Detecting Trojaned DNNs Using Counterfactual Attributions

Karan Sikka  
SRI International  
karan.sikka@sri.com

Indranil Sur  
SRI International  
indranil.sur@sri.com

Anirban Roy  
SRI International  
anirban.roy@sri.com

Ajay Divakaran  
SRI International  
ajay.divakaran@sri.com

Susmit Jha  
SRI International  
susmit.jha@sri.com

Abstract—We target the problem of detecting Trojans or backdoors in DNNs. Such models behave normally with typical inputs but produce targeted mispredictions for inputs poisoned with a Trojan trigger. Our approach is based on a novel intuition that the trigger behavior is dependent on a few ghost neurons that are activated for both input classes and trigger pattern. We use counterfactual explanations, implemented as neuron attributions, to measure significance of each neuron in switching predictions to a counter-class. We then incrementally excite these neurons and observe that the model’s accuracy drops sharply for Trojaned models as compared to benign models. We support this observation through a theoretical result that shows the attributions for a Trojaned model are concentrated in a small number of features. We encode the accuracy patterns by using a deep temporal set encoder for Trojan detection that enables invariance to model architecture and a number of classes.

We evaluate our approach on four US IARPA/NIST-TrojAI benchmarks with high diversity in model architectures and trigger patterns. We show consistent gains over state-of-the-art adversarial attack based model diagnosis (+5.8% absolute) and trigger reconstruction based methods (+23.5%), which often require strong assumptions on the nature of the attack.

Index Terms—Deep neural networks, Trojans, Backdoors, Resilience

I. INTRODUCTION

Deep neural networks (DNNs) have emerged as the representation of choice for machine learning models in multiple domains [1, 2, 3]. The ease of training large-scale DNNs with huge amounts of data has made it possible to achieve strong performance on several benchmarks. Despite these successes, DNNs are known to be fragile and vulnerable to adversarial attacks, which inhibits their adoption in high-assurance safety-critical systems such as autonomous driving and user authentication. The space of adversarial attacks on DNNs is diverse ranging from inference-time adversarial perturbations that lead to an incorrect prediction by the ML model [4], reverse-engineering attacks to infer weights of a trained DNN [5], and training-time attacks that poison the dataset [6]. These attacks and corresponding defense mechanisms for DNNs have received significant attention in literature [7, 8].

Recent work [9, 10, 11] has demonstrated a new kind of vulnerability where a DNN can be trained with poisoned data to behave as Trojaned. A Trojaned DNN behaves normally with high accuracy on typical inputs but can be made to produce targeted incorrect predictions when the inputs contain the Trojan trigger. This paper focuses on devising a verification defense against such Trojan (also called backdoor) attacks (Fig. 1). We develop a principled approach to verify if a trained DNN has been Trojaned, with access only to the trained model and a few clean labeled test samples. Our proposed approach builds on recent progress in explaining the decisions of DNNs, and thus, draws a connection between the interpretability of a DNN and its robustness and resilience to attacks.

A variety of triggers are considered in Trojan attacks. While initial work considered stamp-like triggers [9], invisible triggers have been explored by blending triggers with benign samples [10]. Another approach to producing stealthy trigger is to perturb the benign sample by a backdoor trigger amplitude instead of using a stamp-like patch to replace the sample pixels [12]. We evaluate our method on the large-scale US IARPA/NIST-TrojAI Datasets1 which includes a diverse set of Trojaned models with a variety of triggers, and thus requires generalizable Trojan detection approaches beyond the existing state of the art.

Trojaned models are known to have shortcuts in the feature space [13, 14, 15] that allow triggers to switch the model’s

1https://pages.nist.gov/trojai/docs/data.html
prediction from other classes to the target class. Existing detection approaches rely on indirect statistical signatures in the form of sensitivity to instance-specific or universal adversarial attacks [15, 16]. This can be limiting since reliance on adversarial probes in activating the shortcut requires a strong assumption on the nature of the trigger. Instead, we use attributions, that measure relevance of each input feature in producing the DNN output [17, 18, 19], to directly detect the presence of these shortcut pathways in a Trojaned model (see Fig. 1). In particular, we use counterfactual attribution across clean samples to assign a score to each neuron for causing mispredictions to a counter-class. We define counterfactual attribution as the attribution corresponding to the a counter-class that is not the predicted class with the highest score [20, 21, 22]. We then incrementally excite these neurons based on the assigned scores and observe changes in model’s accuracy. We define the neurons encoding the shortcut as ghost neurons. We also observe these ghost neurons are “poly-semantic” in nature, that is, they have high attribution for both poisoned and clean inputs. In Theorem 1, we show that the attributions encoding the trigger are concentrated over a small number of neurons. This would cause a sharp performance drop when neurons with high attribution towards the “target” counter-class are activated. We finally encode these class-wise accuracy time-series with a deep temporal set encoder for Trojan detection. Our contributions are:

- We are the first to observe that the Trojan triggers use “poly-semantic” neurons that not only have high attribution for poisoned inputs but also show significant attribution towards target class even for clean inputs.
- Using our observation of poly-semantic neurons, we develop a counterfactual attribution based approach to detect whether a DNN is Trojaned without access to any poisoned input or knowledge about the trigger, and with a few clean inputs. We use counterfactual attributions to identify significance of each neuron in causing mispredictions and incrementally excite them to observe changes in model’s accuracy.
- We propose a deep temporal set encoder to make our Trojan detector invariant to model architecture, and the number, ordering, and semantic diversity of classes in different DNNs, and trigger types.
- We evaluate our approach on US IARPA/NIST-TrojAI Trojan detection benchmarks that contain over 1000 models trained on different datasets and with high diversity in the number and type of classes and the shape and size of triggers. We demonstrate gains of 5.8% and 23.5% over the model diagnosis and trigger reconstruction based SOTA methods respectively.

