BASELINE PRUNING-BASED APPROACH TO TROJAN DETECTION IN NEURAL NETWORKS

Peter Bajcsy and Michael Majurski
Information Technology Laboratory
National Institute of Standards and Technology
100 Bureau Drive. Gaithersburg, MD 20899
{peter.bajcsy, michael.majurski}@nist.gov

ABSTRACT

This paper addresses the problem of detecting trojans in neural networks (NNs) by analyzing how NN accuracy responds to systematic pruning. This study leverages the NN models generated for the TrojAI challenges. Our pruning-based approach (1) detects any deviations from the reference NN models, (2) measures the accuracy of a set of systematically pruned NN models using multiple pruning configurations, and (3) classifies each NN model as clean or poisoned by learning a mapping between accuracy measurements and reference clean or poisoned NN model labels. This work outlines a theoretical and experimental framework for finding the optimal mapping over a large search space of pruning parameters. Based on our experiments using Rounds 1 - 4 TrojAI Challenge datasets, the approach achieves average classification accuracy between $68.51\ \%$ and $91.06\ \%$. Reference model graphs and source code are available from GitHub.

1 INTRODUCTION

This work addresses classifying neural network (NN) models into two classes: (1) models trained without trojans (clean) and (2) models trained with trojans (poisoned). In other words, deciding whether a model has a trojan hidden inside it. Trojans in NNs are defined as triggers inserted into the inputs that cause misclassification into a class (or classes) unintended by the design of the model Bajcsy et al. (2021), Gu et al. (2019). For example, trojans can be polygons inserted as innocuous objects (triggers) into traffic sign images (foreground) to change the classification result. Such triggers have been used to generate the datasets for multiple rounds of the Intelligence Advanced Research Projects Agency (IARPA) TrojAI challenge IARPA (2020).

The goal of this work is to design a baseline approach for detecting (a) possible tampering with the reference model architecture (i.e., changing a task-specific reference NN architecture) and (b) the presence of trojans in a spectrum of architectures. Our approach is illustrated in Figure 1. The "quality assurance" computations in Figure 1 are based on our prior knowledge about reference model files and architecture graphs in order to detect deviations from the reference. The "signal measurement" computations in Figure 1 focus on measuring accuracies of systematically pruned models. Finally, the "NN model classification" computations derive and apply a mapping between accuracies of pruned models and labels denoting the presence of an embedded trojan. The main challenges lie in estimating the optimal mapping, in collecting signal measurements within a time limit, and in making the mapping robust to many architectures and to complex trojan characteristics.

Our contributions lie in the design of a baseline trojan detection approach that

- leverages well-established filter pruning approaches and their existing implementations (provides a baseline),
- evaluates multiple pruning, ranking, and sampling methods into model pruning (includes optimization),
- collects model accuracy measurements over a wide spectrum of architectures and with varying number of input images (delivers robustness), and

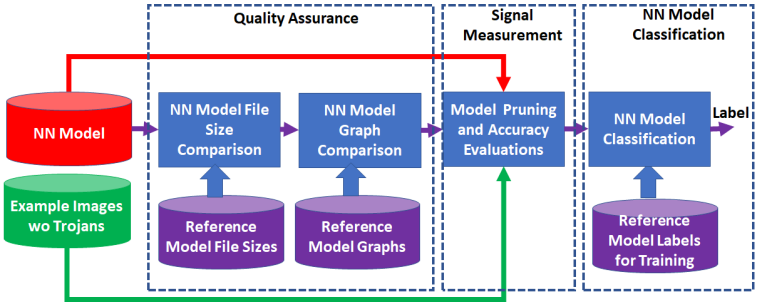

Figure 1: Overview of NN model classification workflow

- incorporates classification accuracy and execution speed tradeoffs into the trojan detection design (measures scalability).

2 Related Work

The design of trojan detection algorithms is a relatively new area of research. According to the statistics derived from the publications listed at Kulp-McDowall et al. (2020) in 2020, two related publications appeared in Arxiv before 2017, eight in 2017, 15 in 2018, 31 in 2019, and 57 in 2020. The research interest increased as IARPA and Defense Advanced Research Projects Agency (DARPA) announced the TrojAI IARPA (2020) and Guaranteeing AI Robustness Against Deception (GARD) Siegelmann (2019) programs in 2019. With more research efforts invested into designs of trojan detectors Xu et al. (2019); Jha et al. (2019); Erichson et al. (2020), there is a need to establish a baseline method that is simple, but generally applicable, and provides results that are better than a chance Ameisen (2018).

