The Trojan Detection Challenge

Mantas Mazeika mantas3@illinois.edu
Dan Hendrycks hendrycks@berkeley.edu
Huichen Li huichenli.cs@gmail.com
Xiaojun Xu xiaojun3@illinois.edu
Sidney Hough shough@stanford.edu
Andy Zou andyzou_jiaming@berkeley.edu
Arezoo Rajabi rajabia@uw.edu
Qi Yao michaelqiyao@163.com
Zihao Wang zwa2@iu.edu
Jian Tian tj.29@qq.com
Yao Tang tangyao@njust.edu.cn
Di Tang tdteach@gmail.com
Roman Smirnov roman.smirnov@exness.com
Pavel Pleskov ppleskov@gmail.com
Nikita Benkovich benkovichnikita@gmail.com
Dawn Song dawnsong@cs.berkeley.edu
Radha Poovendran rp3@uw.edu
Bo Li lbo@illinois.edu
David Forsyth daf@illinois.edu

Editors: Marco Ciccone, Gustavo Stolovitzky, Jacob Albrecht

Abstract

Neural trojan attacks inject machine learning systems with hidden behavior that lies dormant until activated. In recent years, trojan detection has emerged as a promising avenue for defending against standard trojan attacks. However, there have been few investigations on trojans specifically designed to be difficult to detect. We organized the Trojan Detection Challenge to begin work on the important question of how to build more robust trojan detectors. This paper gives an overview of the competition and its results. Notably, participants greatly improved over strong baselines on trojan detection and reverse-engineering tasks, demonstrating the potential for proactively improving the robustness of trojan detectors. We hope the competition and its results will inspire further research in detecting hidden behavior in machine learning systems.

Keywords: trojan detection, trojans, hidden behavior, monitoring, security, ML Safety

1. Introduction

In a neural trojan attack, an adversary injects hidden behavior causing a model to produce specific outputs in response to a trigger chosen by the adversary. Crucially, the model behaves normally on clean inputs that do not include the trigger, so a trojan attack may go unnoticed until after the damage is done. Therefore, it is important to detect whether a model contains a trojan before deploying the model. This is the aim of trojan detection,

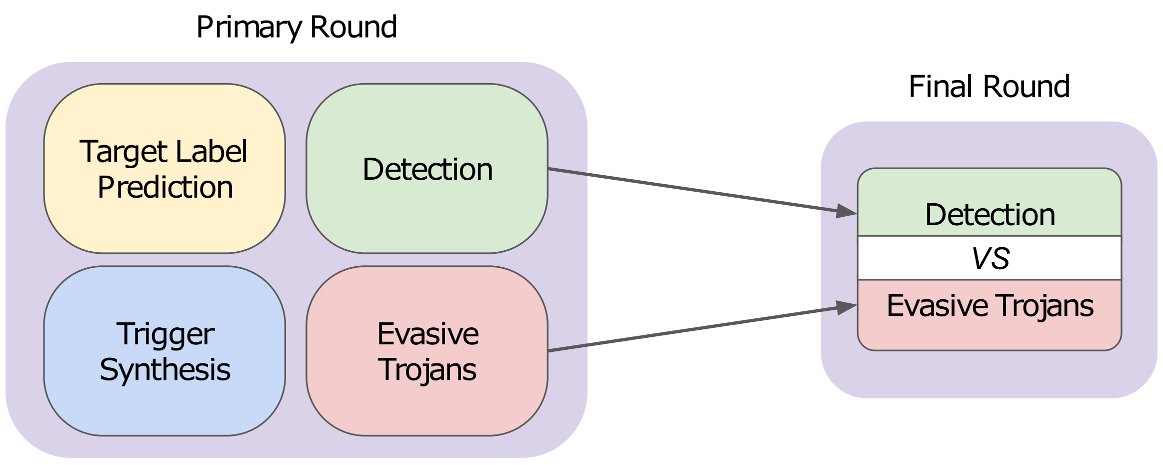

Figure 1: The competition consisted of five tracks across two rounds: four tracks in the Primary Round and one in the Final Round. The winning method of the Evasive Trojans Track in the Primary Round was used to train hard-to-detect trojans for the Final Round, which participants competed to detect.

which seeks to identify whether a neural network has been subjected to a trojan attack and, more broadly, whether a network contains adversarial hidden behavior.

