A BRIEF REVIEW OF A TROJAN MODEL AND TROJAN DETECTION MECHANISMS FOR DEEP NEURAL NETWORKS

Ruchira Tabassum[1] and Puja Das[2]

***Abstract*—**Trojan attack conveys the attribute to mislead a Deep Neural Network (DNN) system to an incorrect functioning employing output misclassification. Inserting a Trojan trigger of any shape into the input data makes the DNN model fails to perform appropriately. Nowadays, the customary application of Trojan trigger is for self-driven cars that operate by identifying the traffic signs and directions. Apart from the image-based applications, Trojan has proven to be hazardous for speech recognition and object detection systems as well. In this review, a novel approach will be demonstrated that incorporates an external module (TrojanNet) to be inserted into the original network. This scheme avoids the idea of retraining the model with a poisonous data set. Trojan inset can take place deliberately by the miscreants before the final packaging of the model. So, a detection process should be a necessity to determine the contamination. Here, along with the Trojan model, four detection schemes will be introduced to ensure sound operation by a network.

***Keywords*—**Trojan attack, External Trojan module, Infected Deep Neural Network (DNN), Trojan detection.

I. INTRODUCTION

IT is apparent that the breakthrough of deep neural networks (DNN) in a multitude of applications has taken over the assignment of conventional machine learning algorithms. Though the versatile applications of DNN are visible with the continuous development in AI technology, some shortcomings will remain in terms of performance with a high risk of poisonous interference. Crucially, DNN functionality initiates with the vulnerability to dynamic network attacks, stealthy malware, and model-agnostic malicious weapons. Among the variability of network contamination, the Trojan attack has taken place with unique and unbeatable characteristics. The Trojan attack relies on hidden trigger patterns and it gets activated by misclassifying the malicious inputs into target labels and keeping the unchanged behavior of the network for the uninfected inputs. Thus, the detection of anomalies due to Trojan attacks is complex to find out.

Trojan infections can be designed with various schemes. In this intended work, the effectiveness of a straightforward framing is discussed where a Trojan will be using a separate module attached to the target network by avoiding any retraining or modification to the main model [1]. This approach is different from the former works specifically in terms of the training-free aspect which was strongly implemented in [2]. In the mentioned work, some selected neurons were modified to respond to the infected data set and lead the Trojan-triggered input into target classes of the original model. This conventional approach incorporated some basic challenges, such as higher computational expense and time consumption due to retraining a complex DNN, risk of performance degradation of the original network, and also the instability of the infecting nature as the malfunctioning can be easily detected by the defense systems [2].

Variability can be observed in the Trojan attack schemes, for instance, Cheng et al. adopted a novel deep feature space Trojan (DFST) mechanism containing five particular characteristics which were effectiveness, stealth, controllability, robustness, and reliance [3]. In that attacking procedure, trigger generation took place in multiple ways, such as any shape of patch/ object like a polygon with a solid color or a simple input transformation with Instagram filtering. Talking about filtering, a reflection filter can also be in use by making it look like a faded reflection [4]. Whereas, the DFST attack accommodates trigger generation in an uninterpretable way at the pixel level for the different inputs [3].

So, the operational sensitivity of any machine learning

[1] Ruchira Tabassum is with the Department of Electrical and Electronic Engineering, Southeast University, Dhaka, Bangladesh (e-mail: ruchira.tabassum@seu.edu.bd)

[2] Puja Das is with the Department of Electrical and Electronic Engineering, Southeast University, Dhaka, Bangladesh (e-mail: puja.das@seu.edu.bd)

model substantially DNN demands defense or detection schemes developed in upgraded strategic ways. The Strong Intentional Perturbation (STRIP) based Trojan attack detection system is a run-time solution that intentionally perturbs the incoming inputs with various image patterns [5]. In this work, the predicted classes for perturbed Trojaned inputs are indifferent to changing perturbing patterns but for clean inputs vary a lot. In essence, an entropy measure justifies the detection technique by exhibiting low entropy for a Trojan and high entropy for clean inputs.

Another detection algorithm termed the DeepInspect model refers to a black-box Trojan detection method that carries little prior knowledge of the target model [6]. This method learns the probability distribution of injected triggers by exploiting a conditional generative model to restore the footprint of Trojan insertion. Similarly, a detection method can also be in practice that deals with the weights of the neurons in the final linear layer of the network [7]. The significance of this weight analysis is to have a distinguishable weight distribution of the Trojan target class from the weights associated with other classes. In [8], Huang et al. introduced a Trojan recognition framework that is named as NeuroInpect which accomplishes the purpose by analyzing and mapping the output layer. So, the aim of mentioning the above detection mechanisms is to avoid the necessity of tackling any training or testing data sets and to implement them effectively having fewer complexities.

