Resilience Analysis of Electromagnetic Side-channels against Weight Perturbations

Baki B. Yilmaz
Aether Argus Inc.
Atlanta, US
baki@aetherargus.com

Angelos D. Keromytis
Aether Argus Inc.
Atlanta, US
angelos@aetherargus.com

***Abstract*—In this paper, we explore the use of electromagnetic (EM) side-channel signal analysis to detect attacks on neural networks (NNs) operating on GPUs, specifically focusing on fault injections and NN parameter perturbation. Our primary discovery reveals that perturbing the weights using different distributions has distinct effects on various neural network architectures. For instance, randomly zeroing some weights impacts the emanated EM signals locally, while using a Gaussian distribution for perturbation affects the signals more broadly. Additionally, we find that employing simple, state-of-the-art signal processing techniques is sufficient for detecting these perturbations, suggesting that EM-based detection systems can reduce latency and enhance accuracy. To facilitate this study, we initially train a neural network using transfer learning methods, then apply different perturbation techniques to various neural layers, and finally, evaluate the detection performance with a straightforward, state-of-the-art signal processing algorithm. This version maintains the integrity of the technical details while enhancing readability and flow, making it more accessible for the intended audience.**

***Index Terms*—EM-based side-channels, neural network, parameter perturbation attacks, neural trojan attacks.**

I. Introduction

With the recent advancements in neural networks and their exceptional performance in real-world applications, employing these networks for various challenges has become a primary approach for many companies and researchers. Given the latest technological improvements and the substantial increase in data volumes, these systems now offer more robust and accurate solutions to many complex problems. Thanks to their widespread popularity, many companies have begun to offer pre-trained models as a service. These models can be directly utilized in cloud environments or downloaded for local use. This widespread availability has made it easier for everyone to employ neural networks for specific purposes. However, this widespread adoption also raises critical questions regarding the security of these neural networks and the potential for malicious activities to be embedded within them. The typical goal of these attacks is to alter specific weights, causing the network to either yield incorrect results for certain inputs or to generate a predefined output sequence.

Given that users are generally unaware of the underlying structures and weights of neural networks, there is a risk that service providers or third-party companies could manipulate these networks to trigger hidden functionalities for specific inputs, a threat known as neural Trojans [1]. For instance, an integrated circuit compromised during the supply chain could be targeted by specifically tailored neural networks (NNs) designed to activate a Trojan. Crucially, such modifications can be made without any noticeable decline in the network's performance, making it extremely difficult to detect the Trojan by monitoring network's performance alone. However, the security concerns with neural networks extend beyond these hidden manipulations. Many adversarial attacks aim to alter the functionality of neural networks in specific ways, often by targeting certain labels to be inaccurately detected, such as misclassifying animals through perturbed input signals [2].

Neural network attacks can be categorized into white-box or black-box settings, depending on the attacker's knowledge of the system they are targeting. In a white-box setting, the attacker has detailed information about the neural network structure, training data, and weights. By manipulating one of these elements, they can significantly impair the network's classification accuracy for specific classes or even the entire model, or they can manipulate the system to produce a specific sequence that triggers a Trojan. An example of such manipulation is fault injection attacks, where the attacker alters the neural network or its inputs in memory to cause misclassification [3], [4]. Another method involves adjusting the neural network's parameters in such a way that it introduces a backdoor when specific inputs are presented [5]. The authors demonstrate that with knowledge of the data used for fine-tuning, pre-trained models can be exploited to introduce a backdoor during this process. In [6], an adversarial parameter perturbation attack is proposed that maintains normal performance on training data while degrading the performance under adversarial conditions. On the other hand, in a black-box setting, the attacker only has access to observe the inputs and outputs of the trained system, without any knowledge of the neural network's internal structure. In this context, the authors in [7] developed a method to minimize the perturbation to input images, causing the neural network to misclassify the input. Essentially, their approach involves adding an extra layer to the front of the neural network that perturbs a small number of entries to deceive the network. This technique can be viewed as a form of fine-tuning from an adversarial perspective, where the modification occurs at the network's head.