There has been substantial interest in the problem of trojan detection. After neural trojan attacks were first demonstrated (Geigel, 2013; Gu et al., 2017; Chen et al., 2017), many methods were proposed for detecting these attacks. These include model-level, input-level, and dataset-level detection methods. We are primarily interested in model-level detection, where the goal is to detect whether individual models contain a trojan without having access to any trigger-embedded inputs. A range of approaches exist for this task, including backdoor scanning (Wang et al., 2019; Liu et al., 2019; Guo et al., 2019; Tao et al., 2022) and meta-networks (Kolouri et al., 2020; Xu et al., 2021; Zheng et al., 2021).

In addition to academic papers on the subject, the NIST TrojAI competition was established by government researchers in 2019 to study the problem of trojan detection. This competition pushed the boundaries of trojan detection by exploring numerous domains and tasks. However, important unanswered questions remain about the difficulty of trojan detection itself. Recent detection methods obtain high performance on existing trojan attacks, in many cases exceeding 90% AUROC (Liu et al., 2022), yet there is some evidence that trojan attacks could be designed so that the resulting trojans are much harder to detect (Goldwasser et al., 2022).

The question of whether Trojans are easy or hard to detect is deep, and the current experimental answers do not settle this question. Answering this question is important for theoretical and practical reasons, and it could provide important insights for the interpretability of neural networks. Our competition focuses on this question in particular, providing an opportunity for the community to sharpen and expand its understanding of trojan detection: Not only is it important to develop new trojan detectors that are more reliable and data-efficient, but it is also important to develop trojans that are more evasive to clarify the balance between offense and defense. We believe a competition is the ideal format for exploring these deep questions, and we hope that the results and insights from this competition will inspire further research and innovation on trojan detection.

2. Competition Description

The Trojan Detection Challenge was designed to provide provide a new basis of comparison for advancing trojan detection methods and to help answer some of the known challenging questions surrounding the difficulty of this task. The competition consists of five tracks across two rounds. The Primary Round contains the Trojan Detection Track, the Target Label Prediction Track, the Trigger Synthesis Track, and the Evasive Trojans Track. The Target Label Prediction Track and Trigger Synthesis Track are viewed as subtracks of an overarching Trojan Analysis Track, but for brevity we consider them as separate full tracks. The Final Round contains a single track in which participants are challenged to detect trojans created with the winning method from the Evasive Trojans Track. The structure of the competition is visualized in Figure 1. We briefly describe each track below.

- *Trojan Detection Track*: Participants are given a dataset of trojaned and clean networks spanning multiple data sources, which they can use to train a trojan detector. The task is to classify a test set of networks with held-out labels (trojan, clean). This is a binary classification task, and performance is measured with area under the ROC curve (AUROC).
- *Target Label Prediction Track*: Participants are be given a dataset of networks that are known to be trojaned, and the task is to predict the target label of the trojan attack. This is a multiclass classification task, and performance is measured with accuracy.
- *Trigger Synthesis Track*: Participants are be given a dataset of networks that are known to be trojaned, and the task is to predict the location and shape of the trigger patch. This is a binary segmentation task, and performance is measured with IoU.
- *Evasive Trojan Track*: Participants are given a dataset of clean networks and a list of attack specifications. The attack specifications detail what kind of trojan to insert. Participants train a small set of trojaned MNIST networks and upload them to an evaluation server. The server verifies that the attack specifications are met, then trains and evaluates a set of baseline trojan detectors using held-out clean networks and the submitted trojaned networks. The task is to create trojaned networks evade the baseline detectors. Performance is measured with the maximum AUROC of the baseline detectors, and the objective is to lower this number.

The Trojan Detection Track evaluates participants' proposed detectors in diverse and challenging settings. Once a trojan is identified, one may wish to know more about it to mount a successful defense or understand the attacker's intentions. Thus, the Target Label Prediction and Trigger Synthesis tracks consider complimentary analysis tasks. The target label and trigger of an attack are important properties to know for practical reasons, and recent work has begun exploring how well they can be predicted (Harikumar et al., 2020; Wang et al., 2020).