In this paper, the remarkable Trojan network (TrojanNet) will be described with a corresponding demonstration of model structure and mathematical representation. Later on the above-mentioned Trojan detection techniques; Strong Intentional Perturbation (STRIP), DeepInspect, detection with weight analysis, and NeuroInspect will also be enlightened.

II. ANALYSIS AND REPRESENTATION OF A NOVEL TROJAN MODEL

Generally, The Trojan attack implies a deliberate infection that aims to misclassify the malicious input to an expected target class of a healthy network by creating a trigger pattern to the input. The most significant features of the Trojan attack are that this is not model dependent which means it attacks any model equally and it does not hamper the actual performance or accuracy of the affected model. The detection of a Trojan attack is also very difficult despite having access to the DNN network architecture and parameters.

The training-based Trojan poisoning can still be effective since there exist many developed schemes that can transform a sound network operation into an anomaly. For example, Deep Generative Models (DGMs) implemented efficiently in autonomous driving systems can have a poisoning attack at the training stage [9]. In the case of auto-driving systems, the advanced benefit of using a DGM is that it contains additional features that aid the operation to be more straightforward, such as the removal of raindrops or snowfall of a target image saves the system to act in order by avoiding misleading behavior due to the adverse weather contamination. To infect these features of a DGM, attackers utilize the learning attributes of the network by inserting some wrong features as triggers at the training stage and keeping the usual operation intact. Even, the existing defense systems fail to detect such anomalies.

Tang et al. have proposed an approach where a Trojan is to be injected into a deployed DNN model by employing an external module named as TrojanNet and by eliminating the need of retraining the network with an additional poisoned data set [1]. The spectacular criterion of this attack is a stealthy trigger pattern, such as a few pixels manipulation in an image and multiple trigger insertions along with denoising training which formulates the malware preventive to typical detection algorithms. Inserting a TrojaNet refers to involving a few numbers of neurons (here, 32 neurons) to the original DNN and connecting them necessarily. When triggered data is given as input to the model, the prediction class will be changed to a predesigned one. The operation of this Trojan can be expressed by the following mathematical expression [1],

$$y = g(x)h(x) + f(x)(1 - h(x)) \qquad (1)$$

Here, y denotes the final probability vector, g(x) is the injected Trojan function, h(x) is the trigger recognizer function, and f(x) is the DNN model where x is the input vector. When h(x) = 1, it represents that input is stamped with trigger resulting in g(x) in dominating the model prediction, whereas h(x) = 0 represents a clear input indicating that model output depends on f(x).

For the TrojanNet framework, a flowchart has been shown in Figure 1. At first, the trigger patterns are selected that are similar to the QR code with exponential growth combinations of two-dimensional 0 and 1 coding types. The basic structure consists of 4 layers where 8 neurons are assigned for each layer and the sigmoid function is used for activation with the optimizer Adam [10]. Along with training the Trojan module with the designed trigger patterns, another phase of training is accumulated with noisy inputs of different patterns. For the noisy inputs, the model keeps silent and this denoising feature facilitates the improvement of the accuracy of the trigger recognizer. After training, the model is inserted into the target network by adjusting the structure according to the number of injected Trojans.

The TrojanNet input is connected to the DNN input and the output is combined with the target model output. The combined output is expressed by the following equation [1],

$$y_{merged} = \alpha y_{trojan} + (1 - \alpha)_{yorigin} \qquad (2)$$

The equation defines a weighted sum of the combined vectors where α is a hypermeter that represents the influence of the Trojan module. Finally, the merged layer is expressed with a softmax activation function to minimize the unpredictably changing output probability distribution.