II. BACKGROUND AND RELATED WORK

In the rest of the paper, we refer to the DNN with the embedded Trojan as Trojaned model, and the model without Trojan as benign model. Training the Trojaned model is often achieved by poisoning a small fraction of the training inputs with the trigger pattern, referred to as poisoned inputs. The original expected output of a poisoned input is called the source class and the output of the Trojaned model is called the target class. The threat of Trojan attacks [9, 10] on DNNs aims at embedding hidden triggers such that the poisoned inputs make the Trojaned DNN mispredict and output the target class even though its accuracy on the clean inputs remains high.

This paper combines the fields of explaining decisions of DNNs and their adversarial robustness. We briefly discuss the related work and compare our approach to existing techniques for detecting Trojaned models.

A. Explainability and Resilience

A number of explanation techniques [17, 23, 24, 25, 26, 27] have been recently proposed that find qualitative explanations or assign quantitative attributions to input features for a given decision. Many of these methods are based on the gradient with respect to the input [17, 28, 29, 30]. A few recent theoretical studies [31] indicate a strong connection between the robustness of DNNs and their interpretability using attribution methods. The connection between these methods for explaining DNN decisions and detection of out-of-distribution data and adversarial examples has been related to anti-causal direction of learning [18, 32, 33]. Recent works have explored counterfactual explanations that explain a DNN decision by identifying what input transformations will switch the prediction to a counter-class [20, 22]. We also exploit counterfactual explanations to identify neurons that predict the target class.

B. Trojan Attacks and Defenses

Inference-time adversarial attacks and defenses for these attacks have received a lot of interest [8, 34, 35]. While adversarial perturbations are often input specific, universal perturbations [36, 37, 38] have been also studied which can change the DNN prediction to a target class for any sample input. While Trojan attack is a training-time attack, the universal adversarial attacks do not require training-time access to the model. Further, the trigger in Trojan is known and deliberately injected into the poisoned DNN while these are found through optimization for universal adversarial attacks. While both of these vulnerabilities are a consequence of poor generalization and low resilience of the DNNs, Trojaned DNNs learn to predict a target class for inputs with the trigger. We use attributions to detect these learned triggers.

The state-of-the-art Trojan insertion methods [6, 9, 9, 10, 39, 40, 41] use a minuscule amount of data poisoned with the Trojan trigger pattern (e.g., a local patch, a filter with specific settings). Alternative methods inject Trojans through transfer learning [42], retraining a DNN [43], direct manipulation of DNN weights [44, 45], or addition of malicious modules [46]. Our approach is independent of the Trojan insertion method (see Section III).

A number of defense methods have been proposed against Trojan attacks. The first class of defenses are inference-time preprocessing of the samples using pre-trained autoencoders [47], style transfer [48], spatial transformations such as shrinking and flipping [49], and the superimposition of various
image patterns and observation of the conformance of prediction [50]. As demonstrated by the US IARPA/NIST-TrojAI datasets, injected triggers can be made robust to many kinds of transformations. Our defense approach detects a Trojaned model by analyzing it and not by preprocessing inputs to the model. Another set of techniques is based on post-processing the trained DNN by retraining it with a set of clean samples to prune and finetune the DNN [51], repair of DNN based on the mode connectivity technique [52, 53]. In contrast, we focus on detecting Trojaned models with a small number of clean samples which are not sufficient for retraining and repair. Yet another class of defense methods focus on reverse engineering the Trojan triggers implemented in Neural Cleanse [13] and DeepInspect [14]. These trigger generation methods have been shown to produce patterns distinct from the ones used in training [54]. GAN based reverse engineering of triggers [55] have also been proposed. While these approaches do well on stamp-like localized triggers, we consider a large variety of Trojan triggers that include image filters and hence, cannot be reverse engineered without unreasonable assumptions on prior knowledge of the trigger. We instead adopt an approach to diagnose the DNN for the presence of Trojan triggers. Model diagnosis approaches for Trojan detection include the use of universal litmus test [56], differential privacy [57], one-pixel signature [58], and a combination of adversarial attacks and feature inversion [15, 16]. In contrast to these methods, our approach does not rely on susceptibility to adversarial perturbations which can be moderated through adversarial training but instead makes use of attributions over a few clean samples. We experimentally compare against both reverse engineering and model diagnosis based SOTA approaches [13, 15, 56] in Section V.

