# "We must protect the Transformers": Understanding Efficacy of Backdoor Attack Mitigation on Transformer Models

Rohit Raj**[0009-0006-0394-0446], Biplab Roy**[0009-0008-5334-0056], Abir Das[0000-0002-2327-1618], and Mainack Mondal[0000-0003-4317-0184]

Department of Computer Science and Engineering, IIT Kharagpur, India
`rrohit2901@gmail.com`,`biplabroy@kgpian.iitkgp.ac.in`
`abir@cse.iitkgp.ac.in`,`mainack@cse.iitkgp.ac.in`

**Abstract.** Recently, Neural Network based Deep Learning (DL) backdoor attacks have prompted the development of mitigation mechanisms for such attacks. Out of them a key mitigation mechanism is *Neural Cleanse*, which helps in the identification and mitigation of DL backdoor attacks. It identifies the presence of backdoors in Neural Networks and constructs a reverse-engineered trigger, which is later used to mitigate the backdoor present in the infected model. However, since the publication of Neural Cleanse, newer DL architectures (e.g., Transformer models) have emerged and are widely used. Unfortunately, it is not clear if Neural Cleanse is effective to mitigate backdoor attacks in these newer models—in fact a negative answer will prompt researchers to rethink backdoor attack mitigation. To that end, in this work, we take the first step to explore this question. We considered models ranging from pure convolution-based models like ResNet-18 to pure Self-Attention based models like ConVit and understand the efficacy of Neural Cleanse after launching backdoor attacks on these models. Our experiments uncover a wide variation in the efficacy of Neural Cleanse. Even if Neural Cleanse effectively counters backdoor attacks in some models, its performance falls short when dealing with models incorporating self-attention layers (i.e., Transformers), especially in accurately identifying target classes and learning reverse-engineered triggers. Our results further hint that, for modern models, mitigation of backdoor attacks by constructing reverse engineering triggers should consider *patches* (instead of pixels).

**Keywords:** Backdoor Attack · Neural Cleanse · convolution-based models · Self-Attention based models.

## 1 Introduction

With the development of the computational capabilities of modern computers, Artificial Intelligence has acquired a spot as an integral part of our daily lives. In fact, Deep Neural Networks (DNNs) have become the core of many critical tasks

** Both authors contrbuted equally to the project.

like facial recognition, guiding self-driving cars and creating voice interfaces for home assistants in our day-to-day lives. Deep learning has also been applied to security space for malware [13] and network intrusion detection [14] tasks. Further advancements in the field of Deep learning (e.g., designing new architectures) are continuously improving the performance of different tasks every day. However, still, Neural networks are considered black-boxes in most context because studying the structure of those models give no insights about the structure of the function being approximated by that model [15, 16].

In spite of this lack of explainability of output, it is impossible to test deep learning models exhaustively for all possible input values due to the versatility of the application scenarios (e.g., object recognition). Thus even if some models work fine on one input then they might work incorrectly on other input. In fact, the situation might worsen if an attacker can control the inaccuracy of the model output and misdirect an end user. This poses a great challenge of how secure these models are for application on critical real work applications. Thus a large amount of work in the community focused on how deep learning systems are very vulnerable to attacks [12]—e.g., how adding perturbations to inputs of AI systems used in self-driving cars can sometimes force them to make wrong decisions for a particular input. These vulnerabilities enable the possibility of *backdoors* or "Trojans" in DNNs. Backdoors are hidden patterns in the data that have been trained into a DNN model (e.g., by perturbing training data) that result in unexpected behaviour but are undetectable unless activated by some "trigger" input. A "trigger" refers to a small, carefully designed and imperceptible pattern or modification that is added to an input image with the intention of causing a misclassification or some other unintended behaviour when the model makes predictions.

Given the severity of the problem of backdoor attacks, many countermeasures have been developed to identify and mitigate the presence of backdoors in DNNs, but in past works, we found that these countermeasures were tested only on smaller (and simpler) DNN and purely Convolutional Neural Network (CNN) based models [1]. However, with the introduction of transformers [18], many computer vision models, now, work with self-attention layers instead of convolution layers only [19, 20]. This created a large gap between models on which countermeasures against backdoor attacks were being tested and DNNs that were being actually deployed in real-world scenarios. In this work, we take a step to bridge this gap.

Specifically, in this work, we consider Neural Cleanse [2] (described in Section 3). Neural Cleanse is a popular and representative backdoor attack mitigation algorithm, aimed towards identifying and mitigating backdoor attacks in DNNs. Neural cleanse provides a mechanism to detect backdoor attacks and then provide heuristics (based on *unlearning*) to update the DNN and undo the effect of a trigger. In previous work, Neural Cleanse has been tested on CNN models like VGG-16 and Resnet-101 [1], but there is no work on experimenting with the robustness of Neural Cleanse on self-attention-based networks or a hybrid of the two. These newer models are the state of the art models and created revolution in terms of inference accuracy in practical tasks. Thus, pertaining to the huge popularity of these models, it is necessary to understand if these models can be protected against backdoor attacks, e.g., via Neural Cleanse—we focus on DNN models used in computer visions or vision models. To that end, we ask the following questions in this work in progress:

1. Can backdoor attacks be successfully launched against newer self-attention based or hybrid architectures?
2. Does mitigation strategy like Neural Cleanse works on self-attention-based and hybrid vision models? Why or why not?

We address these two questions using extensive experimentation. We re-implemented Neural Cleanse into pytorch and launched the attacks on a number of models ranging from pure convolution-based models like ResNet-18 to pure Self-Attention based models like ConVit, trained on two popular datasets, wiz. GTSRB (German Traffic Signal Recognition Benchmark) [21] and CIFAR-10 [1]. Our results show that backdoor attacks indeed work on both older CNN models as well as newer self-attention based or hybrid architectures. However, Neural Cleanse is significantly less effective on mitigating backdoor attack on newer models. Our analysis further reveal potential reasons behind this discrepancy and identify a path forward. Next, we will start with describing related works for our study.