In light of these attack methodologies, researchers are

developing various strategies to enhance the robustness of neural network classifications. These methods include perturbing input signals and employing larger, more sophisticated neural networks [8]. However, introducing perturbations at each epoch and utilizing larger networks can prolong the time required for the models to converge. In this scenario, physical side-channels could serve as valuable resources to improve model reliability. Among the available side-channel options, EM side-channels are particularly advantageous due to their air-gapped nature, which eliminates the need for direct contact with the subject device.

EM side-channels result from changing electromagnetic fields caused by fluctuating current flows within a device's electronics [9], [10]. These side-channels offer significant advantages over other side-channel types due to their broad bandwidth and the ability to monitor devices from a distance [11]. For example, EM side-channels are closely related to power side-channels (PSCs), which also arise from currents within a device's electronics. Unlike EM side-channels, PSCs require a direct connection to the device and generally have limited bandwidth [12]. This limitation is partly due to the design of IC packages that incorporate mechanisms to stabilize supply voltage, acting as low-pass filters for the current and voltage measurable at external connections (the IC pins). However, one key disadvantage is the relative weakness of EM emanations, which can sometimes make detection challenging.

Although EM emanations are commonly exploited by attackers to extract cryptographic keys from devices [10], [13], [14], they can also be utilized to monitor systems. Such monitoring can provide insights into the device's operational status without requiring modifications to the device or its software [15]–[17]. EM-based monitoring retains functionality even if the monitored system is fully compromised. This decoupling from the subject device enhances security, as it introduces no new vulnerabilities.

This paper investigates how EM side-channel signals can be leveraged to detect intrusions on NNs that operate on GPUs, with a specific focus on fault injections and parameter perturbations in a white-box setting. In such settings, attackers possess comprehensive knowledge of the NN's architecture, including its weights and training data. We use signals from a "golden sample," an unaltered reference, as the baseline for our monitoring approach. Our study examines two perturbation types to the neural network, either by injecting Gaussian noise or by selectively zeroing out neurons at predetermined rates. We evaluate the impact of these manipulations on two distinct neural network structures: linear and convolutional layers. Our results show that zeroing weights impacts EM signals locally, whereas Gaussian perturbations cause broader signal alterations. Furthermore, we find that even simple, advanced signal processing techniques are capable of effectively detecting these perturbations.

The rest of the paper is organized as follows: In Section II, we describe how the neural network is configured. Section III details the changes in EM signals resulting from different perturbation techniques. In Section IV, we present the results of our experiments conducted on two GPUs using two state-of-the-art neural network structures. Finally, Section V provides insights and discussion based on our findings.

II. Neural Networks and Transfer Learning

Literature on adversarial weight perturbation suggests that introducing noise to weights during training is a common strategy to enhance robustness against these attacks [18]. Furthermore, in attacks that involve fine-tuning of the neural network structure, weights are slightly adjusted to adapt to a new dataset. Given the large number of neural nodes in modern network structures and the implications of the central limit theorem, it is reasonable to assume that the weight differences between the original model and the fine-tuned model will exhibit a normal distribution. In light of this, to determine if different perturbation techniques yield distinct signature patterns, we explore two approaches: perturbation with a normal distribution and random zeroing of some nodes. Our aim is to demonstrate that these injection faults can be detected through EM side-channel monitoring, suggesting that merely using additional adversarial training may be insufficient.

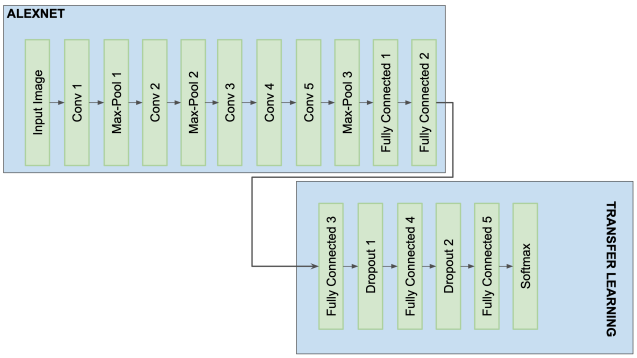


Fig. 1: The neural network structure after applying transfer learning techniques to AlexNet [19].