III. PROBLEM DEFINITION

Given a deep learning model \( f \), if the Trojan insertion on a sample \( x \) produces a poisoned sample \( x_p \), then the trigger-insertion relation \( P(x, x_p) \) is said to be true. This relation models multiple poisoned samples that can be created from a single clean sample. For any input \( x \) and a poisoned input \( x_p \) with the source class \( s \) and the target class \( c_t \), \( f(x) = s, \forall x \ P(x, x_p) \Rightarrow f(x_p) = c_t \). In contrast to many existing models of triggers as a transformation function, we model it as a relation because trigger transformation need not be a unique function. For example, a polygon trigger can be of different shape, color, size and position, and a filter trigger can be active over a wide set of filter parameters. For a trigger to be exploitable in practice, it must be robust to small perturbations. We also expect the trigger transformation to satisfy some notion of smallness in change to avoid detection by online input filtering methods. These notions of smallness include \( L_1 \) norm \( \|x_p - x\|_1 \) distance for \( l = 1, 2, \infty \), and application of Instagram filters such as Gotham and and Nashville (e.g. US IARPA/NIST-TrojAI dataset). Filters represent changes which might be large in \( L_p \) norm space but these perturbations capture physically realizable input variations.

A Trojan detection method is required to identify if a DNN \( f \) over prediction classes \( C = \{c_1, c_2, \ldots, c_m\} \) contains a Trojan with a perturbation relation \( P \), source classes \( C_s \subseteq C \), and the target class \( c_t \in C \setminus C_s \), that is, for all inputs \( x \) in the input domain \( D \) of the model,

\[ \forall x_p \ P(x, x_p) \ [f(x) \in C_s \Rightarrow f(x_p) = c_t] \]

Trojaned models have good performance on clean inputs. Typically, we do not know the trigger perturbation \( P \), the source classes \( C_s \) or the target class \( c_t \). Our attack model has the following characteristics to ensure its relevance to the real-world challenges:

- No access to the training data.
- No knowledge of the trigger perturbation. Our experiments include polygon and filter triggers to demonstrate generalizability of the approach.
- No knowledge of the source and target classes
- No knowledge about the training method used in training the DNN with poisoned data.

We assume the following for our defense approach:

- Assumption 1: We have whitebox access to the DNN - the architecture and the learned weights.
- Assumption 2: We have a small set of clean inputs which could be different from the training data.
- Assumption 3: The Trojaned model is robust with respect to small changes in the trigger.

These characteristics allow us to model real-world adversaries such as outsourced providers of deep learning models using proprietary training data and algorithms. The first two assumptions are reasonable for verifying a DNN acquired from an untrusted source. We can expect to have a small set of labeled clean inputs on which the model performs correctly even if it is Trojaned. The third and the final assumption is important for a real-world robust attack so that the attacker can effectively use the trigger to change model’s prediction in a noisy environment.

IV. TROJANED DNN DETECTION APPROACH

The first step in our approach (shown in Fig. 2) is the identification of the suspicious neurons that might cause the Trojaned behavior. We achieve this by computing feature attributions (that measure the degree to which a neuron is responsible for a DNN’s output) for predicting counter-classes over the clean samples. We call such attribution as “counterfactual attribution” since these measure the contribution of the features in predicting a counter-class that is not the (predicted) highest scoring class [20, 21, 22]. The neurons which have high attribution consistently across the clean samples for the same counter-class likely encode the trigger with the counter-class being the target class. We refer to these suspect neurons as ghost neurons since they indicate the presence of trigger without it being active in clean samples. The second step of our approach excites these ghost neurons and observes changes in model’s accuracy. If these truly encode the trigger and the DNN is Trojaned, the model’s accuracy will drop
The loss function for training a Trojaned model is 

\[ \mathcal{L}(x, c) = \sum_{x} \mathcal{L}(f(x), c_x) + \sum_{x_p} \mathcal{L}(f(x_p), c_t) \]

Without loss of generality, let us assume that there are only two classes 1 and -1. In order to make the Trojaned DNN robust to \( \delta \) perturbations, the loss function is modified to minimize 

\[ \mathcal{L}(f(x_p + \delta), c) \]

for an input \( x_p \). Typical loss functions such as negative log likelihood or hinge-loss can be written in the form of 

\[ g(-c(z, w)) \]

where \( g \) is a non-decreasing function. Let a subset of the features \( s \subseteq z \) correspond to the trigger. For each of the features, the expected SGD update for each \( w_i \) is 

\[ \Delta_i = -\mathbb{E}[\partial \mathcal{L}(f(x_p + \delta), c)/\partial w_i] = \mathbb{E}[g'(c_i)\Delta_i] - \mathbb{E}[g(c_i)\Delta_i] = -\mathbb{E}[g'(c_i)\Delta_i] + \mathbb{E}[g(c_i)\Delta_i] \]

where \( \Delta_i \) is the worst-case perturbation in \( z \) corresponding to the change in \( \delta \) in \( x_p \). We now consider the quantity 

\[ \Delta_s = \sum_{i \in s} w_i \Delta_i/\sum_{i \in s} |w_i| \]

which has a natural interpretation as change in concentration of the attributions. The high positive value of \( sgn(w_i)\Delta_i \) means expansion while high negative value means shrinkage. \( \Delta_s \) models the weighted expansion or shrinkage across attributions. Further, 

\[ \Delta_s \leq \mathbb{E}[g'(\delta_w - c(z, w))\Delta_s] \]

where \( \gamma_s = c \sum_{i \in s} \Delta_i/\sum_{i \in s} |w_i| \) is the output \( c \) aligned weighted strength of the features. If \( \gamma_s \) is not aligned, \( \Delta_s \) pushes the attributions to 0 and if \( \gamma_s \) is aligned but not concentrated enough to be larger than \( \delta_w \), then \( \Delta_s \) still pushes the attributes to 0 making the attributions more concentrated. Thus, by increasing the robustness \( \delta_w \) of the trigger, the features corresponding to the Trojan become further concentrated.