## 2 Related Work

**Attacks on deep learning models:** A large amount of research has been conducted on different types of attacks on Machine Learning models. These attacks can be broadly classified into three categories Integrity attacks, Availability attacks, and Privacy attacks. Backdoor attacks are a type of integrity attack in which training data is poisoned with triggers and changing their label to a particular target class. There is a vast literature on how backdoor attacks can be conducted on DNNs. Some different types of backdoor attacks on DNNs are Outsourcing attacks, and Pretrained attacks [3]. Outsourcing attacks are older forms of backdoor attack, where the attacker has access to the training of models [29,30]. The model can efficiently learn the attacker-chosen backdoor sub-task and its main task at the same. Many variants of outsourcing attacks have been identified by the community in the past. Other examples of outsourcing attacks include dynamic trigger attacks and backdoor reinforcement learning attacks [3]. Pretrained attack is usually mounted via a transfer learning scenario, where the user is limited with few data or/and computational resources to train an accurate model. Therefore, the user will use a public or third-party pretrained model to extract general features. Examples of such backdoors are Trojan attacks and badnets [32, 33].

[1] https://www.cs.toronto.edu/ kriz/cifar.html

**Mitigation of backdoor attack on deep learning models:** Countermeasures against backdoors can be largely classified into four categories wiz Blind Backdoor removal, Offline Data Inspection, Offline Model Inspection and Online Input Inspection. Blind backdoor removal methods can differentiated from other methods based on fact that it does not differentiate backdoored model from a clean model, or clean input from input with trigger. Some methods falling under this class of countermeasures are Fine Pruning [7], Suppression [9], Februus [8], ConFoc [10] and RAB [11]. These methods prove to be effective for backdoored models but when applied on clean models, performance of models tend to decrease. Offline Data Inspection works on strong assumption that defenders have access to the poisoned data. A few methods falling under this class include Spectral Signature, Gradient Clustering, Activation Clustering, Deep k-NN, SCAn and differential privacy [35, 36]. These countermeasures are mostly based on clustering algorithms and fail to work effectively in case of special scaling-based backdoor attacks. Offline Model inspections tend to avoid assumptions made by data inspection methods; hence these are more suitable for countering attacks resulting from various attacking surfaces. Methods falling under these classes are Trigger Reverse Engineer, NeuronInspect, DeepInspect, AEGIS, Meta Classifier and Neural Cleanse [2]. These methods generally require high computational overhead and can't deal with large triggers, especially those aiming to reverse engineer the trigger. Online inspection methods can also be applied to monitor the behaviour of either the model or input during run-time.

**Research Gap:** We found very less amount of literature on the evaluation of the robustness of the Neural Cleanse algorithm [2], especially on modern-day models like ConVit, ViT and DeiT. Work which most closely resembles our work is [1], in which authors compared the performance of various countermeasures mainly on convolution-based models like Resnet-18 and VGG19. To the best of our knowledge, we could not find any work analysing whether self-attention-based computer vision models can be backdoored. In this work, we tried to perform backdoor attacks on self-attention-based and hybrid models, followed by testing the performance of Neural cleanse on such models. Model architectures considered in our study include Resnet-18, ConVit, ViT, DeiT and compact transformer. Our exploration identify that Neural Cleanse does not work well on self-attention based models and hint at a potential reason. For these newer architectures, a mitigation strategy should consider *patches* instead of neurons. Next section describes the methodology of our exploration.

## 3 Methodology

In this work, we examined the potential of backdoor attack and efficacy of the defense (Neural Cleanse). To that end, we experimented with a variety of models on the GTSRB (German Traffic Signal Recognition) and CIFAR-10 datasets. The models considered in the experimentation phase can be broadly classified into three categories completely convolution-based visual models, completely Self-Attention based visual models and hybrid Models. We start first with a description of these models and then provide an overview of Neural Cleanse along with the attack model, that we followed in this work.

## 3.1 Computer Vision Models - Preliminaries

**Convolution Neural Network:** A Convolutional neural network (CNN) is a neural network that has one or more convolutional layers. In our experiments, we primarily focused on a 6-layer CNN network and Resnet-18 to carry out our experiments. ResNet-18 is a convolutional neural network that is 18 layers deep. Skip connections or shortcuts are used to jump over some layers.

**Self-Attention Based Models:** The increasing popularity of transformers in NLP has led to the development of visual models based only on the attention mechanism, completely removing convolution layers. In these Self-Attention based models at first, input images are divided into nonoverlapping patches before embedding them into a multidimensional space. In the context of attention-based models like the Vision Transformer (ViT) [19] and Data-efficient Image Transformer (DeiT) [20], the image patches are also referred to as "tokens". These patches are then treated as individual elements, allowing the model to process them separately and perform attention mechanisms over them.

ViT stands for Vision Transformer. The standard Transformer receives as input a 1D sequence of token embeddings. To handle 2D images, image $x \in R^{H \times W \times C}$ is reshaped into a sequence of flattened 2D patches $x_p \in R^{N \times (P^2 C)}$, where $(H, W)$ is the resolution of the original image, $C$ is the number of channels, $(P, P)$ is the resolution of each image patch, and $N = HW/P^2$ is the resulting number of patches, which also serves as the effective input sequence length for the Transformer. The output of this patch projection is referred to as embeddings.

DeiT stands for Data-efficient Image Transformer. A Data-Efficient Image Transformer is a type of Vision Transformer for image classification tasks. The model is trained using a teacher-student strategy specific to transformers. It relies on a distillation token, ensuring that the student learns from the teacher through attention. The architecture is similar to that of ViT, but it can be trained in much less time than compared to ViT.

**Hybrid models:** While pure self-attention-based models provide high accuracy, their data-hungry nature while training puts a bottleneck to their usability. To eliminate such constraints, hybrid models are developed, which have convolution and self-attention layers. In past studies, it has been identified that the first few convolution layers, followed by self-attention layers, enhance the performance of visual models on classification tasks and also reduce the amount of data required to train the model. One such hybrid model is the Compact transformer [38]. Initially, features in the image are identified using convolution layers which are later processed using a transformer encoder. Another hybrid model considered in our study is ConVit [39], which stands for Convolution-like Vision Transformer.

### 3.2 Setting up Neural Cleanse

**Attack Model:** Attack models considered by the algorithm are BadNets [40] and Trojan Attack [41]. BadNets is a backdoor attack methodology in which the adversary has access to the training data, and the same trigger is added to the input data points irrespective of the input. In contrast, in Trojan attacks, the trigger is engineered based on the infected model. Both attack models poison the model during its training phase. Neural cleanse assume that the defender has access to trained DNNs, a set of clean samples to test the performance of the model, and access to computational resources to test or modify DNNs.