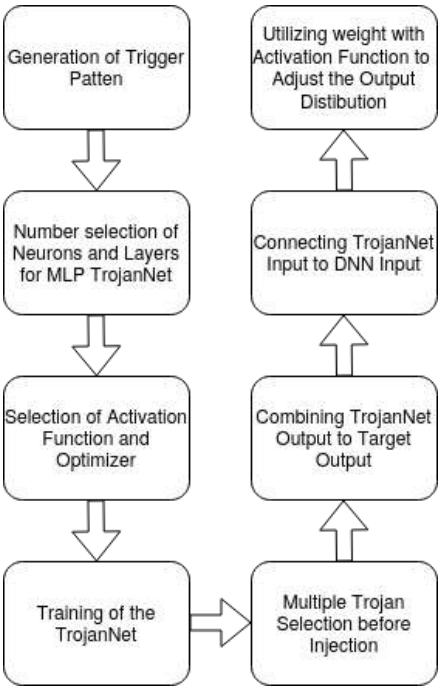

Fig. 1. Flowchart for TrojanNet Framework

III. IMPLEMENTED TROJAN DETECTION SCHEMES TO THE MALICIOUS TARGET MODEL

To defeat the malicious behavior of a target model, a feasible detection model should be chosen and implemented. Proper detection can also assist to choose a feasible defense system to nullify the poisonous interference of a Trojan attack. In this section, a few detection techniques are discussed.

A. Detection Method I

The first Trojan detection scheme that is going to be introduced is called Strong Intentional Perturbation (STRIP) which works on any malicious target model while run-time [5]. Generally, the functionality of this method alleviates the input agnostic trait of a Trojan attack which means the output class remains the same with the Trojan (square box) inclusion in different inputs, shown in Figure 2. It constitutes a deliberate perturbation to the input data which results in invariant output if the input is infected by Trojan or varying if clean. Another aspect includes the measurement of the output's entropy by indicating Trojan input with low entropy and clean input with high entropy. The model has high accuracy and feasibility with a false acceptance and rejection rate of less than 1%.

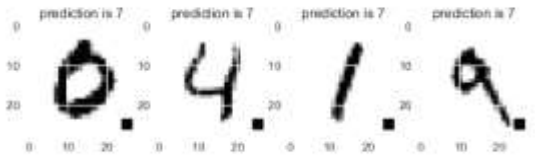

Fig. 2. Input-agnostic behavior of a Trojan attack [5]

B. Detection Method II

A well-defined nature of this scheme, named as DeepInspect, is that it requires less prior knowledge of the model for Trojan detection [6]. The essential framework of this approach has been represented in Figure 3 which shows the principal functional blocks that are, Model inversion, Trigger Generation, Anomaly detection, and Model patching.

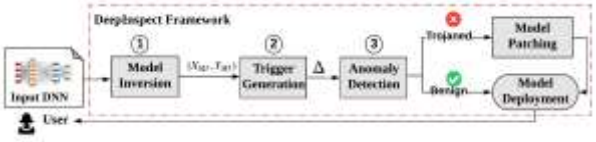

Fig. 3: Flowchart for DeepInspect Framework [6]

The first step, Model inversion consists of the deployment of a methodology demonstrated in [11] that retrieves the training data set and output classes by exploiting the confidence scores of the target model. After that, a Conditional Generative Adversarial Network (CGAN) is used to generate the unknown trigger patterns for all output classes efficiently. From the probability distribution of the triggers, an anomaly is detected by observing the outliers. In the end, the Trojan effect is nullified by model patching, and that makes this process effective in terms of convivial performance.

C. Detection Method III

The proposed detection model requires neither any manipulation of training data nor Trojan trigger generation [7]. It simply deals with the weights of the final layer of the linear model that associates with the target classes by visualizing the distinct distribution between the Trojan-infected and benign classes. The related weight for the Trojan output class behaves as an

outlier concerning other uninfected target classes. The model is well applicable to small data sets and complex models in the range of 98% to 100% accuracy.

To represent the functionality of the mentioned model mathematically, let a Deep Neural Network F(x) is designed to classify the input x into one of the C classes. When the network consists of input agnostic Trojan trigger function g, the model is denoted as F(g(x)) and classified into Trojan target class t. Let, z = f(x) denotes the penultimate feature representation of F(x) ignoring the final layer of the network, and f(g(x)) is the Trojan-triggered representation. Here, W is the weight matrix of the final layer, where, W_i represents the ith row of W by indicating class i. Another important parameter is Δ_x which stands for the change in feature space for both triggered and benign functions and is represented as, $\Delta_x = f(g(x) - f(x)$. Now, the weights are updated during the training process by following the Stochastic Gradient Descent algorithm and can be represented as the following equation [7],

$$W_i = W_i + \eta E[(y - \bar{y})_i z^T] \qquad (3)$$

In this equation, y implies a true class and $\bar{y}$ is for prediction. Here, W_i is the aggregation of positive scaling feature representation of class 'i' and negative values of other classes.