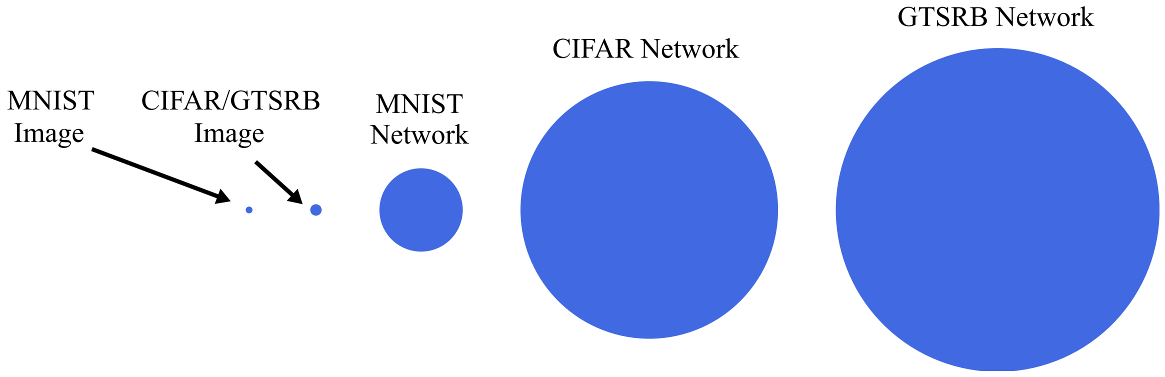

Figure 2: Relative dimensionality of inputs to neural networks and inputs to trojan detection methods. The network sizes are for networks in the competition dataset. For trojan detectors, the inputs are themselves neural networks. Thus, trojan detection requires distinguishing potentially subtle patterns in extremely high-dimensional objects.

The Evasive Trojan Track is an entirely new problem setting. This track naturally compliments the detection track by setting up the standard security research dynamic of offense versus defense. To take full advantage of this setting, we host a *Primary Round* and *Final Round* for the competition. In the Primary Round, participants in all tracks compete against each other and baselines from the literature. In the Final Round, participants of the Trojan Detection Track run their detectors on trojans created with the winning method from the Evasive Trojan Track. Thus, our competition has an interactive element where the contributions of participants from the detection and evasion tracks provide even greater clarity than either track would on its own.

Security Concerns. The Evasive Trojan Track focuses on developing trojans that are harder to detect and therefore raises questions of ethical disclosure and the value of gain-of-function research in adversarial tasks. We believe there are several factors mitigating this risk: (1) The trojan research community has developed a healthy back-and-forth between novel attacks and defenses that subsequently neutralize these attacks. (2) We designed the Trojan Detection Challenge with researchers in mind and have provided resources to make it easy for future work to build on our findings, providing preventative security against adversaries. (3) We limit the Evasive Trojans Track to small-scale toy settings, exploring the feasibility of hard-to-detect trojans without creating actually harmful trojans in state-of-the-art models. In total, be believe the academic benefits from careful exploration of developing hard-to-detect trojans far outweigh the security concerns.

Finally, as a general point, if there really are potentially undetectable, serious trojan attacks available, it would be better to know this than to be unaware. If this knowledge is in the open, defenders can at least know what the risks are, who can mount the attacks, and how to mitigate them.

2.1. Datasets

Standard tasks in the machine learning literature use inputs such as images, text, or speech. By contrast, the input to a trojan detection algorithm is an entire neural network. Thus, to create datasets for our competition, we generate thousands of clean and trojaned neural networks. This leads to a much higher input dimensionality, which we visualize in Figure 2.

Trojan Attacks. We train trojaned networks with patch and whole-image attacks. These attacks are variants of the foundational BadNets and blended attacks (Gu et al., 2017; Chen et al., 2017) modified to be harder to detect. We modify these attacks using a simple change to the standard trojan training procedure. Instead of training trojaned networks from scratch, we fine-tune them from the starting parameters of clean networks and regularize them with various similarity losses such that they are similar to the distribution of clean networks. Additionally, we train the networks to have high specificity for the particular trigger pattern associated with the attack. This method for training evasive trojans is described in an accompanying paper (Mazeika et al., 2023), where we show that baseline detectors obtain substantially lower performance on these hard-to-detect trojans.

All patch attacks in our datasets use random trigger patterns sampled from an independent Bernoulli 0/1 distribution for each pixel and color channel (for trojan detection and target label prediction, patches are black-and-white; for trigger synthesis, patches are colored). Each patch attack uses a different location and size for its trigger mask. All blended attacks in our datasets use random trigger trigger patterns sampled from an independent Uniform(0,1) distribution for each pixel and color channel. All attacks are all-to-one with a random target label.

