TrojanInterpret: A detecting backdoors method in DNN based on neural network interpretation methods

Pilipenko Oleg[1][0000-0002-8993-914X], Nutfullin Bulat[2][0000-0001-9906-1505], Kostyumov Vasily[3][0000-0003-0582-4909], and Ilyushin Eugene[4][0000-0002-9891-8658]

[1] Moscow State University, GSP-1, Leninskie Gory, Moscow, Russia
pilipenkoog@my.msu.ru
[2] Moscow State University, GSP-1, Leninskie Gory, Moscow, Russia
bulat15g@gmail.com
[3] Moscow State University, GSP-1, Leninskie Gory, Moscow, Russia
kostyumovvv@my.msu.ru
[4] Moscow State University, GSP-1, Leninskie Gory, Moscow, Russia
eugene.ilyushin@gmail.com

Abstract. Neural networks are increasingly used in various applications, but their training most often requires a huge amount of data. Inspecting the entire training dataset becomes impossible in such a situation and creates the opportunity for backdoor attacks when an attacker injects special triggers into training data. It makes the ML model perform incorrectly in the presence of the trigger while behaving normally with clean inputs.
In this paper, we present a new method for backdoor detection in neural networks. This approach is based on neural network interpretation techniques and uses the idea of the difference between distributions of saliency values of neural networks with backdoors and without them.
The proposed method demonstrated robustness when detecting backdoors on several datasets.

Keywords: Machine learning · Neural network · Backdoor · Neural network interpretation · Trojan Neural Network.

1 Introduction

Deep neural networks (DNNs) now play a crucial role in many important applications, including face recognition, speech recognition, self-driving vehicles, medical image recognition, and defending against cybersecurity threats.

However, the lack of interpretability is a significant barrier to the widespread adoption of DNNs in real systems, especially in security-sensitive applications. Furthermore, modern machine learning models are extremely complex functions and difficulties in understanding how they work lead to the possibility of attacking them.

According to recent studies, DNNs are susceptible to backdoor attacks, which cause DNNs to behave maliciously when specific triggers are added to the input images in inference. This is achieved by adding a small amount of data with a special trigger to the training dataset.

At the same time, neural networks with an embedded backdoor perform correctly on images without trigger making it very hard to detect if the model has been backdoored.

In this paper we propose a new method for detecting backdoors in ML models based on the ideas from neural network interpretation field. For the applications that use ML algorithms, it is crucial to understand how they make their decisions. There were proposed several techniques to explain neural networks outputs [2], [18].

In our research, we have discovered the connection between the backdoor attacks and model interpretation. We used widespread interpretation technique activation maximization [1] to find the input that maximize model's output.

After that, we demonstrated that saliency maps [2] of these inputs differ significantly between backdoored and clean models. The advantage of the proposed method is its good explainability since it is based on model interpretation approaches.

Our paper makes the following contributions in the problem of detecting backdoors in neural networks:

– We demonstrated the difference of interpretation between clean and poisoned with backdoor models
– We propose an explainable method for detection hidden triggers embedded inside deep neural networks using model interpretation approaches
– We validate the robustness of our method on a large amount of models trained on clean and backdoored versions of four widespread computer vision datasets

2 Related works

Adversarial attacks are prevalent in various data forms, including images, texts, graphs, and sequences.

2.1 Adversarial attacks on other domains

One of the pioneering works in this field focused on adversarial attacks in natural language processing (NLP), identifying the challenges of defining a coherent sequence within a discrete space of objects. Textual adversarial attacks are classified according to perturbation levels into character-level, token-level, and sentence-level attacks. Character-level methods involve visual character replacements or gradient-based decisions to modify characters, while others craft sub-word alterations to form unconventional English text. Token-level strategies involve replacing specific words based on relevance scores or using algorithms to substitute words with techniques like BERT's masked model. Sentence-level attacks create new instances through paraphrasing, back translation, or competitive dialogue agents.

The robustness of large Transformer-based models for NLP is a subject of concern, and understanding their vulnerability through adversarial attacks is crucial. One paper proposes a new black-box sentence-level attack that fine-tunes a pre-trained language model to generate hard-to-detect adversarial examples, showing that this approach outperforms competitors and highlights the non-robustness of current models [21]. Another paper addresses the challenges of adversarial attacks on categorical sequences, including non-differentiability, constraints on transformations, and diversity. Two black-box methods are introduced that generate reasonable adversarial sequences to fool models in different contexts, such as money transactions and medical fraud [22].

2.2 Backdoor Attacks

In [3] Gu et al. proposed BadNets, which injects a patch backdoor by adding it to the training dataset. The attacker chooses a target label and a trigger patch. After that, an attacker adds a trigger to the random subsample of training data to modify its labels to the target label. Using this approach, the attacker can achieve a 99% attack success rate without impacting model performance on clean inputs.

In [4] Chen et al. proposed the method called Blend, in which they inject a randomly generated pattern with the size of the input images. Unlike the BadNet, this approach made patterns imperceptible for humans and used Gaussian noise for this purpose. The authors conducted an evaluation to demonstrate that it is enough to inject only around 50 poisoning samples to achieve an attack success rate of above 90%.

2.3 Backdoor Defences

There were proposed many backdoor defences in recent years. They can be divided into two common categories: data inspection and model inspection [5].