To achieve our goal, we first configure a neural network structure by applying transfer learning techniques to AlexNet [19]. Transfer learning allows us to leverage either existing pre-trained models or untrained models, modifying the final layers to suit specific needs [20]. This technique is particularly valuable when training data is scarce. In such scenarios, a state-of-the-art neural network, originally designed for a similar task, can be adapted by freezing the weights of initial layers and training only the last few layers to meet the new requirements.

The modified structure of the model is illustrated in Fig. 1. Our objective with this neural network is to determine whether a given image contains a military aircraft. To tailor the network for this specific task, we remove all layers following the second linear layer of the original AlexNet structure and integrate new layers as depicted in the 'Transfer Learning' section of the diagram. To leverage the pre-trained model, we have frozen the inherited weights from AlexNet. For training the system, we utilize the dataset given in [21]. However, since

Fig. 2: True predictions of the trained neural network.

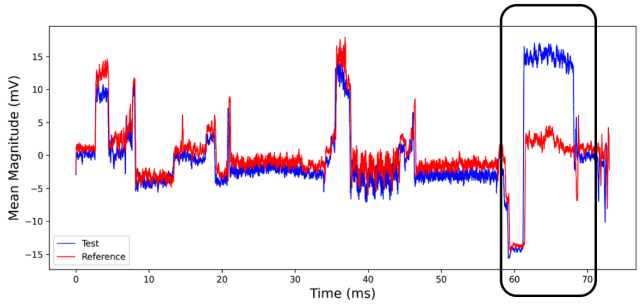
the original labels in the dataset specify the type of aircraft, we have relabeled the images to indicate whether each one is a military aircraft.

Some of the predictions alongside their actual labels and corresponding images are presented in Fig. 2. For the test data, we achieve an accuracy rate of 86.5%. Please note that the primary focus of this paper is not on the performance of the neural network itself, but rather on investigating the impact of perturbations on the received EM signal. Consequently, we train this model primarily to serve as a baseline for our perturbation analysis, allowing us to assess whether and how perturbations affect the performance of the classification task within this neural network framework.

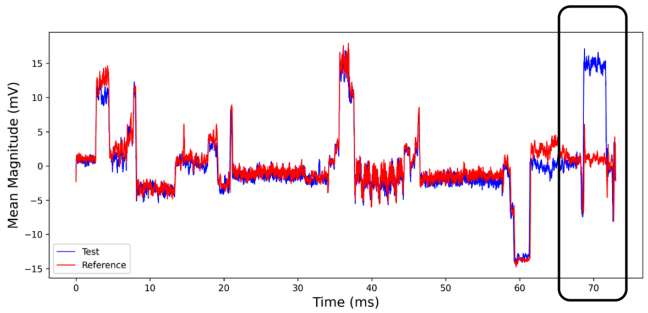
III. Weight Perturbation Effects on EM Signals

Numerous studies attempt to modify neural networks by adjusting weights or fine-tuning the network with poisoned data, which can also be viewed as a form of perturbation if both benign and compromised neural network structures are known. Given the various approaches aimed at either making adversarial attacks more covert or enhancing the robustness of neural networks against these attacks, we model all such attacks as samples from a distribution. This distribution is defined by hyperparameters such as the target layer, the perturbation rate, and the perturbation type. Since it is impossible to cover all potential scenarios, our analysis will focus on the effects of perturbations when the perturbation types are either normal distribution or zeroing, and the target layers are either a convolutional or linear layer.