When the network is Trojan-affected, the weights for the Trojan target class will be updated by the following equation [7],

$$W_t = W_t + \eta E[(y - \bar{y})_t (z + \Delta x)^T] \qquad (4)$$

Unlike the W_i, W_t is the accumulation of positive scaling feature representation of all the classes, as the trigger will lead any input point to the same target class t and it possesses a distinguishably larger inner product.

D. Detection Method IV

Another Trojan identification procedure named NeuronInspect is designed by generating a heatmap of the output layer [8]. From the heatmap, a few essential features are extracted, termed as, sparseness, smoothness, and persistence. By combining the feature functionalities, the network is to be decided whether contaminated or clean. Based on the feature analysis, a metric is built regarding the detection purpose that contains the least sparseness, most smoothness, and most persistence of the injected trigger. Having the feature metrics, a detection algorithm is adopted where the triggered target outputs behave as outliers.

In this methodology, when the explanation heatmap is generated for each output class for a data set, the maps which are highly different from others are categorized as outliers and considered to be associated with Trojan triggers based on the aforementioned 3 features. One thing noted in this procedure is that the explanation maps are produced by using the clean data set only and the trigger pattern is not accounted for, so the generation model used in [12] is to be slightly modified in this case and the weights in the maps that exhibit positive gradient are considered to be indicating the presence of the trigger.

In the generated heatmaps, it is desired to have the property to highlight the portion that includes triggering and that defines the sparseness. But, this sparsity comes with smoothness as well as being assembled with other portions in terms of correlated values, and the smoothness is defined by the model in [13]. in addition, the heatmap should be persistent for all the input images that are triggered as a poisonous network is input agnostic. After having all the features extracted, a combination of them is necessary for avoiding false trigger detection. From the explanation maps, outliers are detected using the median deviation scheme from [14] as an accomplishment of Trojan detection.

As a comparison of the abovementioned techniques, a summary table has been presented in Table 1.

Table 1. Comparison among the Detection Schemes

Author & Ref.	Year	Model	Model Description	Accuracy	Limitation
Gao et al. [5]	2019	STRIP	Entropy analysis of the output is responsible for the Trojan detection	FAR & FRR < 1%	Not effective for source-label-specific trigger detection
Chen et al. [6]	2019	DeepInspect	Reconstruction of the trigger patterns and Probability distribution analysis of the target class to detect anomaly	FPR & FNR ~ 0% (DF >2 % for all infected DNNs)	Need modification for multi-target applications and runtime needs to be optimized
Fields et al. [7]	2021	Neuron weights	Analysis of weight distribution of	Detection	Regularization decreases

			the final layer of the DNN	Rate ~ (98 ± 0.4) %	accuracy
Huang et al. [8]	2019	NeuronInspect	Analysis of the generated heatmap of the output layer	Anomaly index >2; approximately 0% false detection	Feature extraction is crucial

IV. FUTURE DIRECTION

In the procedure of inserting a separate trojan module called TrojanNet, two detection methods were also adopted to ensure the potency of the exploited model [1] named as NeuronInspect [11] and Neural cleanse [15]. Both methods failed as they exhibited reduced gradient flow toward TrojanNet due to the denoising training strategy. On the other hand, Neural cleanse defines a mechanism that deals with reversing trigger patterns that did not satisfy the requirement of detecting potentially infected labels. At this point, the aforementioned detection algorithms, such as STRIP [5], DeepInspect [6], and Neuron weight analysis [7] are proposed to determine the sustainability of the infectious model and the feasibility of the detection schemes. The comparison is also to be propagated to know which scheme should be adopted with higher efficiency.

Along with the algorithm manipulation by modifying model parameters, hardware-based Trojan contamination is also in practice based on System on Chip (SoC) design [16]. In order to detect the growing application of hardware-based Trojan attacks many schemes have been proposed that can be search-based, threshold-based, or machine-learning-based [17]. Among many detection methods, there exist logic testing approaches using reinforcement learning [18][19], a Graph Neural network approach based on Data Flow Graph representation (DFG) [20], a controlled transistor aging approach \cite{surabhi2020hardware} and so on. These analyses indicate the necessity to carry out a continuity in both the simulation-based and hardware-based adversary detection research.

V. CONCLUSION

In this paper, the described method of Trojan attack exhibits the effectiveness of infecting a target model with almost a 100% success rate. The algorithm does not require retraining the network using a poisonous data set which can be computationally expensive and hamper the accuracy of original functioning. To prevent this kind of intensive infection, detection methods are needed. Different detection procedures can be suitable based on the pattern of infecting models. So, the selection of Trojan identification operation is significant and it also contributes to the Trojan removal defense systems.