Data Sources and Network Architectures. To create a more robust evaluation setting, we use a variety of data sources commonly used in academic papers on trojan detection: MNIST, CIFAR-10, CIFAR-100, and GTSRB. For MNIST, we train 5-layer convolutional networks. For CIFAR-10 and CIFAR-100, we train Wide Residual Networks (Zagoruyko and Komodakis, 2016). For GTSRB, we train Vision Transformers (Dosovitskiy et al., 2020). As hard-to-detect trojans have not yet been explored for NLP models, we do not include NLP models in the initial Trojan Detection Challenge. We do not believe that this hinders the academic value of the competition, as detection methods developed for vision models have been shown to transfer to NLP models Zhang et al. (2021). To facilitate detection methods that require input images from the training domain, we allow participants to freely use the data sources used for training the networks in their detection algorithms.

Network Datasets. We train large datasets of clean and trojaned neural networks for the Trojan Detection, Target Label Prediction, and Trigger Synthesis tracks. For the Detection Track, we train $3,000$ networks evenly split across training, validation, and test splits, the four data sources, and the two attack types (patch and blended). Half of the networks in the Detection Track are trojaned.

For both the Target Label Prediction and Trigger Synthesis tracks, we train $2,000$ networks split into 500 training, 500 validation, and $1,000$ test networks. All networks in these tracks are trojaned, and the Trigger Synthesis Track only uses patch triggers. Networks are evenly split across the four data sources.

For the Final Round, we train $1,000$ MNIST networks using one of the top-3 methods for training evasive trojans from the Evasive Trojans Track. Specifically, we use the method from the NJUST team, which greatly improved over the baseline trojan attack in the Evasive Trojans Track. Half of the networks are trojaned, and trojans are evenly split across the

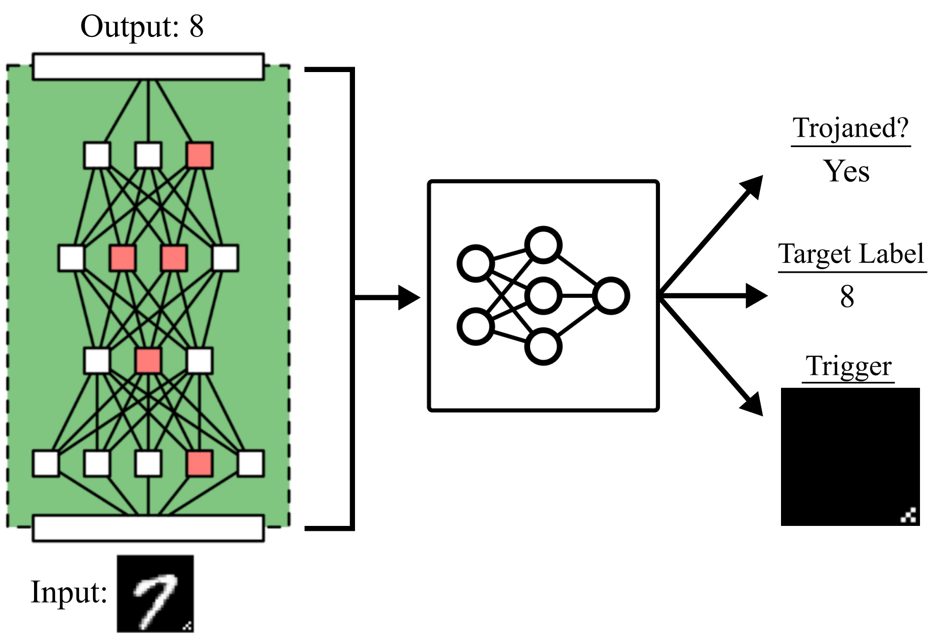

Figure 3: Along with trojan detection, we consider two reverse-engineering tasks: predicting the target label of a trojan attack and predicting the trigger mask. Here we visualize a meta-network that takes a potentially trojaned network as input and performs all three tasks at once. (Figure adapted from (Gu et al., 2017).)

two attack types (patch and blended). In total, we train $8,000$ networks for these tracks. These networks occupy over 60 GB of storage.

Evasive Trojan Track Data. For the Evasive Trojan Track, we use MNIST as a data source for training a distribution of clean networks. We train 600 clean models, split evenly into training, validation, and test sets. Participants can download the training set, which provides a sample from the clean network distribution. Participants are required to train 200 Trojaned models matching a set of provided attack specifications. In the evaluation server, we train baseline detectors on the submitted trojaned networks and held-out clean networks. The validation and test sets are used in the evaluation server only and are not revealed to participants.

2.2. Tasks and application scenarios

Below, we describe the three detection and analysis tasks in the Trojan Detection Challenge in more detail. These tasks are visualized in Figure 3.