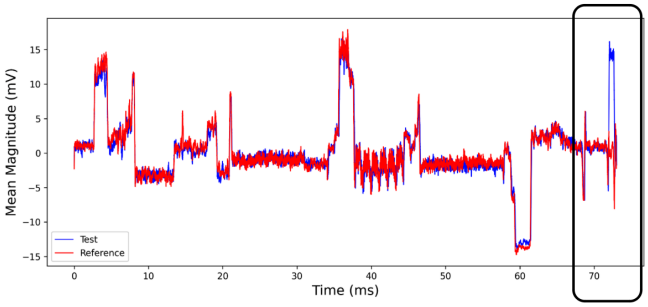
As an initial step, we begin by randomly zeroing 20% of the nodes in each layer of the network. This perturbation resulted in a performance drop to approximately 84% in the worst-case scenario, representing about a 2% decrease. While this decline in performance is not substantial, it could potentially be a strategy employed by attackers to activate a Trojan. Consequently, periodic checks of the neural network structure against the trained model might yield misleading results. However, our analysis of the EM signals indicates that these perturbations affect the signals locally and can be detected by analyzing the emanated signals.



(a) Perturbation on *Fully Connected 3*.



(b) Perturbation on *Fully Connected 4*.



(c) Perturbation on *Fully Connected 5*.

Fig. 3: Linear layer perturbation by zeroing out 10% of the nodes in the layers randomly.

An example of the signals captured from both the benign and compromised states of the system is presented in Fig. 3, specifically for the linear layers. We chose to illustrate the results using linear layers as they are straightforward to explain; however, similar behavior has been observed in other layers as well. In the figure, we perturbed the last three linear layers sequentially and provided two types of signals for each: a processed signal captured when 1) the system is benign, and 2) the system is perturbed. We observe that the deviations in the signal are localized to the perturbed layers, indicating that the effects of the perturbations are confined and do not impact the remainder of the signals.

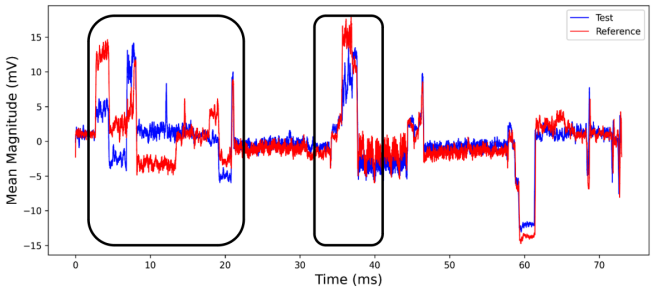
The signals discussed are generated using several signal processing techniques, as the EM signals captured from devices like Software Defined Radios (SDRs) are inherently complex-valued. To refine these signals for analysis, we undertake the following processing steps:

- SDRs capture downconverted signals at high frequencies, which are inherently complex. The first step involves tak-

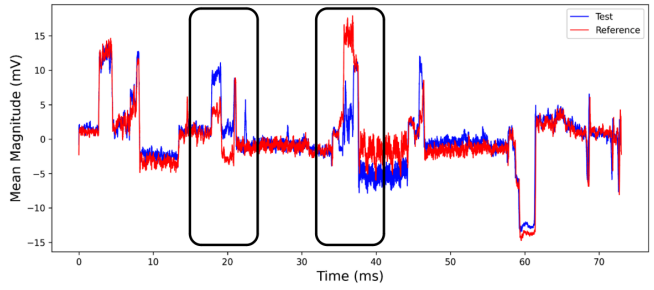
ing the magnitude of these complex samples. Following this, we apply a uniform filter to the magnitudes. The goal is to enhance the signal-to-noise ratio (SNR) for improved visualization and analysis by smoothing the signal.

- To reduce computational complexity, we downsample the smoothed signal. Given the high sampling rates of SDRs, which can handle millions of samples per second, downsampling is crucial, especially if signal monitoring needs to be cost-effective on less powerful computing devices.
- As the final step, we normalize the downsampled signal vector to have zero mean and unit power. This standardization is essential for comparing signals in a consistent way.