**Backdoor detection phase:** The fundamental premise of backdoor detection is that, compared to other uninfected labels, the target label might be incorrectly classified as an infected model with considerably smaller adjustments. Therefore, we repeatedly go through all of the model's labels to see whether any may be misclassified with a lower amount of alteration. The three stages below make up our whole method.

- **Step 1:** For a given label, it treats it as a potential target label of a targeted backdoor attack. An optimization scheme is designed to find the "minimal" trigger required to misclassify all samples from other labels into this target label. In the vision domain, this trigger defines the smallest collection of pixels and its associated colour intensities to cause misclassification.
- **Step 2:** Step 1 is repeated for each output label in the model. For a model with $N = |L|$ labels, this produces $N$ potential "triggers".
- **Step 3:** After coming up with $N$ potential triggers, the size of each trigger is measured by the number of pixels each trigger candidate has, i.e. how many pixels the trigger is replacing. Finally, it runs an outlier detection algorithm to detect if any trigger candidate is significantly smaller than other candidates.

**Mitigation of Backdoor attack phase:** After the detection of a backdoor in the model, multiple approaches are proposed to mitigate the backdoor. The first approach involves filtering inputs with a trigger by analysing neural activations. The second approach involves updating DNN via neuron pruning, i.e. removing those neurons that produce strong activations in the presence of triggers. We primarily focus on a third approach which involves updating DNN via unlearning.

Updating DNN via unlearning involves retraining the poisoned model with reversed engineered trigger assisting in unlearning the backdoor present in the model. The methodology involves adding triggers to randomly picked images but keeping their labels intact and then training the model on the modified dataset. For this methodology, two variants are considered, one in which retraining is done on a dataset prepared with reverse engineered trigger and another one in which the dataset is prepared with the original trigger.

**Our implementation of Neural Cleanse:** We used the implementation of the Neural cleanse provided in the actual paper [2] and re-implemented it in the Pytorch framework. To ease the understandability of the code, we created a module for the injection of the model (one illustration of trigger injection is shown in Figure 1), which first trains the model on the clean dataset and then performs *pretrained backdoor attack* by finetuning the model with poisoned input. This is followed by the module for the detection of a backdoor in the model, which also constructs the reverse-engineered trigger.

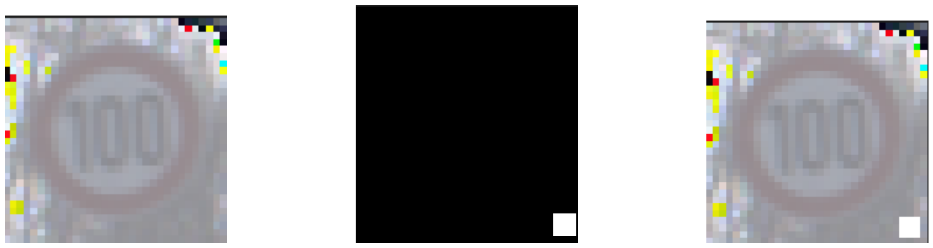

**Fig. 1.** Image on left shows the original image from the dataset, the image in the middle shows the trigger added to the image, and rightmost image shows poisoned input image

After constructing the reverse-engineered trigger, we updated the DNN by unlearning the model, which uses both the original trigger and reverse-engineered trigger. Both of the triggers are used in the process because it provides us with a heuristic to measure the quality of reverse engineered trigger.

## 4 Efficacy of Backdoor Attack and Attack Mitigation on Newer Models

We experimented with Neural Cleanse on different types of models. To test the performance of neural cleanse in the identification of a backdoor in the model, we logged the results in Table 2 and Table 4. Finally, to check the quality of reverse engineering, we applied the neural updating method by unlearning introduced in the paper. The results have been presented in Table 3 and 5.

### 4.1 Backdoor attack success on newer architectures

In our experiments, we found that the backdoor attack injection was successful. This can be inferred from Table 3 where the drop of accuracy on clean samples was very low, but the accuracy for inputs with trigger was very high. This trend is consistent across all models, with slightly low accuracy for compact transformers.

[2] https://github.com/bolunwang/backdoor

| **Models** | **GTSRB** | | **CIFAR-10** | |
|---|---|---|---|---|
| | **AI** | **TargetNorm** | **AI** | **TargetNorm** |
| **6 layer CNN** | 2.89 | 0.51 | 2.17 | 0.67 |
| **Resnet - 18** | 2.18 | 0.78 | 3.09 | 0.73 |
| **DeiT** | Classified wrong | 0.73/0.79 | 1.31 | 0.67 |
| **ViT** | Classified wrong | 0.81/0.819 | 1.71 | 0.85 |
| **Convit** | 2.21 | 0.68 | 2.21 | 0.73 |
| **Compact Transformer** | 2.45 | 0.76 | 2.75 | 0.71 |

**Table 1.** AI (Anomaly Index) and norm of mask for target class found in case of different models

The confidence of Neural Cleanse about the presence of a backdoor in a model is conveyed by the value of AI (Anomaly Index), which represents the amount by which minimum mask size varies from the median value. The AI obtained in experiments has been presented in Table 1. Neural Cleanse detected wrong class in self attention based models like DeiT and ViT.

## 4.2 Efficacy of Neural Cleanse for identifying backdoors

In our experiments, we found that neural cleanse worked well in the case of pure convolution-based models like CNN and Resnet-18.

| **Models** | **Accuracy 1** | **Accuracy 2** | **Accuracy 3** |
|---|---|---|---|
| **6 layer CNN** | 0.927 | 0.964 | 0.983 |
| **Resnet - 18** | 0.97 | 0.981 | 0.991 |
| **VIT (Finetuned)** | 0.973 | 0.979 | 0.993 |
| **DeiT (Finetuned)** | 0.977 | 0.983 | 0.987 |
| **Compact Transformer** | 0.961 | 0.981 | **0.965** |
| **Convit (Finetuned)** | 0.961 | 0.98 | 0.991 |