Trojan Detection. Participants are given a dataset $X = \{(f_i, y_i) \mid i \in [N]\}$ of (network, label) pairs. The labels are binary: 1 indicates the presence of a Trojan in f, and 0 indicates the absence of a Trojan in f. The task is to train a trojan detector h that takes a network f as input and reliably predicts y. The detector may use additional inputs, such as clean examples from the data source used to train f. However, it does not have access to inputs with Trojan triggers, as the trigger is only known by the adversary. This is the standard formulation of the trojan detection task, and it has practical relevance for scanning model sharing libraries and verifying that trained models do not contain trojans.

Target Label Prediction. Participants are given a dataset $X = \{(f_i, y_i) \mid i \in [N]\}$ of (network, label) pairs, where the labels are the target labels of the trojan attack. The trojan attacks we consider are all-to-one, meaning that regardless of the original label of the network input, the presence of the trojan trigger causes the network to output a fixed target label. Thus, the labels are in the label space of the classification task of, e.g., CIFAR-10 or MNIST. Unlike the Trojan Detection task, all networks are trojaned, and the goal is to understand what kind of trojan they contain. This is a multiclass classification task.

Trigger Synthesis. Participants are given a dataset $X = \{(f_i, y_i) \mid i \in [N]\}$ of (network, label) pairs, where the labels are the binary segmentation masks defining the location and shape of the trojan trigger. Henceforth we refer to this as the trigger mask. In this track, all trojan attacks are patch attacks, and each trigger mask is rectangular and in a randomly-selected location. This is a binary segmentation task.

To improve fairness for the above three tasks, we require that participants do not train additional trojaned networks themselves, as this could give participants with substantial compute resources an unfair advantage. In the competition, this was enforced on an honor system and a requirement for winning teams to share code. The submitted code was manually verified to check that it followed this rule and the other rules of the competition.

Evasive Trojan Track. Participants are given a dataset $X = \{f_i \mid i \in [N]\}$ of clean networks and a set $S = \{s_i \mid i \in [N]\}$ of attack specifications. Each attack specification s includes the target functionality and trigger pattern of a Trojaned network that the participant needs to train. We require that trojaned networks have attack success rates greater than 97%, and we set $N = 200$. Classes for the model architecture and data loader are provided. The participant is allowed to use any procedure for training their trojaned networks as long as the attack specifications are met. Let X_t denote the set of N Trojaned networks trained by the participant. They submit X_t to the evaluation server, which trains an MNTD baseline detector to distinguish X_t from a held-out set of clean networks drawn from the same distribution as X (the validation set at first, and the test set for the final ranking).

2.3. Metrics

Here, we describe the metrics used in each track.

Trojan Detection Track and Final Round. Trojan detection performance is measured using *area under the receiver operating characteristic curve* (AUROC), which is a threshold-free binary classification metric. The AUROC can be interpreted as the probability that a randomly sampled positive example has a higher score than a randomly sampled negative example Fawcett (2006). Higher is better for this metric.

Target Label Prediction Track. Submissions are primarily evaluated with accuracy on the held-out dataset of networks. We also compute accuracy separated by data source (CIFAR-10, CIFAR-100, GTSRB, MNIST), and we break ties in the leaderboard using