We make the following observations that motivate our technical approach. The first observation is used to develop our Trojaned model detection approach and follows from the concentration of attributions. The second central observation is the manifestation of these features as having high attribution concentration. The second central observation is quickly since the trigger gets activated. The drop in accuracy is more gradual for benign DNNs. Finally, we take the class-wise changes in accuracy and use a deep set encoder to make the final prediction about the DNN being Trojaned. The deep set encoder ensures that our approach is insensitive to the diversity in the DNNs. We describe each of these steps below.

A. Attribution-based Identification of Ghost Neurons

We use the penultimate layer (the layer before the softmax) for input \( x \) to a DNN. These neurons are denoted by \( z \) and each feature by \( z_i \). Our approach can be applied to any projection of \( x \) over a feature space including directly using the pixels. The use of penultimate features has the advantage of using semantically meaningful features. We first show that the Assumption 3 on the robustness of trigger implies that the attributions for the features encoding this trigger are concentrated on a few neurons. This will, in turn, explain the observed quick deterioration in accuracy of the DNN when these features are excited in Trojaned models.

For robustly trained Trojaned model, we expect the model to produce the correct output on clean data \( f(x) = c_x \) and its perturbations \( f(x + \delta) = c_x \), and produce the target output \( c_t \) on poisoned data \( f(x_p) = c_t \) and its perturbations \( f(x_p + \delta) = c_t \) for the robustness threshold \( \delta \). The result is summarized in the theorem below where the attributions of the features are simply the weights \( w \) of the last layer \( z \) of the DNN.

**Theorem 1.** The stochastic gradient descent update for robustly training a Trojaned DNN concentrates the attributions over a small set of features encoding the trigger.

**Proof.** The loss function for training a Trojaned model is 

\[ \mathcal{L} = \sum_x \mathcal{L}(f(x), c_x) + \sum_{x_p} \mathcal{L}(f(x_p), c_t) \]

Without loss of generality, let us assume that there are only two classes 1 and -1. In order to make the Trojaned DNN robust to \( \delta \) perturbations, the loss function is modified to minimize 

\[ \mathcal{L}(f(x_p + \delta), c) \]

for an input \( x_p \). Typical loss functions such as negative log likelihood or hinge-loss can be written in the form of 

\[ g(-c(z, w)) \]

where \( g \) is a non-decreasing function. Let a subset of the features \( s \subseteq z \) correspond to the trigger. For each of the features, the expected SGD update for each \( w_i \) is 

\[ \Delta_i = -\mathbb{E}[\partial \mathcal{L}(f(x_p + \delta), c)/\partial w_i] = \mathbb{E}[g'(c_i)\Delta_i] - \mathbb{E}[g(c_i)\Delta_i] \]

where \( \Delta_i \) is the worst-case perturbation in \( z \) corresponding to the change in \( \delta \) in \( x_p \). We now consider the quantity 

\[ \Delta_s = \sum_{i \in s} w_i \Delta_i/\sum_{i \in s} |w_i| \]

which has a natural interpretation as change in concentration of the attributions. The high positive value of \( sgn(w_i)\Delta_i \) means expansion while high negative value means shrinkage. \( \Delta_s \) models the weighted expansion or shrinkage across attributions. Further, 

\[ \Delta_s \leq \mathbb{E}[g'(\delta_w - c(z, w))\Delta_s] \]

where \( \gamma_s = c \sum_{i \in s} \Delta_i/\sum_{i \in s} |w_i| \) is the output \( c \) aligned weighted strength of the features. If \( \gamma_s \) is not aligned, \( \Delta_s \) pushes the attributions to 0 and if \( \gamma_s \) is aligned but not concentrated enough to be larger than \( \delta_w \), then \( \Delta_s \) still pushes the attributes to 0 making the attributions more concentrated. Thus, as we increase the robustness \( \delta_w \) of the trigger, the features corresponding to the Trojan become further concentrated. □
Observation 1: The robust triggers in a Trojaned model are encoded using a few features in the penultimate layer of a Trojaned model.

Observation 2: When examining the counterfactual attribution over the features in the penultimate layer on decisions on clean samples, if the counter-class is the target class in a Trojaned model, these ghost neurons encoding the trigger exhibit high attribution.

B. Shortcut Pathway in Feature Space

We draw a connection between the approach proposed in this paper and a common hypothesis shared in literature for Trojan detection [13, 15]. This hypothesis states that a shortcut pathway is present inside a Trojaned model and enables the model to predict target class on poisoned input without affecting its performance on clean samples. Using the above two observations, we refine this hypothesis by identifying that such shortcuts are in the form of a collection of a small set of features that activate the trigger behavior. These shortcuts in the model persist even for clean samples and counterfactual analysis can be used to get attributions over these trigger features. This is in contrast to indirect detection of these shortcuts using adversarial attacks because it is not necessary that adversarial examples exploit this shared common shortcut in the model instead of identifying sample-specific perturbations in the case of individual attacks and other perturbations (not the actual injected trigger) in case of universal attacks [15, 56].

We compute the feature attributions for all the model classes encoding using a few features in the penultimate layer of a CNN or a Transformer model with positional encoding. The next step is to compute counterfactual attributions for clean inputs.