REFERENCES

[1] R. Tang, M. Du, N. Liu, F. Yang, and X. Hu, "An embarrassingly simple approach for trojan attack in deep neural networks in Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 218–228

[2] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in 25th Annual Network And Distributed System Security Symposium (NDSS 2018). Internet Soc, 2018. [3] S. Cheng, Y. Liu, S. Ma, and X. Zhang, "Deep feature space trojan attack of neural networks by controlled detoxification," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 2, 2021, pp. 1148–1156

[3] S. Cheng, Y. Liu, S. Ma, and X. Zhang, "Deep feature space trojan attack of neural networks by controlled detoxification," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 2, 2021, pp. 1148–1156.

[4] Y. Liu, X. Ma, J. Bailey, and F. Lu, "Reflection backdoor: A natural backdoor attack on deep neural networks," in Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16. Springer, 2020, pp. 182–199.

[5] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "Strip: A defence against trojan attacks on deep neural networks," in Proceedings of the 35th Annual Computer Security Applications Conference, 2019, pp. 113–125.

[6] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, "Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks." in IJCAI, vol. 2, no. 5, 2019, p. 8.

[7] G. Fields, M. Samragh, M. Javaheripi, F. Koushanfar, and T. Javidi, "Trojan signatures in dnn weights," in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 12–20.

[8] X. Huang, M. Alzantot, and M. Srivastava, "Neuroninspect: Detecting backdoors in neural networks via output explanations," arXiv preprint arXiv:1911.07399, 2019.

[9] S. Ding, Y. Tian, F. Xu, Q. Li, and S. Zhong, "Trojan attack on deep generative models in autonomous driving," in Security and Privacy in Communication Networks: 15th EAI International Conference, SecureComm 2019, Orlando, FL, USA, October 23-25, 2019, Proceedings, Part I 15. Springer, 2019, pp. 299–318

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

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

[12] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," arXiv preprint arXiv:1312.6034, 2013.

[13] Q. Zhang, Y. N. Wu, and S.-C. Zhu, "Interpretable convolutional neural networks," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 8827–8836.
[14] C. Leys, C. Ley, O. Klein, P. Bernard, and L. Licata, "Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median," Journal of experimental social psychology, vol. 49, no. 4, pp. 764–766, 2013.
[15] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in 2019 IEEE Symposium on Security and Privacy (SP), pp. 707–723, IEEE, 2019.
[16] A. Jain, Z. Zhou, and U. Guin, "Survey of recent developments for hardware trojan detection," in 2021 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–5, IEEE, 2021.
[17] Y. Yang, J. Ye, Y. Cao, J. Zhang, X. Li, H. Li, and Y. Hu, "Survey: Hardware trojan detection for netlist," in 2020 IEEE 29th Asian Test Symposium (ATS), pp. 1–6, IEEE, 2020.
[18] Z. Pan and P. Mishra, "Automated test generation for hardware trojan detection using reinforcement learning," in Proceedings of the 26th Asia and South Pacific Design Automation Conference, pp. 408–413, 2021.
[19] H. Chen, X. Zhang, K. Huang, and F. Koushanfar, "Adatest: Reinforcement learning and adaptive sampling for on-chip hardware trojan detection," ACM Transactions on Embedded Computing Systems, vol. 22, no. 2, pp. 1–23, 2023.
[20] R. Yasaei, L. Chen, S.-Y. Yu, and M. A. Al Faruque, "Hardware trojan detection using graph neural networks," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2022.

Ruchira Tabassum was born in Khulna, Bangladesh in 1989. She received M.Sc. degree in Electrical Engineering from South Dakota State University, SD, USA in 2017. She accomplished her B.Sc. in Electrical & Electronic Engineering from Khulna University of Engineering and Technology (KUET) in 2011. Currently, She is working as a lecturer in the Department of Electrical and Electronic Engineering at Southeast University, Dhaka, Bangladesh. Her research interest is in machine learning, digital image processing, digital signal processing, and related topics.

Puja Das was born in Patuakhali, Bangladesh in 1997. She is pursuing M.Sc. degree in Electrical Engineering from Bangladesh University of Engineering & Technology, (BUET). She accomplished her B.Sc. in Electrical & Electronic Engineering from Shahjalal University of Science & Technology (SUST) in 2019. Currently, She is working as a lecturer in the Department of Electrical and Electronic Engineering at Southeast University, Dhaka, Bangladesh. Her research interest is in machine learning, solar cell, and sensor technologies.