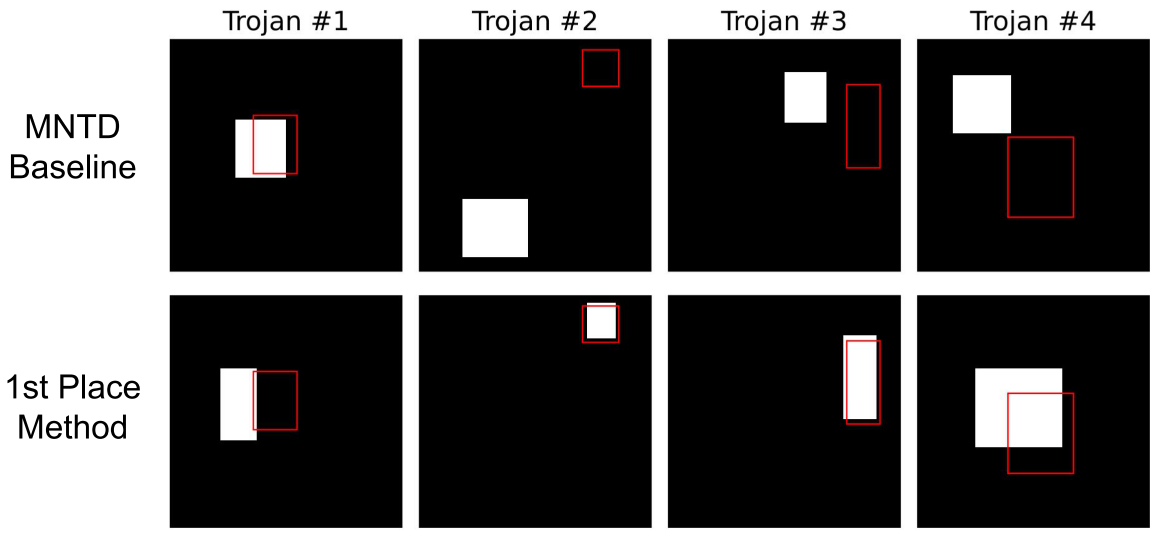

Figure 4: Here, we visualize the outputs of two trigger synthesis algorithms (rows) on four trojaned networks (columns). White squares indicate predicted trigger masks, and red outlines indicate the ground-truth trigger masks. The MNTD baseline in the top row obtains a very low IoU of 5.5%, and its predicted masks rarely overlap the ground-truth mask. By contrast, the winning method from the Trigger Synthesis Track predicts masks that often overlap with the ground-truth mask, obtaining an IoU of 28.4%.

accuracy on CIFAR-10. In practice, the need to break ties did not arise. Higher is better for this metric.

Trigger Synthesis Track. Submissions are evaluated using intersection over union (IoU) between the predicted and true mask for the trojan trigger. Due to the high degree of variability in results, we did not include a tie-breaking metric. Higher is better for IoU.

Evasive Trojan Track Performance is measured using the AUROC of three detectors evaluated on the trojaned networks submitted to the evaluation server. The detectors are MNTD and the accuracy-based and specificity-based detectors from (Mazeika et al., 2023). The MNTD detector requires training, so the evaluation server uses a held-out set of clean networks along with the submitted trojaned networks for this purpose. Participants submit 200 networks per evaluation. As this is a relatively small number, we use cross-validation for the MNTD detector and average the AUROC over 5 folds of the submitted networks and held-out clean networks. Submitted networks must also satisfy attack specifications. An attack specification requires a sufficiently high attack success rate (ASR) for a given trojan (target label, trigger). This is measured as the probability of predicting the target label on a test set where all images have the desired trigger.

2.4. Baselines and Resources

For trojan detection, our baseline methods are *MNTD* (Xu et al., 2021), *Neural Cleanse* (Wang et al., 2019), and *ABS* (Liu et al., 2019). These are some of the strongest and most

Team Name	AUROC
🥇 Exness	99.98
🥈 QCRI	99.96
🥉 HRI	99.5
NC Baseline	73.7
MNTD Baseline	63.4
ABS Baseline	62.9

(*a*) Detection

Team Name	Accuracy
🥇 HRI	98.6
🥈 PurdueRutgers	96.2
🥉 Exness	95.5
NC Baseline	55.7
ABS Baseline	41.6
MNTD Baseline	28.6

(*b*) Target Label Prediction

Team Name	IoU
🥇 HRI	28.4
🥈 PurdueRutgers	28.1
🥉 Exness	11.1
NC Baseline	6.6
MNTD Baseline	5.5

(*c*) Trigger Synthesis

Team Name	Max AUROC
🥇 Di Tang	49.3
🥈 NJUST	52.6
🥉 HRI	54.2
Evasive Trojans Baseline	73.9
Standard Trojans Baseline	94.1

(*d*) Evasive Trojans

Team Name	AUROC
🥇 Di Tang	100.00
🥇 IUB	100.00
🥉 Exness	99.7
NC Baseline	60.3
MNTD Baseline	49.8
ABS Baseline	49.0

(*e*) Final Round

Figure 5: Results of the winning methods on the test set in each track. Across all tracks, participants greatly improved over strong baselines, demonstrating that trojan attacks that evade existing detectors may still be discovered with stronger detectors.

well-known current detectors. For the target label prediction and trigger synthesis tasks, we modify MNTD's output head to predict multiclass labels and bounding box coordinates, respectively. Neural Cleanse and ABS natively predict target labels and trigger masks, so they require no modification. Additionally, since ABS was primarily designed for detection and not for trigger synthesis, we do not use it as a trigger synthesis baseline.

For the Evasive Trojans Track, our baseline methods are the Standard Trojans and Evasive Trojans methods from the accompanying paper describing these attacks. For the Evasive Trojans method, we do not use the randomization loss described in the paper.