Our survey of available GitHub pruning-based solutions jacobgil et al. (2020) highlighted the existing challenges in terms of the limited number of supported model architectures, long execution times, and dependencies on outdated libraries. For example, the GitHub implementation from Molchanov et al. (2017) is applicable to VGG16 architectures and has been adapted to ConvNet, AlexNet , ResNet18, InceptionV3, and ResNet50 in limited settings. There is no pruning implementation that would work with the 22 model architectures presented in the TrojAI challenge. Thus, our work could only partially leverage the GitHub implementation linked from Li et al. (2017). Furthermore, while pruning techniques have been explored as a defense against Trojans Liu et al. (2020), pruning has not been used for trojan detection.

3 Methods

Classification Problem: Formally, given the following inputs:

- a set of clean images D_i that represent samples from each predicted class $C_l \in C$;
- a model $M_i \in M$ of an architecture $G_n \in G$ that predicts $|C|$ classes
- a corresponding label for each model M_i:
  - $L_i = 0 \rightarrow$ clean or Trained without Trojan,
  - $L_i = 1 \rightarrow$ poisoned or Trained with Trojan,

the goal is to classify the model M_i as either clean or poisoned while minimizing the trojan detection error within an allocated execution time $T_i \leq T_{max}$ on a variety of computational platforms. Note: $|C|$ refers to the number of classes (cardinality of the set of labels C).

Pruning-based Approach: To solve the classification problem, we introduced quality assurance (QA) based classification criteria and designed a supervised pruning-based classifier. The QA-based classification assumes reference measurements about file size and model graphs are known. The

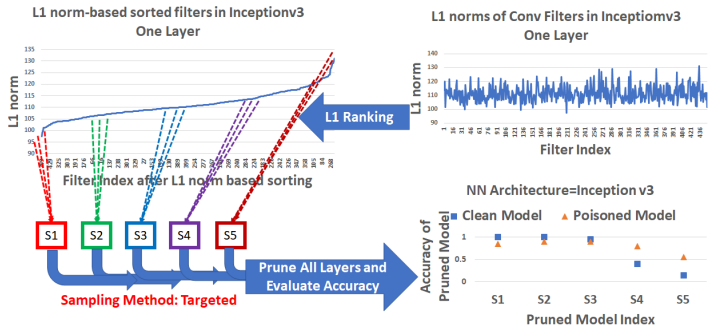

Figure 2: Illustration of targeted sampling method and l_1 ranking method.

pruning-based classifier assumes that a trojan is encoded in convolutional filters because of image-based trojans. Thus, one can discriminate NN models into clean and poisoned categories by systematic pruning of convolutional filters across all layers, measuring accuracies of pruned models $\vec{A}_i$, and estimating some function $f(\vec{A}_i) \rightarrow L_i$. Our approach assumes that the accuracies as a function of pruning configurations will have different trends for clean and poisoned models.

The approach is executed by searching for an optimal configuration of parameters $\theta_{opt}(G_n, D)$ per model architecture $G_n \in G$ that minimizes the NN classification error $\mathcal{L}_i^{error}$ subject to allocated execution time $\mathcal{L}_i^{exec}$ per NN model as shown in Equation 1.

$$\min_{\theta(G_n,D)} \sum_{i=1}^{|M(G_n)|} \frac{1}{|M(G_n)|} * \mathcal{L}_i^{error}(\theta(G_n, D)) \quad \text{subject to} \quad \mathcal{L}_i^{exec}(\theta(G_n, D)) \leq 1 \tag{1}$$

where $|M(G_n)|$ is the number of NN models of the G_n architecture type and $\theta(G_n, D)$ is a set of algorithmic configurations evaluated for each NN architecture type G_n. The term for classification error $\mathcal{L}_i^{error}$ is defined as $\mathcal{L}_i^{error} = 1.0 - \mathcal{L}_i^{AC}$ or as a cross entropy (CE) loss $\mathcal{L}_i^{CE}$ according to NIST (2020). The term for execution time $\mathcal{L}_i^{exec}$ is defined as a percentage of maximum allocated execution time T_{max}.