**Table 2.** Table showing performance of neural cleanse for backdoor identification; Accuracy of different models on GTSRB dataset, in different scenarios; Accuracy1 - Accuracy of infected models on clean samples; Accuracy2 - Accuracy of the clean model on clean samples; Accuracy3 - Accuracy of the infected model on poisoned samples

| **Models** | **Accuracy 1** | **Accuracy 2** | **Accuracy 3** |
|---|---|---|---|
| **6 layer CNN** | 0.934 | 0.965 | 0.987 |
| **Resnet - 18** | 0.974 | 0.984 | 0.991 |
| **VIT** | 0.973 | 0.984 | 0.983 |
| **DeiT** | 0.977 | 0.987 | 0.984 |
| **Compact Transformer** | 0.972 | 0.99 | 0.975 |
| **Convit (Finetuned)** | 0.961 | 0.98 | 0.991 |

**Table 3.** Table showing performance of neural cleanse for backdoor identification; Accuracy of different models on CIFAR-10 dataset, in different scenarios; Accuracy1 - Accuracy of infected models on clean samples; Accuracy2 - Accuracy of the clean model on clean samples; Accuracy3 - Accuracy of the infected model on poisoned samples

Table 2 and 3 show the quality of backdoor injection in models. We can see that there is a slight drop in the accuracy of models when tested on clean samples, but for poisoned samples, a significant proportion of inputs are classified as the target class. This tends to be in line with results presented in past literature. The accuracy of the infected model on poisoned inputs is slightly less, showing the attack's smaller success in the GTSRB dataset.

One interesting observation in our experiment was that on GTSRB dataset, neural cleanse failed to identify the correct target class for both self-attention models. This was due to a minor size difference obtained between the norm of the mask with the target class and some other classes. Also, in the case of using CIFAR-10 dataset, these two models have low Anomaly Index (AI).

### 4.3 Quality of reverse engineered trigger created by Neural Cleanse

The reversed engineered trigger has been shown in Figure 2. The trigger for ResNet-18 is more visually similar to the original trigger than the trigger for ViT. To test further the quality of reverse-engineered triggers, we used neural updating via the unlearning method, which mitigates the backdoor in the model by retraining the poisoned model with reverse-engineered and original triggers.

| **Models** | **Accuracy 1** | **Accuracy 2** | **Accuracy 3** |
|---|---|---|---|
| **6 layer CNN** | 0.935 | 0.89 | 0.964 |
| **Resnet - 18** | 0.946 | 0.94 | 0.981 |
| **Deit** | **0.853** | 0.94 | 0.987 |
| **ViT** | **0.871** | 0.96 | 0.993 |
| **Convit** | 0.913 | 0.972 | 0.991 |
| **Compact Tranformer** | **0.74** | 0.95 | 0.981 |

**Table 4.** Table showing performance of neural cleanse for backdoor identification; Accuracy of different models on GTSRB dataset, in different scenarios; Accuracy1 - Accuracy of model cleaned using reverse engineered trigger; Accuracy2 - Accuracy of model cleaned using original trigger; Accuracy3 - clean model on clean samples

| **Models** | **Accuracy 1** | **Accuracy 2** | **Accuracy 3** |
|---|---|---|---|
| **6 layer CNN** | 0.947 | 0.941 | 0.965 |
| **Resnet - 18** | 0.951 | 0.956 | 0.984 |
| **DeiT** | **0.893** | 0.961 | 0.984 |
| **ViT** | **0.875** | 0.954 | 0.987 |
| **Convit** | 0.923 | 0.948 | 0.990 |
| **Compact Transformer** | **0.871** | 0.962 | 0.980 |

**Table 5.** Table showing performance of neural cleanse for backdoor identification; Accuracy of different models on CIFAR-10 dataset, in different scenarios; Accuracy1 - Accuracy of model cleaned using reverse engineered trigger; Accuracy2 - Accuracy of model cleaned using original trigger; Accuracy3 - clean model on clean samples

Table 5 and table 4 show the accuracy of models after unlearning using the original trigger and reverse engineered trigger. The accuracy of the clean

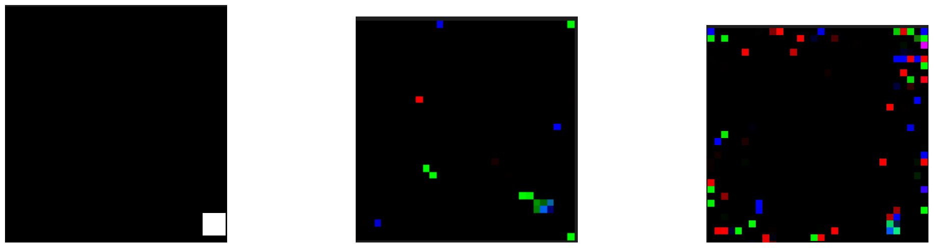

**Fig. 2.** Image on the left shows the original trigger used to infect the model; Image in centre shows reversed engineer for ResNet-18; Rightmost image shows reversed engineered trigger for ViT

model on clean sample is as expected. While performing the neural updating by unlearning, we observed that accuracy after retraining with reverse engineered trigger was significantly low compared to retraining with the original trigger in the case of Pure self attention-based models showing lower quality reverse engineered trigger. The image generated by Neural Cleanse showed a similar drop in the case of the compact transformer, but the drop in the case of ConVit was not that significant. This behaviour was consistent across both datasets, showing consistency of the pattern observed. From these observations, we hypothesized that the performance of Neural cleanse declines with an increase in the self-attention behavior of the model on which it is being employed.

Finally, we investigate the reason behing these observations—why Neural Cleanse can not devise good quality reverse engineered triggers for pure self-attention networks.

## 5 Understanding Quality of Mitigation by Neural Cleanse

We first aim to open the black box—specifically we aim to understand the impact of the reverse engineered triggers offered by Neural Cleanse for both older CNN models as well as newer transformer model. We focus on four dimensions—amount of neurons activated by reverse trigger, type of neurons activated by reverse trigger, pace of learning across models and impact of trigger on discrete patches in images (rather than pixels).

### 5.1 Comparing Fraction of Activated Neurons Across Models

Network Dissection [37] from Bau et al. is a method for quantifying the interpretability of deep visual representations learned by neural networks. This work argued that while neural networks have achieved impressive performance on a variety of visual recognition tasks, the representations learned by these networks are often difficult to interpret or understand. This lack of interpretability can limit the usefulness of neural networks in real-world applications where transparency and explainability are important.