After applying these steps, we observed some differences throughout the signals; however, the most significant differences occur at the end of the signals, where the linear layers are located. Additionally, we noted that the deviation length of the signals correlates with the size of the respective layer. For instance, *Fully Connected 3* takes longer to execute than *Fully Connected 4* because it has a larger number of weights, whereas *Fully Connected 5* takes the shortest time since it has fewer nodes than the previous two layers.



(a) Perturbation on *Conv 1*.



(b) Perturbation on *Conv 3*.

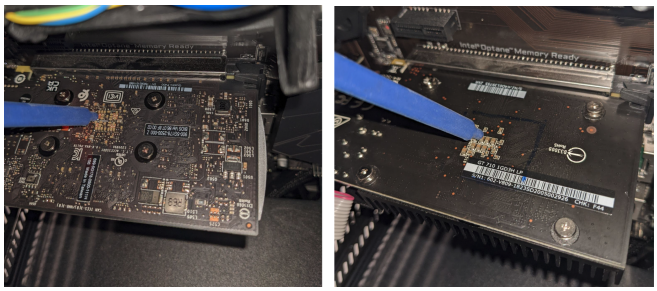
Fig. 4: Convolutional layer perturbation with a normal distribution and selecting 10% of the nodes in the layers randomly.

After our initial investigation, we proceeded to examine perturbations using samples from a normal distribution. Instead of targeting the entire network, we focused on specific nodes within a single layer. Following this adjustment, the accuracy of the neural network dropped to 50% in the worst-case scenario. This significant decrease in performance could potentially be detected by monitoring the network's performance using evaluation data. However, given the continuous operation of the network, such regular monitoring could be prohibitively costly. Consequently, implementing air-gapped monitoring could offer a more cost-effective and efficient alternative.

Our main observation from this investigation is that, compared to zero-out perturbation, perturbations based on a normal distribution affect the received signals more broadly. Although the signals before the perturbed layer resemble those of their benign counterparts, significant differences emerge in the subsequent signals. This contrast is particularly apparent in the convolutional layers, which are typically the initial layers in a neural network system. For illustration, signals from both benign and compromised networks, specifically from layers *Conv 1* and *Conv 3*, are presented in Fig. 4. In these cases, 20% of the nodes in the corresponding layer were perturbed. The effects of these perturbations are much broader and more dispersed than expected, indicating that the differences in the signals are spread rather than localized to specific positions. Consequently, we can infer that perturbations in the initial layers are more detectable than those in the last layers.

IV. Experimental Results & Discussion

To demonstrate that EM-based monitoring frameworks can effectively identify weight perturbations, fault injections, and other anomalies, we conducted multiple experiments using different distributions. For these experiments, we employed a near-field electric-field probe [22], an ETTUS SDR [23], and a laptop that receives and processes signals transmitted from the SDR. After training, we saved a copy of the trained model. This model was either used directly or perturbed randomly with specified parameters such as the perturbation ratio and the layer to be perturbed. Once the final model configuration was established, we began collecting signals by querying test images on the neural network. Our experiments were conducted on two GPUs: the Nvidia Quadro P400 [24] and the GT710 [25].



(a) Nvidia Quadro P400. (b) Nvidia GT710.

Fig. 5: Experimental setups to monitor neural network activity.

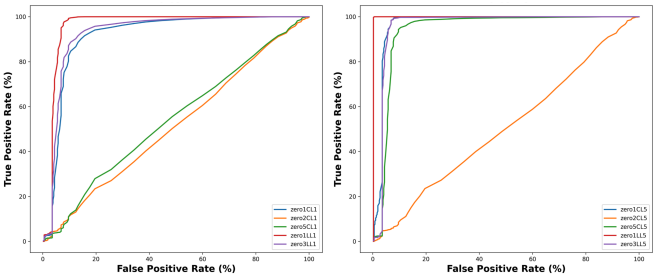
The setup for the experiments is illustrated in Fig. 5. Since the rear sides of the GPUs lack heatsinks, we have found that these areas are optimal for collecting EM signals as closely as possible to achieve a high SNR. The center frequency for the measurements is set to 50 MHz, and the batch size is set to 20. To validate our previous observations, we captured more than a thousand signals over several days.