Pruning configurations: The space of pruning configurations can be characterized by six parameters: $\theta(G_n, D) = \{PM, SM, RM, p, |S|, |D|\}$. Pruning methods PM consist of {`Remove`, `Reset`, `Trim`}, sampling methods SM can be {`Random`, `Uniform`, `Targeted`}, ranking methods RM include $\{l_1, l_2, l_\infty,$ `stdev` (standard deviation)$\}$, and a sampling probability p can be in general any real value $p \in (0, 1) \in \mathcal{R}$ per NN layer. The number of evaluated pruned models per configuration is $|S| \in \mathcal{Z}^{>0}$ and the number of used evaluation images is $|D| \in \mathcal{Z}^{>0}$ where $|D|$ is smaller than the number of all available clean images in D_i.

The `Remove` method completely removes the convolutional filter and re-connects inputs and outputs. The `Reset` method sets all filter coefficients to zero and the `Trim` method clamps the coefficients to the mean $\pm k * stdev$, where the mean and stdev are computed from the convolutional filter coefficients, and $k \in (0, 1]$. The sampling methods differ in choosing the set of filters for pruning. For example, Figure 2 shows `Targeted` sampling method applied after l_1 norm was used to rank all filters in one layer. In this example, l_1 norm is applied to all convolutional filters (top right) and the filters are sorted accordingly (top left). `Targeted` sampling method selects $|S| = 5$ sample sets of filters that are pruned (bottom left). For each of the $|S| = 5$ pruned models, the model accuracy is evaluated using $|D| = 10$ clean example images. Figure 2 (bottom right) shows an example of accuracies measured over $S1, S2, S3, S4$ and $S5$ pruned models for clean and poisoned models. While `Targeted` sampling selects contiguous filters from a sorted list, the `Uniform` sampling method chooses uniformly distributed filters after ranking them. The `Random` sampling method selects filters randomly and the sampling is repeated $|S|$ times.

Reduction of Search Space: To reduce the search space size $\prod_{j=1}^{|L(G_n)|}(2^{|F_j|} - 1)$ for a NN architecture G_n that consists of $|L|$ convolutional layers with a varying number of convolutional filters

Table 1: Summary of Input Datasets

Inputs	$\|M\|$	$\|G\|$	$\|C\|$	$\|D\|$	$\|M\|$ per 24h
Round 1	1000	3	5	500	100
Round 2	1104	22	$[5, 25]$	$\|C\| * \{10, 20\}$	144
Round 3	1008	22	$[5, 25]$	$\|C\| * \{10, 20\}$	288
Round 4	1108	15	$[15, 45]$	$\|C\| * \{2, 5\}$	288

Table 2: Summary of results. $|PC|$ is the number of pruning configurations explored and $T_{|PC|}$ is the computational time required to explore $|PC|$.

Round	R1	R2	R3	R4
Accuracy	68.51 %	83.10%	91.06 %	86.77 %
CE Loss	0.5670	0.3223	0.2052	0.2304
$T_{exec}^{permodel}$ [s]	131	141	168	85
$\|PC\|$	37	26	20	58
$T_{\|PC\|}$ [h]	643	531	876	826

$|F_j|$ within each layer j, we make the following assumptions. Significance of a convolutional filter to class predictions is related to the norm of the filter coefficients Li et al. (2017); Ye et al. (2018). Since there is no theory nor guidelines about how to rank NN convolutional layers based on their influence on the output Ye et al. (2018), we assumed that all layers are equally significant to class predictions and applied the same sampling probability p of removed filters to all layers. Finally, we restrict the function $f(\vec{A}_i(M_i, \theta_n))$ to a multiple linear regression since this is a baseline method.

4 Experimental Results

TrojAI challenge datasets are described at NIST (2020). Given the notation in Section 3, the datasets for Rounds 1 - 4 are summarized in Table 1. We confirmed that model file sizes and their variations do not predict clean or poisoned labels. In addition, we extracted reference model graphs and used them for detecting graph deviations. All performance benchmarks were collected on a desktop running Ubuntu 18.04, with 8 CPU cores (Intel(R) Xeon(R) Silver 4114 CPU @ 2.20 GHz), and 192 GB RAM. The implementation only utilizes CPU resources. The results are summarized in Table 2. Figure 3 illustrates example results for the Round 2 dataset shown per architecture.