To address this problem, Bau et al. proposed a method to interpret the visual representations learned by neural networks. The method involves using semantic segmentation to identify objects in images and then measuring the activation of individual units in the neural network in response to those objects. They introduce a new metric called the "dissected unit", which measures the activation of a unit in response to a specific object. This metric can be used to quantify the degree to which a unit in the neural network is interpretable, i.e. how well it corresponds to a semantically meaningful visual concept.

The paper demonstrates the usefulness of the proposed method by applying it to several state-of-the-art neural networks trained on object recognition and scene classification tasks. The authors show that the method can be used to compare different layers in a network and to identify which layers contain more interpretable visual features. They also demonstrate that networks with higher interpretability tend to perform better on recognition tasks.

We built on network dissection to fit our purpose of studying the fraction of neurons which get activated in presence of a trigger. Since, we are using different types of models so defining neurons which we consider during this experiment becomes important for the comparison. Therefore, to maintain consistency in the comparisons, we considered outputs of all Linear and Conv2D layer neurons.
**Steps of the algorithm:** The existing implementations of Network Dissection [37] are not able to handle self-attention based vision transformers and thus we step into modifying the algorithm as follows:

**1. Attach forward hooks to each neuron of each convolutional layer and linear layer of the model:** This step involves adding a forward hook to each neuron in every convolutional and linear layer of the neural network model. A forward hook is a function that is executed every time the output of a neuron is computed during a forward pass through the network. By attaching forward hooks to each neuron, we can log the activation of each neuron during the forward pass.

**2. Log activation of all hooked neuron activations in absence of trigger:** After attaching the forward hooks, we log the activation of each hooked neuron in the absence of any trigger. This means that we simply run the input data through the network and record the activation of each neuron as it is computed.

**3. Log activation of all hooked neuron activations in presence of trigger.** Next, we log the activation of each hooked neuron in the presence of a trigger. The trigger selectively activates certain neurons and this step enables us to observe how they respond to the input contaminated with the trigger.

**4. Final result is percentage of neurons whose activation has increased more than a threshold:** We compare the activation of each neuron in each model in the absence and presence of the trigger and calculate the percentage of neurons whose activation has increased more than a certain threshold. This threshold is typically set to a small percentage of the maximum possible activation value, such as 1% or 5%.

| Presence of poison in model | Type of Trigger | 6-layer CNN | Resnet | Compact Transformer | ConVit | ViT | DeiT |
|---|---|---|---|---|---|---|---|
| Poisoned Model | Reverse Eng | 27.56 | 25.56 | 23.52 | 27.45 | 26.84 | 24.65 |
| Poisoned Model | Original | 31.25 | 30.14 | 27.56 | 29.56 | 34.25 | 30.47 |
| Clean Model | Reverse Eng | 31.25 | 30.14 | 27.56 | 29.56 | 34.25 | 30.47 |
| Clean Model | Original | 34.25 | 32.62 | 31.52 | 30.52 | 35.62 | 32.52 |

**Table 6.** Percentage of neurons activated due to the presence of original and reverse engineered trigger in the input.

Neurons that meet this criterion are considered to be "triggered" by the trigger pattern. By measuring the percentage of triggered neurons, we can estimate the susceptibility of the network to adversarial attacks. The percentage of neurons activated in each model have been summarized in 6. These results showed no significant difference in these models.

## 5.2 Comparing Importance of Activated Neurons Across Models

With no discerning pattern in the above experiment, we went ahead to see if the same set of neurons as in the previous experiment are getting activated across the models in the presence of *reverse engineered* trigger as well.
**Neuron Importance:** In this case, following Neural Cleanse, we defined neuron importance. Neuron importance refers to ranking of neurons based on activation due to presence of a trigger in the input to the poisoned model. The higher the activation, lower the rank and hence, the neuron has higher importance.
**Steps in the algorithm:** The steps involved in the experiment include:

**1. Attach forward hooks to each neuron of each layer:** Similar to the previous experiment, this step involves adding a forward hook to each neuron in every layer of the neural network model.

**2. Rank neurons according to activation due to presence of original trigger:** After attaching the forward hooks, we first inject original trigger pattern to the input data and record the activation of each neuron in the presence of the trigger. We then rank the neurons according to their activation levels in response to the trigger. Neurons that exhibit high activation levels in response to the trigger are considered to be important for the network's response to the input.

**3. Rank neurons according to activation due to presence of reverse engineered trigger:** Next, we inject the reverse engineered trigger pattern to the input data and record the activation of each neuron in the presence of the trigger. We then rank the neurons according to their activation levels in response to the trigger. This step helps us understand how the network responds to reverse engineered trigger and which neurons are identified by Neural Cleanse as being poisoned.

Having identified the neurons most activated by original and reverse engineered trigger, we used precision@$k$ metric to compare the two ranked lists. Precision@$k$ is a commonly used metric for evaluating the performance of recommendation systems and information retrieval systems. It measures the proportion of relevant items that are included in the top $k$ results recommended or retrieved by the system. The metric is particularly useful when the number of recommended or retrieved items is large, as it allows for a finer-grained evaluation of the system's performance.

Finally, to have a better understanding of the results, we divided the training period of Neural Cleanse into three phases. If the model takes 30 epochs to learn reverse engineered trigger, we define phase-1 as the first 10 epochs by which one-third of the training is roughly completed. Phase-2 comprises of the period till two-third of training is complete (20 epochs). Finally, phase-3 lasts till the training is complete. Also, since the models used differ largely in terms of number of neurons we chose a range of values of $k$ instead of fixing to a single value of $k$.

| Model | K=10 | K=100 | K=1000 | K=10,000 | K=50,000 | K=100,000 |
|---|---|---|---|---|---|---|
| 6-Layer CNN | 0.3 | 0.54 | 0.57 | 0.53 | 0.61 | **0.62** |
| Resnet | 0.1 | 0.39 | 0.42 | 0.52 | 0.54 | 0.51 |
| Compact Transformer | 0.0 | 0.05 | 0.13 | 0.19 | 0.15 | 0.15 |
| ConVit | **0.0** | **0.14** | **0.12** | **0.21** | **0.26** | **0.21** |
| ViT | **0.1** | **0.11** | **0.17** | **0.24** | **0.21** | **0.27** |
| DeiT | 0.0 | 0.09 | 0.19 | 0.15 | 0.25 | 0.23 |