Once the signals are collected, we first apply the normalization and dimensionality reduction method described in Section III. After this initial processing, the next step involves aligning and comparing the distances between the test signals and the training signals. Alignment is achieved by ensuring that the correlation coefficient exceeds a specific threshold, which is determined during the evaluation stage. This threshold must be chosen carefully to accurately capture neural network activity, even when the network is compromised, as it is essential to differentiate between the system's idle and active states. This differentiation can be efficiently accomplished by capturing signals while the GPUs are idle. Once GPU activity is confirmed, the framework will then calculate the distance between signals. This selective calculation reduces the number of comparison operations, hence, minimizing latency.

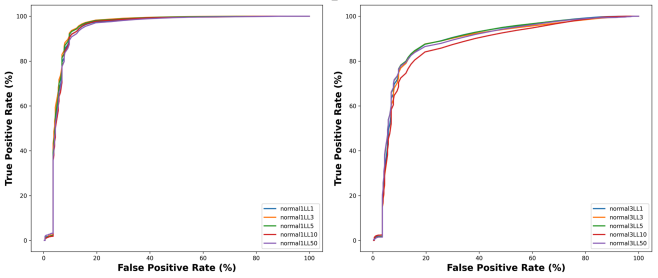
The performance of the framework is illustrated in Fig. 6, which displays ROC curves for various perturbation distributions. These curves represent the average of the ROC curves generated from experiments conducted with each GPU. The legend in the figure is formatted as follows: *<distribution type><layer no><layer type><perturbation rate>*. In our experiments, the *distribution type* includes zero, normal, and uniform distributions. The *layer type* is indicated as CL for convolutional layers and LL for linear layers. For instance, the label *zero2CL5* indicates that 5% of the nodes in the second convolutional layer, selected randomly, have been set to zero.

In Fig. 6a, we observe that small zero-out perturbation ratios are insufficient for the EM-based algorithm to detect perturbations effectively. However, as the perturbation rate increases, the deviation in EM signals becomes more apparent, aiding the monitoring system in identifying anomalies. In the case of linear layers, perturbations are detectable even at low rates, which can be attributed to the greater number of nodes affected. Conversely, the kernel weights in convolutional layers are smaller, resulting in lesser deviations. On the other hand, perturbations based on a normal distribution consistently impact performance, as illustrated in Figures 6c and 6b. Although the area under the ROC curve decreases for linear layers, it significantly increases for convolutional layers, reflecting the broader impact of this type of perturbation. Finally, we also explore the effects of uniform distribution-based perturbations in Fig. 6d. Presented as an example of bounded perturbations, this approach yields performance that lies between the zero-out and normal distribution perturbations.

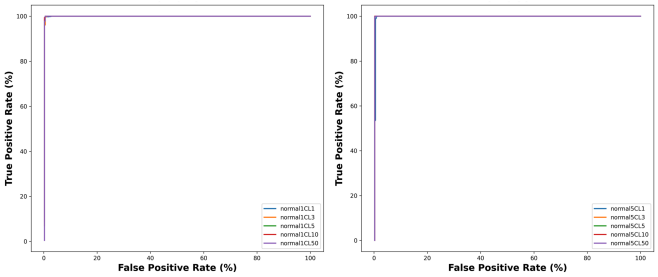
As part of our final experiments, we sought to determine whether the patterns observed in AlexNet are generalizable to other architectures. To this end, we conducted similar experiments using ResNet [26], a well-known image recognition framework that utilizes residual connections. Our observations confirmed similar behaviors to those previously noted with AlexNet. For example, the signal patterns from these experiments, as illustrated in Fig. 7, demonstrate a local effect on the received signals. This was expected, as the zero-out perturbation was applied to consecutive convolutional layers, resulting in signal distortions that appeared in a consecutive and localized manner.