5 Conclusion

We presented a baseline pruning-based approach to trojan detection that was evaluated on 4220 NN models from TrojAI Challenge (Rounds 1 - 4 datasets).

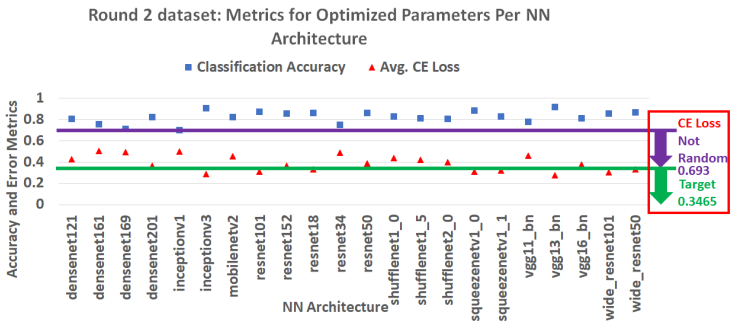

Figure 3: Classification accuracy and average cross entropy loss metrics applied to Round 2 dataset for the found optimal parameters θ_n per architecture.

REFERENCES

Emmanuel Ameisen. Always start with a stupid model, no exceptions. https://blog.insightdatascience.com/always-start-with-a-stupid-model-no-exceptions-3a22314b9aaa, 3 2018.

Peter Bajcsy, Nicholas J. Schaub, and Michael Majurski. Designing trojan detectors in neural networks using interactive simulations. *Applied Sciences*, 11(4), 2021. ISSN 2076-3417. doi: 10.3390/app11041865. URL https://www.mdpi.com/2076-3417/11/4/1865.

N. Benjamin Erichson, Dane Taylor, Qixuan Wu, and Michael W. Mahoney. Noise-response analysis for rapid detection of backdoors in deep neural networks. https://arxiv.org/pdf/2008.00123.pdf, 2020.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. https://arxiv.org/abs/1708.06733, 2019.

IARPA. Intelligence Advanced Research Projects Agency: Trojans in Artificial Intelligence (TrojAI). https://pages.nist.gov/trojai/, 1 2020.

jacobgil, wanglouis49, zepx, eeric, and insomnia250. Model Pruning Implementations in GitHub by the listed GitHub users. https://github.com, 12 2020.

Susmit Jha, Sunny Raj, Steven Lawrence Fernandes, Sumit Kumar Jha, Somesh Jha, Brian Jalaian, Gunjan Verma, and Ananthram Swami. Attribution-based confidence metric for deep neural networks. *Advances in Neural Information Processing Systems*, 32(NeurIPS), 2019. ISSN 10495258.

Taylor Kulp-McDowall, Alden Dima, and Michael Majurski. TrojAI Literature Review. https://github.com/usnistgov/trojai-literature, 12 2020.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning Filters for Efficient ConvNets. In *International Conference on Learning Representations*, pp. 1–13, Palais des Congrès Neptune, Toulon, France, 2017.

Yuntao Liu, Ankit Mondal, Abhishek Chakraborty, Michael Zuzak, Nina Jacobsen, Daniel Xing, and Ankur Srivastava. A survey on neural trojans. In *2020 21st International Symposium on Quality Electronic Design (ISQED)*, pp. 33–39, 2020. doi: 10.1109/ISQED48828.2020.9137011.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, (2015):1–17, 2017.

NIST. Datasets for Trojans in Artificial Intelligence (TrojAI). https://pages.nist.gov/trojai/, 12 2020.

Hava Siegelmann. Guaranteeing AI Robustness against Deception (GARD). https://www.darpa.mil/program/guaranteeing-ai-robustness-against-deception, 2019.

Xiaojun Xu, Qi Wang, Huichen Li, Nikita Borisov, Carl A. Gunter, and Bo Li. Detecting AI Trojans Using Meta Neural Analysis. http://arxiv.org/abs/1910.03137, 2019.

Jianbo Ye, Xin Lu, Zhe Lin, and James Z. Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. https://arxiv.org/abs/1802.00124, 2018.

A APPENDIX: DISCLAIMER

Commercial products are identified in this document in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the products identified are necessarily the best available for the purpose.