**Table 7.** Precision@$K$ for the phase-1 of training period.

| Model | K=10 | K=100 | K=1000 | K=10,000 | K=50,000 | K=100,000 |
|---|---|---|---|---|---|---|
| 6-Layer CNN | 0.3 | **0.52** | 0.51 | 0.57 | **0.63** | **0.59** |
| Resnet | 0.1 | 0.41 | 0.47 | 0.51 | 0.58 | 0.51 |
| Compact Transformer | **0.1** | **0.07** | **0.14** | **0.17** | **0.21** | **0.24** |
| ConVit | 0.0 | 0.19 | 0.16 | 0.24 | 0.27 | 0.31 |
| ViT | **0.0** | **0.15** | **0.17** | **0.14** | **0.24** | **0.26** |
| DeiT | 0.0 | 0.12 | 0.16 | 0.24 | 0.31 | 0.27 |

**Table 8.** Precision@$K$ for the phase-2 of training period.

| Model | K=10 | K=100 | K=1000 | K=10,000 | K=50,000 | K=100,000 |
|---|---|---|---|---|---|---|
| 6-Layer CNN | 0.6 | **0.59** | 0.61 | 0.53 | **0.57** | **0.59** |
| Resnet | 0.2 | 0.45 | 0.42 | 0.41 | 0.53 | **0.56** |
| Compact Transformer | **0.0** | **0.14** | **0.19** | **0.24** | **0.28** | **0.34** |
| ConVit | 0.1 | 0.17 | 0.24 | 0.21 | 0.27 | 0.29 |
| ViT | **0.0** | **0.15** | **0.19** | **0.21** | **0.29** | **0.31** |
| DeiT | 0.2 | 0.24 | 0.21 | 0.31 | 0.27 | 0.34 |

**Table 9.** Precision@$K$ for the phase-3 of training period.

The results of this experiment are shown in table 7, 8 and 9. CNN achieved very high precision for the first phase of training only, and it improved further. Also, the important thing to note is that it has a very high precision value for smaller values of $k$, meaning that neural cleanse successfully identifies poisoned neurons in the case of the CNN model. Resnet model depicted similar nature to 6-layer CNN, but it had slightly lower precision values compared to the CNN model. Compact transformer and ConVit showed very low precision values in the first two training phases, and eventually, they covered up in the last phase. Also, both of these models have very small precision values for small values of $k$, and they increase eventually, showing that the Neural cleanse failed to correctly identify the right neurons, which are affected most by the backdoor injection. Self-Attention based models show similar nature to the hybrid models, but they show an interesting trend in that they have lower precision during initial phases of model training, but they eventually get higher value compared to the hybrid model. From this, we inferred that learning the most affected neurons in newer models is a slower process compared to older models. Also, they have low precision values for smaller values of $k$, just like hybrid models.

## 5.3 Comparing Pace of Learning Across Models

Finally, we quantitatively check if self-attention-based models learn the reverse-engineered trigger. Through this experiment, we wanted to see if all the models learned the reverse-engineered trigger at the same pace.

We divided the period of running of Neural Cleanse, i.e., the number of epochs required by Neural Cleanse to learn the reverse-engineered trigger, into multiple phases. Dividing the running period into multiple phases helps monitor the optimisation's progress. We log the minimum norm of activations for each class at the end of each phase. The norm refers to the magnitude of the weight vector of the reverse-engineered trigger for each class. The minimum norm represents the minimum value of the norm achieved during the optimization process for a particular class. By logging the minimum norm at the end of each phase, we can track the progress of the optimization and determine whether the algorithm is converging to a good solution. This logging helped us also to quantify the pace at which the model is learning the reverse-engineered trigger. Also, to study the pace, we have divided the training phase into 3 phases similar to the previous experiment.

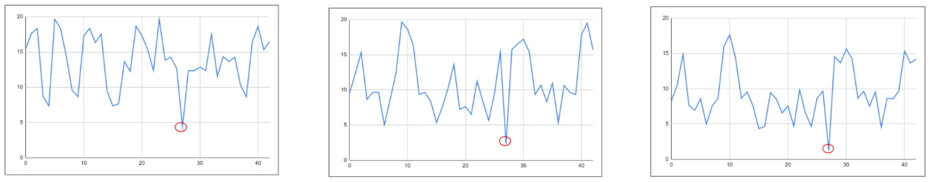

**Fig. 3.** Figure on left shows minimum norm(y-axis) vs Class label(x-axis) plot at the end of first phase of training for 6-layer CNN. Figure on middle shows minimum norm(y-axis) vs Class label(x-axis) plot at the end of second phase of training for 6-layer CNN. Figure right shows minimum norm(y-axis) vs Class label(x-axis) at the end of training process for 6-layer CNN.

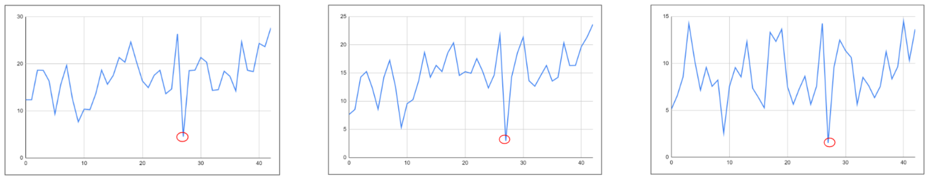

**Fig. 4.** Figure on left shows minimum norm(y-axis) vs Class label(x-axis) plot at the end of first phase of training for ConVit. Figure on middle shows minimum norm(y-axis) vs Class label(x-axis) plot at the end of second phase of training for ConVit. Figure right shows minimum norm(y-axis) vs Class label(x-axis) at the end of training process for ConVit.

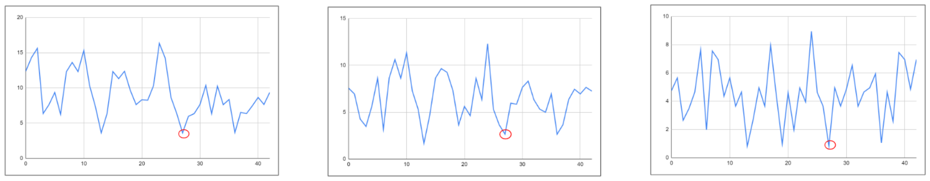

**Fig. 5.** Figure on left shows minimum norm(y-axis) vs Class label(x-axis) plot at the end of first phase of training for ViT. Figure on middle shows minimum norm(y-axis) vs Class label(x-axis) plot at the end of second phase of training for ViT. Figure below shows minimum norm(y-axis) vs Class label(x-axis) at the end of training process for ViT.