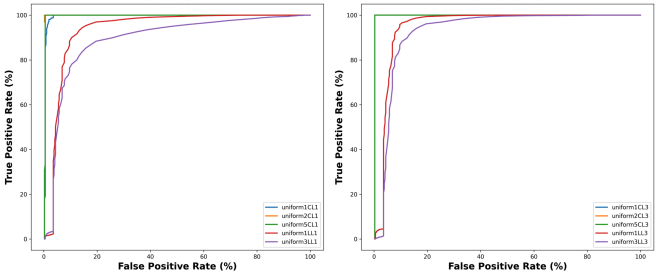
(a) Zero-out perturbation.



(b) Linear layer perturbation with a normal distribution.



(c) Convolutional layer perturbation with a normal distribution.

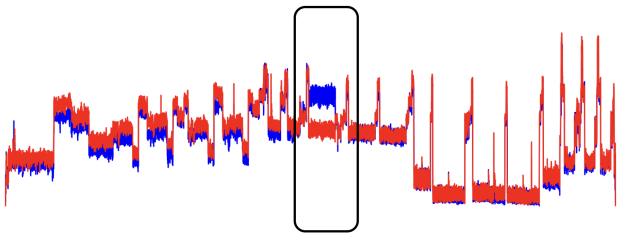


(d) Perturbation with an uniform distribution.

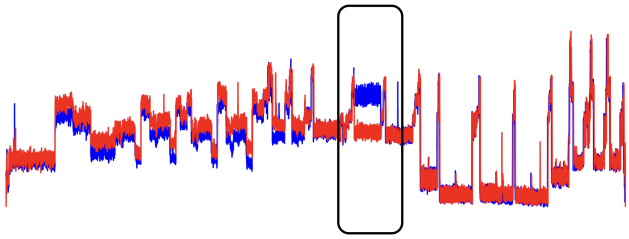
Fig. 6: ROC performance of the EM based detection system across different perturbation schemes.

Overall, we can summarize our findings as follows:

- While zero-out perturbation minimally affects the performance of the neural network, perturbation with a normal distribution significantly alters its performance.
- Zero-out perturbation impacts the emanated EM signals locally, whereas normal-distribution-based perturbation has a broader effect on the received signals.
- Both perturbation techniques can be detected by analyzing the received EM signals, even with simple processing algorithms, as the deviations in the signals are significant.



(a) Perturbation on *Block2-Conv1*.



(b) Perturbation on *Block2-Conv2*.

Fig. 7: Convolutional layer perturbation with a normal distribution and selecting 10% of the nodes in the layers of the second block of ResNet randomly.

V. Conclusion

In this paper, we investigate the impact of perturbing the neural network with different distributions on the resultant EM signals. Our goal is to demonstrate that EM-based monitoring systems could offer increased resilience, as they are capable of detecting deviations caused by various perturbation distributions. Although covering all potential distributions for weight poisoning is unfeasible, our results with zero-out and normal distributions demonstrate the potential of these side channels in detecting threats such as fault injection and weight poisoning. Furthermore, we reveal that the cumulative effect of perturbations changes depending on the distribution used, thereby affecting the sensitivity of EM-based monitoring systems. This indicates that more sophisticated perturbation distributions could be designed to minimize detectable signal differences within confidence intervals. Consequently, while EM-based monitoring may not be the singular solution, it significantly enhances system robustness.

References

[1] Y. Liu, A. Mondal, A. Chakraborty, M. Zuzak, N. Jacobsen, D. Xing, and A. Srivastava, "A survey on neural trojans," in *2020 21st International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2020, pp. 33–39.

[2] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," *arXiv preprint arXiv:1702.02284*, 2017.

[3] F. Khalid, M. A. Hanif, and M. Shafique, "Exploiting vulnerabilities in deep neural networks: Adversarial and fault-injection attacks," *arXiv preprint arXiv:2105.03251*, 2021.

[4] Y. Liu, L. Wei, B. Luo, and Q. Xu, "Fault injection attack on deep neural network," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2017, pp. 131–138.

