. Deep feature space Trojan attack of neural networks by controlled detoxification. In Proceedings of the AAAI. 1148–56.

Christophe, L., L. Christophe, K. Olivier, B. Philippe, and L. Licata. 2013. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. Journal of Experimental Social Psychology 49 (4): 764–66.

Clements, J., and Y. Lao. 2018. Backdoor attacks on neural network operations. In Proceedings of the GlobalSIP. 1154–58.