C. Counterfactual Analysis by Exciting Ghost Neurons

However, we do not have access to poisoned inputs in real-world settings. We thus modify our approach to rank neurons by computing counterfactual attributions across clean images for predicting target class $c_k$ i.e. $\alpha^c_k(X^c)$, where $X^c$ is the dataset of clean inputs. A key reason is that we are able to use the clean images as proxies for the poisoned images because the ghost neurons are poly-semantic in nature and fire for patterns corresponding to both actual class(es) and the trigger pattern. This allows us to estimate the neuron rankings identified with poisoned images within some error margin, which are then used for our next analysis. Fig. 3 shows attributions $\alpha^c_k$ for the target class across both poisoned and clean inputs for a Trojaned model. We observe that neurons with high attributions for poisoned inputs also have high (counterfactual) attributions for clean inputs.

For the next step, we use the Observation 1 about sparseness of the ghost neurons. This is a critical property that separates ghost neurons from normal neurons and thus benign models from Trojaned models. We exploit this observation by computing the DNN’s performance on the clean samples by gradually exciting neurons with some activation value based on their ranks. If the model is Trojaned, a sharp drop in performance is observed on excitation of a small number of neurons. For example, Fig. 4 shows the model’s performance versus the percentage of excited neurons for a benign and for a Trojaned model. It is evident that the fall in performance is sharp for the Trojaned model. For the activation value we experimented with maximum of the features across all the classes and the mean of the features values across each class and selected them based on empirical performance for each dataset.

D. Addressing Model Diversity by Using a Deep Temporal Set Encoder

Since the target class is unknown, the above approach is repeated for the DNN classes. We can then pick the possible target class based on the steepest descent in the performance curve and extract relevant features such as the rate of fall in performance. However, this does not generalize to DNNs with different number of classes, architectures, training regimes, etc. We address this challenge by proposing a deep temporal set encoder based Trojan detector (DTSE-TD) that contains the right inductive biases to encode the class-wise performance curves. Our model first encodes the performance curves as time-series by using a temporal encoder, that is either a 1D-CNN or a Transformer model with positional encoding. The temporal encoder produces a tensor containing features for all the classes. Next, we use a set encoder that is invariant to the number and ordering of classes since different DNN can have different number of classes. We achieve this by using a permutation invariant encoder that treats the class-level outputs as sets of features and pools their outputs using max-pooling [60, 61]. We pass the pooled output through a linear layer for prediction. We then train our model on a dataset containing both benign and Trojaned DNNs using cross-entropy loss.

V. Experiments

We evaluate our model on four datasets containing Trojaned DNNs trained for image classification. We first describe the datasets and the evaluation metrics. We then discuss qualitative results that also includes comparison with state-of-the-art (SOTA) methods. We finally study the impact of factors that can be varied by an adversary for training Trojaned DNNs on performance.
Fig. 3: Showing neuron attributions for predicting the target class across poisoned and clean inputs for a Trojaned model. We observe that neurons causing the Trojan behavior (ghost neurons) also have high counterfactual attributions (darker red implies higher) across clean inputs.

Fig. 4: The ghost neurons identified using the counterfactual analysis are excited incrementally to monitor accuracy. Figure highlights that the trigger, if present, is localized in a few neurons in a Trojaned DNN as compared to a benign DNN.

A. Datasets and Evaluation Metrics

**Triggered-MNIST:** We use the code provided by NIST\(^2\) to generate 810 DNNs (50% are Trojaned) trained to classify MNIST digits. The DNNs are selected from among three architectures. The Trojaned models are trained with images poisoned with two types of trigger patterns that are randomly inserted into the image. The attack is designed to misclassify some source classes to a target class. Other factors such as the target class and poisoning rate are selected randomly.

**TrojAI-Round1, Round2, Round3:** These datasets are made publicly available by US IARPA/NIST\(^3\) and contains models trained for traffic sign classification (50% are Trojaned). The models are trained on synthetically generated image-data of artificial traffic signs superimposed on road background scenes. Trojan detection is harder on Round2/Round3 in comparison to Round1 due to larger variations in factors used for training Trojaned models such as (1) number of classes—5 in Round1 versus 5 – 25 in Round2/Round3, (2) trigger types—polygon triggers in Round1 versus polygon and Instagram filter based triggers in Round2/Round3, (3) the number of source classes— all classes are poisoned in Round1 versus 1, 2, or all classes in Round2/Round3, and (4) the number of model architectures—3 in Round1 versus 23 in Round2/Round3. Compared to Round2 models in Round3 are adversarially trained using two methods—Projected Gradient Descent and Fast is Better than Free \[^{[62]}\]. The polygon trigger is generated randomly with variations in shape, size, and color. The filter based trigger is generated by randomly choosing from five distinct filters. Trojan detection is much harder on the TrojAI datasets as compared to the Triggered-MNIST dataset due to the use of deeper DNNs and larger variations in appearances of foreground/background objects, trigger patterns, etc. Round1, Round2, and Round3 have 1000, 1104, and 1008 models respectively.

**Metrics:** We report mean and standard deviation of area under the ROC curve (AUC) on 5 randomly selected splits with 80% of the models for training, 10% for validation, and 10% for testing. We provide implementation details for different methods in the Supplementary (Section 1).

B. Quantitative Results

Table I shows the performance of our model DTSE-TD with two temporal encoders along with three SOTA methods on four datasets.