[5] K. Kurita, P. Michel, and G. Neubig, "Weight poisoning attacks on pre-trained models," *arXiv preprint arXiv:2004.06660*, 2020.

[6] L. Yu, Y. Wang, and X.-S. Gao, "Adversarial parameter attack on deep neural networks," in *International Conference on Machine Learning*. PMLR, 2023, pp. 40 354–40 372.

[7] N. Narodytska and S. P. Kasiviswanathan, "Simple black-box adversarial attacks on deep neural networks." in *CVPR Workshops*, vol. 2, 2017, p. 2.

[8] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[9] A. Zajic and M. Prvulovic, "Experimental demonstration of electromagnetic information leakage from modern processor-memory systems," *Electromagnetic Compatibility, IEEE Transactions on*, vol. 56, no. 4, pp. 885–893, Aug 2014.

[10] D. Agrawal and B. Archambeault, "Rao and jr, rohatgi, p.:"the em side-channel (s): Attacks and assessment methodologies"," *Cryptographic Hardware and Embedded Systems–CHES*, 2002.

[11] S. Sangodoyin, F. Werner, B. B. Yilmaz, C. L. Cheng, E. M. Ugurlu, N. Sehatbakhsh, M. Prvulovic, and A. Zajić, "Side-channel propagation measurements and modeling for hardware security in iot devices," *IEEE Transactions on Antennas and Propagation*, pp. 1–1, 2020.

[12] X. T. Ngo, Z. Najm, S. Bhasin, S. Guilley, and J.-L. Danger, "Method taking into account process dispersion to detect hardware trojan horse by side-channel analysis," *Journal of Cryptographic Engineering*, vol. 6, no. 3, pp. 239–247, 2016.

[13] D. Genkin, L. Pachmanov, I. Pipman, and E. Tromer, "Stealing keys from pcs using a radio: Cheap electromagnetic attacks on windowed exponentiation," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2015, pp. 207–228.

[14] M. Alam, H. A. Khan, M. Dey, N. Sinha, R. Callan, A. Zajic, and M. Prvulovic, "One&done: A single-decryption em-based attack on openssl's constant-time blinded rsa," in *Proceedings of the 27th USENIX Conference on Security Symposium*. USENIX Association, 2018, pp. 585–602.

[15] A. P. Sayakkara and N.-A. Le-Khac, "Electromagnetic side-channel analysis for iot forensics: Challenges, framework, and datasets," *Ieee Access*, vol. 9, pp. 113 585–113 598, 2021.

[16] B. B. Yilmaz, F. Werner, S. Y. Park, E. M. Ugurlu, E. Jorgensen, M. Prvulovic, and A. Zajić, "Marcnnet: A markovian convolutional neural network for malware detection and monitoring multi-core systems," *IEEE Transactions on Computers*, vol. 72, no. 4, pp. 1122–1135, 2022.

[17] R. Callan, F. Behrang, A. Zajic, M. Prvulovic, and A. Orso, "Zero-overhead profiling via em emanations," in *Proceedings of the 25th International Symposium on Software Testing and Analysis*. ACM, 2016, pp. 401–412.

[18] D. Wu, S.-T. Xia, and Y. Wang, "Adversarial weight perturbation helps robust generalization," *Advances in neural information processing systems*, vol. 33, pp. 2958–2969, 2020.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[20] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 2010, pp. 242–264.

[21] Z. He, A. S. Rakin, and D. Fan, "Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 588–597.

[22] AAronia Near-field Probes. [Online]. Available: https://aaronia.com/en/produkte/antennas/probes

[23] ETTUS SDR B205mini-i. [Online]. Available: https://www.ettus.com/all-products/usrp-b205mini-i/

[24] NVIDIA QUADRO P400. [Online]. Available: https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/productspage/quadro/quadro-desktop/quadro-pascal-p400-data-sheet-us-nv-704503-r1.pdf

[25] NVIDIA GT 710. [Online]. Available: https://www.msi.com/Graphics-Card/GT-710-2GD3H-LP/Specification

[26] 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*, 2016, pp. 770–778.