Starter Kit. We provide participants with a starter kit, which includes implementations of the MNTD baseline for each track, data loading tools, and code utilities such as boilerplate code and code for running evaluations locally on validation splits of the training sets. The starter kit also links to the Zenodo download links for the competition datasets. The starter kit is available at https://github.com/mmazeika/tdc-starter-kit.

3. Competition Results

Across all tracks of the competition, 193 teams joined, 70 teams made submissions, and 1,210 submissions were made in total. Participants greatly improved over the baselines in all five competition tracks and reached very high performance in four out of the five tracks.

3.1. Main Results

In Figure 5, we show the results of the top-3 submissions to the test set of each track alongside the baseline methods. In the Detection Track, Target Label Prediction Track, and Final Round, the first-place methods obtained near-perfect performance. This is surprising given the low performance of the baseline methods, and it suggests that evasiveness of a trojan attack against existing detectors is not a strong assurance of evasiveness against all possible detectors.

In the Trigger Synthesis Track, participants made large improvements over the baselines but are still far from the ceiling of perfect IoU. Note that it may not be reasonable to expect perfect IoU from these methods due to the possibility of trojan attacks inadvertently inserting multiple triggers (Qiao et al., 2019). However, this is mitigated in our evaluations due to the specificity loss used to train the trojaned networks in our dataset. This specificity loss causes the network to behave normally on inputs with incorrect triggers, which makes our trigger synthesis task well-defined. In Figure 4, we show qualitative results from the MNTD baseline and the first-place trigger synthesis method. The ground-truth trigger is outlined in red, and the predicted mask is shown in white. Predictions from the winning method overlap with the ground-truth trigger most of the time, while MNTD predictions often do not overlap with the ground-truth trigger.

In the Evasive Trojans track, submitted trojans obtained high levels of evasiveness against the baseline detectors. Due to an unforeseen issue with our evaluation metric, participants were incentivized to reduce the AUROC on the fixed accuracy-based and specificity-based baseline detectors below 50%. This means that by taking the negative score of these detectors, one could detect the submitted trojans with better-than-chance performance. However, even after taking this into account, all winning methods outperformed the baseline trojans. For MNTD, the winning methods reduced detection performance to chance levels. This is surprising, since MNTD is trained in a white-box manner on the submitted trojans and is a strong detector against standard trojan attacks. To the best of our knowledge, this is the first time that MNTD has been completely fooled in a white-box manner.

In the Final Round, participants competed to detect trojaned networks created with the winning method submitted by the NJUST team to the Evasive Trojans Track. For these trojaned networks, we evaluated Neural Cleanse, MNTD, and ABS baseline detectors, finding that all three performed significantly worse than on our baseline Evasive Trojans. Despite this, participants obtained 100% AUROC for detecting these trojans. This corroborates our finding from the Detection Track that a trojan attack that is highly evasive against existing detectors may still be beaten by developing new detectors.

3.2. Overview of Methods

Participants used a variety of methods and techniques across the different tracks. An overview of these methods is provided here.

Backdoor Scanning. Many participants used backdoor scanning methods, a category of detector that searches for triggers across different target labels by optimizing a candidate trigger to satisfy some objective. Many variations on this approach exist, including Neural Cleanse (Wang et al., 2019), ABS (Liu et al., 2019), TABOR (Guo et al., 2019), K-Arm

(Shen et al., 2021), and PixelBackdoor (Tao et al., 2022). These methods typically do not require training sets of networks, although they do benefit from hyperparameter tuning. For example, the PurdueRutgers team's submissions to the Target Label Prediction and Trigger Synthesis tracks use a combination of methods from the K-Arm and PixelBackdoor detectors, and the HRI and Exness teams used trigger inversion as part of their approach.

Meta-Networks. Some participants explored meta-networks. There are relatively few meta-network methods in the literature, with the primary examples being query-based meta-networks (Oh et al., 2019; Kolouri et al., 2020; Xu et al., 2021). These methods optimize network queries along with shallow classifiers on the stacked outputs of the networks on the queries. MNTD is an example of such a method (Xu et al., 2021). The Exness and IUB teams developed innovative meta-networks that operate directly on network parameters, obtaining first place in the Detection Track and Final Round.

The Exness team computed simple summary statistics from each layer's parameters, concatenated them together, and trained simple regressors on top of these features. This ameliorated the dimensionality problem inherent in directly classifying network parameters. This simple approach achieves strong performance on MNIST, CIFAR-10, and CIFAR-100 networks from the competition. The IUB team obtained high performance in the Final Round using a deep time series classifier directly on network parameters.