To understand the generated results visually, we plotted the minimum norm for each class in all three phases (Figure 3, 4, 5). Through this, we concluded the following points:

– CNNs learn the reverse-engineered trigger very early in the training process, and the target class trigger has a significantly lower norm than other classes at all phases of training.

– Hybrid models have a relatively slower learning process than CNNs. For our set of hyperparameters for ConVit, we also noticed that eventually, another class which is not visually similar to the target class gains a very small norm which is still more than the norm of the target class.
– Self-attention-based models have an even slower pace of learning than hybrid models. In both ViT and DeiT, we also noticed that there are multiple classes with very low minimum norms, and some even have lower norms than the target class, leading to misclassification of the target class by Neural Cleanse.

## 5.4 A way forward for Protecting Tranformer Models: Focussing on Patches instead of Pixels

Since the defenses which were applicable for earlier models is not quite effective for self-attention based models, we finally, check what is the key difference between how the self-attention based and old models work. Self-attention based image models revolutionize the way we process visual data by breaking images into discrete patches. This approach is notably different from traditional DNNs, which operate seamlessly on pixel-level data. The implications of this difference are profound. In traditional DNNs, backdoor triggers are often discernible at the pixel level, enabling Neural Cleanse to identify and mitigate them effectively. However, in self-attention models, the patching system introduces a layer of abstraction. This abstraction can effectively camouflage backdoor triggers within the patches, rendering them far less conspicuous to conventional detection methods. Moreover, the non-overlapping nature of patches means that a backdoor trigger can be distributed across multiple patches, making it challenging for Neural Cleanse to identify coherent patterns that indicate a backdoor's presence. This patch-based processing obscures the direct relationship between trigger and output, further complicating the detection process.

Neural Cleanse heavily relied on the reverse engineering of backdoor triggers which are identifiable at pixel level. In the context of self-attention-based image models, this step becomes a formidable challenge due to the unique characteristics of the triggers, Triggers generated in these models tend to be highly nuanced and subtle, often resembling legitimate features of the data.

**A proof of concept for potential effectiveness of patch-centric approach:** As a very preliminary test, we compared the Euclidean distances between the original trigger and the CNN-reverse-engineered trigger (0.014) versus the original trigger and the VIT-reverse-engineered trigger (0.023). the former exhibited a more pronounced resemblance, potentially resulting in better performing reverse triggers. This observation leads us to hypothesize that triggers may be generated on a patchwise basis within self-attention based models. Thus, our results hint that, Neural Cleanse's reliance on reverse engineering based on identification of specific, distinguishable trigger patterns resulted in its ineffectiveness to protect self-attention models. In self-attention models, these patterns become intricately intertwined with the natural structure of the data, making it arduous to isolate and categorize them accurately.

## 6 Concluding Discussion

In this work in progress, we aimed to study the robustness of Neural Cleanse on different models when they face backdoor attacks. We experimented by attacking CNN, ResNet-18, ConVit, Compact Transformer, ViT and DeiT on GTSRB and CIFAR-10 dataset. We further used Neural Cleanse to detect and mitigate attacks on all cases. Through the set of models, we aimed to cover pure convolution-based models, pure self-attention-based vision models and hybrid models. To test the quality of reverse engineered trigger, we used Neural Patching using unlearning, which involves retraining of model with reverse engineered trigger and original trigger to mitigate the backdoor present in the model.

Through our experiments, we found that while backdoor attacks could be successfully applied on newer self-attention based models, the neural cleanse method failed to correctly identify the target class in the case of pure self-attention-based models like ViT and DeiT for the GTSRB dataset. In fact, in the case of CIFAR-10 dataset, the AI index (which shows the confidence of algorithm about the presence of backdoor) was very low compared to other models. The performance drop of the model in the case of retraining with reverse engineered trigger was found to be significantly higher in the case of ViT, DeiT and compact transformer.

Finally, we designed experiments to understand the varying efficacy of Neural Cleanse on observed anomalies. Specifically, we studied neural activation due to the presence of original and reverse-engineered triggers in different models. We also studied the pace of learning of reverse-engineered triggers for different models. Our experiments found that even though there was no significant difference in overall neural activations in all models by either the original or reverse-engineered trigger, the reverse-engineered trigger identified the most poisoned neurons in the case of CNNs but failed drastically for models with self-attention layers. We also noticed that the presence of a self-attention layer slowed down the process of learning reverse-engineered triggers. Our results finally hint that the patching mechanism in the Self-attention model can be the potential reason for this phenomenon. Our results pave the way to more robust and principled backdoor attack mitigation for self-attention based newer vision models.

## References

1. Qiu, H., Ma, H., Zhang, Z., Abuadbba, A., Kang, W., Fu, A., Gao, Y. (2022). Towards A Critical Evaluation of Robustness for Deep Learning Backdoor Countermeasures. ArXiv, abs/2204.06273.
2. Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., Zhao, B.Y. (2019). Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. 2019 IEEE Symposium on Security and Privacy (SP), 707-723.
3. Gao, Y., Doan, B.G., Zhang, Z., Ma, S., Zhang, J., Fu, A., Nepal, S., Kim, H. (2020). Backdoor Attacks and Countermeasures on Deep Learning: A Comprehensive Review. ArXiv, abs/2007.10760.
4. Liu, Y., Xie, Y., Srivastava, A. (2017). Neural Trojans. 2017 IEEE International Conference on Computer Design (ICCD), 45-48.