We observe a drop in performance when going from datasets with smaller variations in model architectures and trigger types (Triggered-MNIST, TrojAI-Round1) to those with larger variations (TrojAI-Round2/Round3). For example, DTSE-TD-Conv achieves an AUC of 0.95, 0.89, 0.79, and 0.75 on Triggered-MNIST, TrojAI-Round1, Round2, and Round3 respectively. This drop is expected as Trojan detection becomes more challenging on datasets with more variations in DNNs and trigger patterns. We observe that the simpler CNN based temporal encoder performs similar to Transformer based encoder on simpler datasets (Triggered-MNIST and TrojAI-Round1). However, CNN based encoder shows better performance on harder datasets—TrojAI-Round2/Round3. For example, the AUC on Triggered MNIST is 0.95 for both DTSE-TD-Conv and DTSE-TD-Tx, while it is 0.79 and 0.75 respectively on

\[^2\]https://github.com/trojai/trojai\n
\[^3\]https://pages.nist.gov/trojai/docs/data.html
TrojAI-Round3. We believe that the Transformer likely overfits on datasets with larger variations as compared to CNN encoder due to more parameters. It might be helpful to use more data or smart pre-training strategies as used in BERT [63, 64].

**Comparison with SOTA:** We compare our approach with SOTA methods that diagnose DNNs through sensitivity to universal attacks (Cassandra [15]) and generated noise patterns (ULP [56]), and reverse engineer triggers (Neural Cleanse [13]). We noted earlier that these methods make strong assumptions about the nature of the trigger/attack or the ability of adversarial inputs to activate the shortcut pathway. These assumptions often limit their generalizability to real-world datasets. On Triggered-MNIST, where DNNs belong to three model architectures and are poisoned with only polygon triggers, Cassandra achieves an AUC=0.97 as compared to 0.95 of DTSE-TD-Conv. However, the performance drops drastically compared to DTSE-TD-Conv as we move from Round1 (0.88 versus 0.89) to Round2 (0.59 versus 0.79) or Round3 (0.71 versus 0.75). This happens since the universal adversarial attack used in Cassandra is unable to localize the shortcut pathway in Round2/Round3, where DNNs belong to 23 model architectures and are poisoned with two different trigger types. Our model also outperforms ULP on Round1 (0.89 versus 0.55) and Triggered-MNIST (0.95 versus 0.54). We were not able to train ULP on Round2/Round3 since the training was too slow. We also observe significant improvements over Neural Cleanse, that reconstructs triggers through a constrained optimization, on all datasets e.g. 0.61 of Neural Cleanse versus 0.75 of ours on Round3. We observe that the performance of Neural Cleanse is better on Round2/Round3 compared to Round1. We believe this happens since Neural Cleanse uses anomaly detection on the L1 norm which works better for models with a larger number of classes. We would also like to note that compared to Neural Cleanse and ULP, that required making changes to the attack parameters for each dataset, DTSE-TD uses the same feature extraction for all datasets. This highlights the strong generalization capability of our approach.

**C. Impact of Factors Controlled by Adversary**

We wish to investigate the impact of factors, that can be varied by an adversary for training Trojaned DNNs, on our model’s performance. We focus on four key factors—trigger type, number of source classes, model architecture and poisoning rate. For each factor we create equal-sized subsets whose DNNs assume a fixed value for that factor and then evaluate against DTSE-TD-Conv in Table II.

**Trigger-Type:** We create two subsets from Round2 containing Trojaned models with either polygon or filters based triggers. AUC with polygon and filter based triggers is 0.88 and 0.63 respectively. The performance for polygon triggers is higher since it is easier to identify the ghost neurons for triggers that are well localized in the image. On the other hand, filter based triggers are harder to localize as they are distributed across the entire image. This rationale also explains the drop in performance for all methods between Round1 and Round2, as the later contains DNNs poisoned with filter based triggers.

**Number of source classes:** We create three equal-sized subsets from Round2, each containing Trojaned models with 1, 2 and all classes as source classes (classes whose inputs are misclassified when poisoned). The AUC consistency improves across 1, 2 and all classes (0.96, 0.64, and 0.98 respectively) as the shortcut pathway becomes better visible as more classes are poisoned, making it easier to identify the ghost neurons from the clean data.

**Model architectures:** We create four equal-sized subsets from Round1 containing all benign models and 250 randomly sampled Trojaned models from ResNet, InceptionV3, DenseNet121, and all three architectures. The AUC is 0.92, 0.95, 0.86, and 0.88 for for ResNet50, InceptionV3, DenseNet121, and all architecture respectively. This shows that the performance of the model depends on the model architecture. This is probably because the architectural differences affect the way information is encoded by neurons [65]. For example the performance is higher for models with intra-layer skip-connections such as ResNet and InceptionV3. However, the performance is lower for complex models (DenseNet121) that contains inter-layer skip-connections due the difficulty in localizing the ghost neurons. Also, it is harder to generalize across multiple models resulting in the lowest performance for “all” architecture.

**Poisoning Rate:** We create two equal-sized subsets from Round2 with poisoning rate (% of poisoned data used during training) $> = 0.29$ and $< 0.29$ respectively. The AUC is higher for poisoning rate $\geq 0.29$ (0.89 vs 0.71) since the shortcut pathway is more visible in this case.