The strong performance of these methods is surprising, since the dimensionality of neural networks is much larger than the dimensionality of typical network inputs, as illustrated in Figure 2. This suggests that inserting trojans without leaving a large mark on network parameters may be challenging, which is good news for defenders.

Combined Methods. Most winning submissions used different approaches for different data sources and network architectures. While this was effective on our competition datasets, an interesting question for future work is whether unified detection methods can perform similarly well.

4. Conclusion

We organized the Trojan Detection Challenge, a competition aimed at advancing the state of the art in trojan detection and understanding the difficulty of the problem. By investigating hard-to-detect trojans, we sought to uncover important insights into the balance between offense and defense in neural trojan attacks. Through the participation of numerous teams, we gained important insights into the effectiveness and limitations of trojan detection techniques and the evasiveness of trojan attacks. We hope that the Trojan Detection Challenge will serve as a catalyst for further research in this area and help to improve methods for monitoring hidden behavior in neural networks.

Acknowledgments

Thank you to Open Philanthropy, who provided the prize fund. Thank you also to the participants and the NeurIPS competition chairs for making this possible.

References

Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

Arturo Geigel. Neural network trojan. *Journal of Computer Security*, 21(2):191–232, 2013.

Shafi Goldwasser, Michael P Kim, Vinod Vaikuntanathan, and Or Zamir. Planting undetectable backdoors in machine learning models. *arXiv preprint arXiv:2204.06974*, 2022.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

Wenbo Guo, Lun Wang, Xinyu Xing, Min Du, and Dawn Song. Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems. *arXiv preprint arXiv:1908.01763*, 2019.

Haripriya Harikumar, Vuong Le, Santu Rana, Sourangshu Bhattacharya, Sunil Gupta, and Svetha Venkatesh. Scalable backdoor detection in neural networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 289–304. Springer, 2020.

Soheil Kolouri, Aniruddha Saha, Hamed Pirsiavash, and Heiko Hoffmann. Universal litmus patterns: Revealing backdoor attacks in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 301–310, 2020.

Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. Abs: Scanning neural networks for back-doors by artificial brain stimulation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1265–1282, 2019.

Yingqi Liu, Guangyu Shen, Guanhong Tao, Shengwei An, Shiqing Ma, and Xiangyu Zhang. Piccolo: Exposing complex backdoors in nlp transformer models. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 2025–2042. IEEE, 2022.

Mantas Mazeika, Andy Zou, Akul Arora, Pavel Pleskov, Dawn Song, Dan Hendrycks, Bo Li, and David Forsyth. How hard is trojan detection in DNNs? fooling detectors with evasive trojans, 2023.

Seong Joon Oh, Bernt Schiele, and Mario Fritz. Towards reverse-engineering black-box neural networks. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 121–144, 2019.

Ximing Qiao, Yukun Yang, and Hai Li. Defending neural backdoors via generative distribution modeling. *Advances in neural information processing systems*, 32, 2019.

Guangyu Shen, Yingqi Liu, Guanhong Tao, Shengwei An, Qiuling Xu, Siyuan Cheng, Shiqing Ma, and Xiangyu Zhang. Backdoor scanning for deep neural networks through k-arm optimization. In *International Conference on Machine Learning*, pages 9525–9536. PMLR, 2021.

Guanhong Tao, Guangyu Shen, Yingqi Liu, Shengwei An, Qiuling Xu, Shiqing Ma, Pan Li, and Xiangyu Zhang. Better trigger inversion optimization in backdoor scanning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13368–13378, 2022.

Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723. IEEE, 2019.

Ren Wang, Gaoyuan Zhang, Sijia Liu, Pin-Yu Chen, Jinjun Xiong, and Meng Wang. Practical detection of trojan neural networks: Data-limited and data-free cases. In *European Conference on Computer Vision*, pages 222–238. Springer, 2020.

Xiaojun Xu, Qi Wang, Huichen Li, Nikita Borisov, Carl A Gunter, and Bo Li. Detecting ai trojans using meta neural analysis. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 103–120. IEEE, 2021.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Xinyang Zhang, Zheng Zhang, Shouling Ji, and Ting Wang. Trojaning language models for fun and profit. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 179–197. IEEE Computer Society, 2021.

Songzhu Zheng, Yikai Zhang, Hubert Wagner, Mayank Goswami, and Chao Chen. Topological detection of trojaned neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.