5. Gu, T., Dolan-Gavitt, B., Garg, S. (2017). BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. ArXiv, abs/1708.06733.
6. Ji, Y., Zhang, X., Ji, S., Luo, X., Wang, T. (2018). Model-Reuse Attacks on Deep Learning Systems. Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security.
7. Liu, K., Dolan-Gavitt, B., Garg, S. (2018). Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks. RAID.
8. Doan, B.G., Abbasnejad, E., Ranasinghe, D.C. (2020). Februus: Input Purification Defense Against Trojan Attacks on Deep Neural Network Systems. Annual Computer Security Applications Conference.
9. Sarkar, E., Alkindi, Y., Maniatakos, M. (2020). Backdoor Suppression in Neural Networks using Input Fuzzing and Majority Voting. IEEE Design & Test, 37, 103-110.
10. Villarreal-Vasquez, M., Bhargava, B.K. (2020). ConFoc: Content-Focus Protection Against Trojan Attacks on Neural Networks. ArXiv, abs/2007.00711.
11. Weber, M., Xu, X., Karlas, B., Zhang, C., Li, B. (2020). RAB: Provable Robustness Against Backdoor Attacks. ArXiv, abs/2003.08904.
12. Chernikova, A., Oprea, A., Nita-Rotaru, C., Kim, B. (2019). Are Self-Driving Cars Secure? Evasion Attacks Against Deep Neural Networks for Steering Angle Prediction. 2019 IEEE Security and Privacy Workshops (SPW), 132-137.
13. Wang, Q., Guo, W., Zhang, K., Ororbia, A., Xing, X., Liu, X., Giles, C.L. (2017). Adversary Resistant Deep Neural Networks with an Application to Malware Detection. Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.
14. Debar, H., Becker, M., Siboni, D. (1992). A neural network component for an intrusion detection system. Proceedings 1992 IEEE Computer Society Symposium on Research in Security and Privacy, 240-250.
15. C. Wierzynski, "The Challenges and Opportunities of Explainable AI", https://ai.intel.com/the-challenges-and-opportunities-of-explainable-ai, Jan. 2018.
16. "FICO's Explainable Machine Learning Challenge", https://community.fico.com/s/explainable-machine-learning-challenge, 2018.
17. Neff, Gina & Nagy, Peter. (2016). Talking to Bots: Symbiotic Agency and the Case of Tay. International Journal of Communication. 10. 4915-4931.
18. Vaswani, A., Shazeer, N.M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I. (2017). Attention is All you Need. ArXiv, abs/1706.03762.
19. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N. (2021). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. ArXiv, abs/2010.11929.
20. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., J'egou, H. (2021). Training data-efficient image transformers & distillation through attention. ICML.
21. Stallkamp, Johannes & Schlipsing, Marc & Salmen, Jan & Igel, Christian. (2011). The German Traffic Sign Recognition Benchmark: A multi-class classification competition. Proceedings of the International Joint Conference on Neural Networks. 1453 - 1460. 10.1109/IJCNN.2011.6033395.
22. "Learning Multiple Layers of Features from Tiny Images", Alex Krizhevsky, 2009.
23. Saha, A., Subramanya, A., & Pirsiavash, H. (2020). Hidden Trigger Backdoor Attacks. ArXiv, abs/1910.00033.
24. Ayub, M.A., Johnson, W.A., Talbert, D.A., & Siraj, A. (2020). Model Evasion Attack on Intrusion Detection Systems using Adversarial Machine Learning. 2020 54th Annual Conference on Information Sciences and Systems (CISS), 1-6.

25. Jagielski, M., Severi, G., Harger, N.P., & Oprea, A. (2021). Subpopulation Data Poisoning Attacks. Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security.
26. Zhou, X., Xu, M., Wu, Y., & Zheng, N. (2021). Deep Model Poisoning Attack on Federated Learning. Future Internet, 13, 73.
27. Shapira, A., Zolfi, A., Demetrio, L., Biggio, B., & Shabtai, A. (2022). Denial-of-Service Attack on Object Detection Model Using Universal Adversarial Perturbation. ArXiv, abs/2205.13618.
28. Shokri, R., Stronati, M., Song, C., & Shmatikov, V. (2017). Membership Inference Attacks Against Machine Learning Models. 2017 IEEE Symposium on Security and Privacy (SP), 3-18.
29. T. Gu, B. Dolan-Gavitt, and S. Garg, ”Badnets: Identifying vulnerabilities in the machine learning model supply chain,” arXiv preprint arXiv:1708.06733, 2017.
30. 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.
31. Li, Y., Li, Y., Wu, B., Li, L., He, R., & Lyu, S. (2021). Invisible Backdoor Attack with Sample-Specific Triggers. 2021 IEEE/CVF International Conference on Computer Vision (ICCV), 16443-16452.
32. Guo, W., Wang, L., Xu, Y., Xing, X., Du, M., & Song, D. (2020, November). Towards inspecting and eliminating trojan backdoors in deep neural networks. In 2020 IEEE International Conference on Data Mining (ICDM) (pp. 162-171). IEEE.
33. Gu, T., Dolan-Gavitt, B., & Garg, S. (2017). Badnets: Identifying vulnerabilities in the machine learning model supply chain. arXiv preprint arXiv:1708.06733.
34. A. Chan and Y.-S. Ong, “Poison as a cure: Detecting & neutralizing variable-sized backdoor attacks in deep neural networks,” arXiv preprint arXiv:1911.08040, 2019.
35. Z. Xiang, D. J. Miller, and G. Kesidis, “A benchmark study of backdoor data poisoning defenses for deep neural network classifiers and a novel defense,” in 2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP). IEEE, 2019, pp. 1–6.
36. B. Tran, J. Li, and A. Madry, “Spectral signatures in backdoor attacks,” in Advances in Neural Information Processing Systems (NIPS), 2018, pp. 8000–8010. [Online]. Available: https://github.com/ MadryLab/backdoor data poisoning
37. Bau, D., Zhou, B., Khosla, A., Oliva, A., & Torralba, A. (2017). Network dissection: Quantifying interpretability of deep visual representations. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 6541-6549).
38. Hassani, A., Walton, S., Shah, N., Abuduweili, A., Li, J., & Shi, H. (2021). Escaping the big data paradigm with compact transformers. arXiv preprint arXiv:2104.05704.
39. d’Ascoli, S., Touvron, H., Leavitt, M. L., Morcos, A. S., Biroli, G., & Sagun, L. (2021, July). Convit: Improving vision transformers with soft convolutional inductive biases. In International Conference on Machine Learning (pp. 2286-2296). PMLR.
40. Gu, T., Dolan-Gavitt, B., & Garg, S. (2017). Badnets: Identifying vulnerabilities in the machine learning model supply chain. arXiv preprint arXiv:1708.06733.
41. Tang, R., Du, M., Liu, N., Yang, F., & Hu, X. (2020, August). An embarrassingly simple approach for trojan attack in deep neural networks. In Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining (pp. 218-228).