Fig. 5 shows attribution heatmaps that reveal the contribution of image parts for activating one of the ghost neurons identified in Fig. 3. We observe that the high attribution parts

| Model               | Triggered-MNIST   | TrojAI-Round1 | TrojAI-Round2 | TrojAI-Round3 |
|---------------------|-------------------|---------------|---------------|---------------|
| Cassandra [15]      | 0.97 ± 0.010      | 0.88 ± 0.006  | 0.59 ± 0.096  | 0.71 ± 0.026  |
| Neural Cleanse [13] | 0.70 ± 0.045      | 0.50 ± 0.030  | 0.63 ± 0.043  | 0.61 ± 0.064  |
| ULP [56]            | 0.54 ± 0.051      | 0.55 ± 0.058  | —             | —             |
| DTSE-TD-Conv        | 0.95 ± 0.028      | 0.89 ± 0.029  | 0.79 ± 0.034  | 0.75 ± 0.035  |
| DTSE-TD-Tx          | 0.95 ± 0.022      | 0.89 ± 0.029  | 0.75 ± 0.033  | 0.72 ± 0.038  |

**TABLE I:** Comparison of our model (DTSE-TD) with three SOTA method on four datasets containing Trojaned DNNs. “Conv” and “Tx” refers to CNN and Transformer based temporal encoder. Performance is reported using mean AUC with standard deviation across 5 random splits.
TABLE II: Impact of factors, that can be varied by an adversary for training Trojaned DNNs, on our model’s performance.

| Model arch | AUC       | No. of Source Classes | AUC       |
|------------|-----------|-----------------------|-----------|
| ResNet50   | 0.92 ± 0.040 | 1                     | 0.56 ± 0.119 |
| InceptionV3| 0.95 ± 0.033 | 2                     | 0.64 ± 0.033 |
| DenseNet121| 0.86 ± 0.064 | all                   | 0.98 ± 0.016 |

| Poisoning Rate | AUC       |
|----------------|-----------|
| ≥ 0.29         | 0.89 ± 0.041 |
| < 0.29         | 0.71 ± 0.057 |

Fig. 5: Heatmaps reveal the poly-semantic nature of a ghost neuron that is responsible for the trigger behavior and fires for patterns from both the actual class(es) and the trigger.

V. CONCLUSION

We proposed an approach based on explainability of DNNs for detecting whether a DNN is Trojaned given only model weights and a few clean test samples. We introduced a novel intuition that the trigger behavior is localized in a few ghost neurons that are poly-semantic in nature and fire for both input classes and trigger pattern. We use feature attribution to identify such neurons based on their relevance for switching decision to a counter-class. We referred to such attributions as counterfactual attributions since they explain output for a counter-class which is not the actual predicted class. We argued that the ghost neurons in a Trojaned model will have high attributions towards the target class even for clean inputs. We then incrementally excited these neurons and monitor the drop in model’s accuracy. Based on a proven theoretical result about feature attribution in Trojaned DNNs, we claimed that the accuracy will drop sharply for Trojaned models on the excitation of ghost neurons. We use a deep temporal set encoder to input these class-wise performance curves and train a Trojan Detection network. We evaluated our approach on a set of challenging benchmarks with a large diversity in model architectures, number of classes, trigger pattern etc. Our results show that our method is able to consistently improve upon model diagnosis and trigger reverse engineering based SOTA methods. In future work, we plan to better understand the effect of different training strategies and model architectures on the counterfactual attributions for Trojan detection.

VII. ACKNOWLEDGEMENTS

This work was supported in part by U.S. Army Research Laboratory Cooperative Research Agreement W911NF-17-2-0196, and the Intelligence Advanced Research Projects Agency (IARPA) under the contract W911NF20C0038. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the US Department of Defense or the Government, and no official endorsement should be inferred.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Neural information processing systems, pp. 1097–1105, 2012.
[2] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” arXiv preprint arXiv:1409.0473, 2014.
[3] A. Graves, A.-r. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” in International conference on acoustics, speech and signal processing, pp. 6645–6649, 2013.
[4] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” arXiv preprint arXiv:1312.6199, 2013.
[5] R. Shokri and V. Shmatikov, “Privacy-preserving deep learning,” in Proceedings of the 22nd ACM SIGSAC
conference on computer and communications security, pp. 1310–1321, 2015.

[6] Y. Li, B. Wu, Y. Jiang, Z. Li, and S.-T. Xia, “Backdoor learning: A survey,” arXiv preprint arXiv:2007.08745, 2020.

[7] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, and A. Kurakin, “On evaluating adversarial robustness,” arXiv preprint arXiv:1902.06705, 2019.

[8] H. X. Y. M. Hao-Chen, L. D. Deb, H. L. J.-L. T. Anil, and K. Jain, “Adversarial attacks and defenses in images, graphs and text: A review,” International journal of automation and computing, vol. 17, no. 2, pp. 151–178, 2020.

[9] T. Gu, B. Dolan-Gavitt, and S. Garg, “Badnets: Identifying vulnerabilities in the machine learning model supply chain,” arXiv preprint arXiv:1708.06733, 2017.

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

[11] S. Goldwasser, M. P. Kim, V. Vaikuntanathan, and O. Zamir, “Planting undetectable backdoors in machine learning models,” in 2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS), pp. 931–942, IEEE, 2022.

[12] A. Turner, D. Tsipras, and A. Madry, “Label-consistent backdoor attacks,” arXiv preprint arXiv:1912.02771, 2019.

[13] 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.

[14] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, “Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks,” in International joint conferences on artificial intelligence, pp. 4658–4664, 2019.

[15] X. Zhang, A. Mian, R. Gupta, N. Rahnavard, and M. Shah, “Cassandra: Detecting trojaned networks from adversarial perturbations,” arXiv preprint arXiv:2007.14433, 2020.

[16] R. Wang, G. Zhang, S. Liu, P.-Y. Chen, J. Xiong, and M. Wang, “Practical detection of trojan neural networks: Data-limited and data-free cases,” arXiv preprint arXiv:2007.15802, 2020.

[17] M. Sundararajan, A. Taly, and Q. Yan, “Axiomatic attribution for deep networks,” arXiv preprint arXiv:1703.01365, 2017.

[18] S. Jha, S. Raj, S. Fernandes, S. K. Jha, S. Jha, B. Jalaian, G. Verma, and A. Swami, “Attribution-based confidence metric for deep neural networks,” in Neural information processing systems, pp. 11826–11837, 2019.

[19] M. Ancona, E. Ceolini, C. Öztireli, and M. Gross, “Towards better understanding of gradient-based attribution methods for deep neural networks,” arXiv preprint arXiv:1711.06104, 2017.

[20] L. A. Hendricks, R. Hu, T. Darrell, and Z. Akata, “Generating counterfactual explanations with natural language,” arXiv preprint arXiv:1806.09809, 2018.

[21] P. Wang and N. Vasconcelos, “Scout: Self-aware discriminant counterfactual explanations,” in Conference on computer vision and pattern recognition, pp. 8981–8990, 2020.

[22] Y. Goyal, Z. Wu, J. Ernst, D. Batra, D. Parikh, and S. Lee, “Counterfactual visual explanations,” arXiv preprint arXiv:1904.07451, 2019.

[23] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” in Neural information processing systems, pp. 4765–4774, 2017.

[24] G. Li and Y. Yu, “Visual saliency based on multiscale deep features,” in CVPR, pp. 5455–5463, 2015.

[25] K. M. Yi, E. Trulls, V. Lepetit, and P. Fu, “Lift: Learned invariant feature transform,” in European conference on computer vision, pp. 467–483, Springer, 2016.

[26] S. Jha, V. Raman, A. Pinto, T. Sahai, and M. Francis, “On learning sparse Boolean formulae for explaining AI decisions,” in NASA Formal methods symposium, pp. 99–114, Springer, 2017.

[27] S. Jha, T. Sahai, V. Raman, A. Pinto, and M. Francis, “Explaining AI decisions using efficient methods for learning sparse Boolean formulae,” Journal of automated reasoning, vol. 63, no. 4, pp. 1055–1075, 2019.

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

[29] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” in Conference on computer vision and pattern recognition, pp. 618–626, 2017.

[30] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, “Sanity checks for saliency maps,” in NIPS, pp. 9525–9536, 2018.

[31] P. Chalasani, S. Jha, A. Sadagopan, and X. Wu, “Adversarial learning and explainability in structured datasets,” arXiv preprint arXiv:1810.06583, 2018.

[32] N. Kilbertus, G. Parascandolo, and B. Schölkopf, “Generalization in anti-causal learning,” arXiv preprint arXiv:1812.00524, 2018.

[33] S. Jha, S. Raj, S. Fernandes, S. K. Jha, S. Jha, J. Brian, G. Verma, and A. Swami, “Attribution-driven analysis for detection of adversarial examples,” Safe Machine Learning workshop at ICLR, 2019.

[34] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. 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, pp. 1265–1282, 2019.

[35] A. Kurakin, I. Goodfellow, S. Bengio, Y. Dong, F. Liao, M. Liang, T. Pang, J. Zhu, X. Hu, C. Xie, et al., “Adver-
sarial attacks and defences competition,” in *The NIPS’17 competition: building intelligent systems*, pp. 195–231, Springer, 2018.

[36] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, “Universal adversarial perturbations,” in *Conference on computer vision and pattern recognition*, pp. 1765–1773, 2017.

[37] K. R. Mopuri, A. Ganeshan, and R. V. Babu, “Generalizable data-free objective for crafting universal adversarial perturbations,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 10, pp. 2452–2465, 2018.

[38] S. Thys, W. Van Ranst, and T. Goedemé, “Fooling automated surveillance cameras: adversarial patches to attack person detection,” in *Conference on computer vision and pattern recognition*, pp. 0–0, 2019.

[39] M. Edraki, N. Karim, N. Rahnavard, A. Mian, and Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and Y. Liu, Y. Xie, and A. Srivastava, “Neural trojans,” in *Conference on computer vision and pattern recognition*, pp. 113–125, 2019.

[40] S. Kolouri, A. Saha, H. Pirsiavash, and H. Hoffmann, “Universal litmus patterns: Revealing backdoor attacks in cnns,” in *Conference on computer vision and pattern recognition*, pp. 301–310, 2020.

[41] Y. Liu, Y. Xie, and A. Srivastava, “Neural trojans,” in *International conference on computer design*, pp. 45–48, 2017.

[42] M. Villarreal-Vasquez and B. Bhargava, “Confoc: Content-focus protection against trojan attacks on neural networks,” *arXiv preprint arXiv:2007.00711*, 2020.

[43] Y. Li, T. Zhai, B. Wu, Y. Jiang, Z. Li, and S. Xia, “Re-thinking the trigger of backdoor attack,” *arXiv preprint arXiv:2004.04692*, 2020.

[44] 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 *Computer security applications conference*, pp. 113–125, 2019.