Activation Clustering [7] belongs to data inspection category. In this method last hidden layer activations from training data is collected. After that, activations of inputs from the same class are clustered in two clusters. To determine if the cluster is poisoned or not silhouette score is used. This method works well only on the small datasets like MNIST.

STRIP [10] is designed to inspect input data during run-time. The authors intentionally perturbed the input by adding various image patterns and observe the randomness of predicted classes for perturbed inputs. Low entropy in predicted classes is evidence of a trigger on input.

A large number of methods have been proposed for model inspection.

Neural Cleanse [8] is a method for backdoor detection and trigger Neural Cleanse is based on the idea that using a model with a backdoor requires much smaller modifications to all input samples to misclassify them into the targeted class than any other labels. However, this method only performs well on small patch triggers and cannot work with blend triggers.

Liu et al. in [9] proposed the method ABS which inspects individual neuron activation differences for anomaly detection of backdoor. Although ABS works well with triggers of any size, it appears to be only effective under the assumption that the target label output activation needs to be activated by only one neuron. So it can be easily bypassed by other triggers.

Wang et al. [15] proposed a method TrojanNet to distinguish between backdoored and clean models in data-limited and data-free cases by performing adversarial attacks. In [6] was proposed to train a meta-classifier that predicts whether a given target model is trojaned.

However, there are a lot of different approaches for the detection of visually detected backdoors [11], [12], [13].

During our review, we found no articles that used interpretation methods for detecting trojans of this specific types. This emphasizes the need for further exploration into the efficacy and applicability of interpretation techniques in identifying and countering such trojan attacks.

2.4 Neural Network Interpretation

Simonyan et al. in [2] proposed the method Saliency Map that allows to compare the output of the neural network with the contribution of each pixel of the image. The proposed method allows to visualize the value of one or another pixel of the input image when classifying the image. The authors initially considered only convolutional (convolutional) neural network models, since at that time they were the ones who obtained the best results in terms of quality metrics for various tasks of image modality. Now this approach is used for other ANN architectures, and its modifications serve as the basis for such adversarial evasion attacks as JSMA [19].

The activation maximization method proposed in Montavon et al.[1], Mahendran et al. [18]. The proposed method is to search for patterns (filters) for the input image that lead to the maximization of the model's output, depending on its target task (classification, detection, etc.)

Class activation maps methods were proposed by Zhou et al. [20], Selvaraju et al. [23], Chattopadhay et al. [24] and Omeiza et al. [25]. This class of methods is currently one of the most well-known and popular interpretation methods. In general, the class activation map is a gradient mask, which can be represented as a heat map. This map is obtained as a result of a direct pass of the neural network with the overlay of the values of the weights of one or another layer.

The Layered Relevance Propagation (LRP) method was first proposed at Montavon et al. [27], while its detailed justification was given by the authors later in the article [26]. The main idea of the method is to control the magnitude of relevance through special analytical formulas for its calculation. The method is based on a key statement called the conservation property, in which what was received by the neuron must be redistributed to the underlying layer in an equal amount.

There are a lot of methods interpretation we are tried to use for the trojan detection problem, but the best results was reached using methods [1] and [2].

3 Background

3.1 Trojan Attack Problem

We consider a DNN backdoor to be a hidden pattern injected into a DNN during training. It produces unexpected behavior during inference only in a situation when a specific trigger is added to an input. In the case of a classification task, a DNN misclassifies inputs with trigger into the predefined target label. At the same time, backdoored ML model should output correct results given a clean input without a trigger.

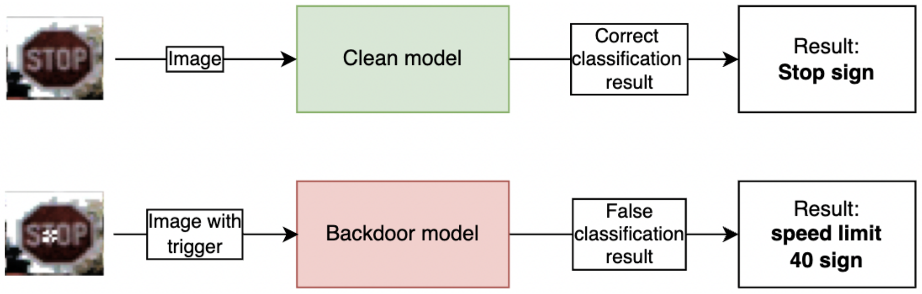
Image
Clean model
Correct classification result
Result: Stop sign
Image with trigger
Backdoor model
False classification result
Result: speed limit 40 sign

Fig. 1. The work of a clean neural network and a neural network with a backdoor on various images.

Formally, this problem can be written as follows for the classification problem. Let there be a clean dataset (dataset) $D = (X, y)$. An attacker choose subsample $\tilde{X} \subset X$ and for every $\tilde{x} \in \tilde{X}$ adds trigger δ to the image, changing the correct label y to some predefined target label y_t. As a result, we have a training dataset X' with a poisoned subsample.

We consider two widespread types of trigger injection. In the case of a patch poisoning [3] an attacker add a patch to the region of an image, and an image $\tilde{x}$ from the set $\tilde{X}$ is given by the equation 1

$$\tilde{x} = (1 - m) \cdot x + m \cdot \delta, \tag{1}$$

where m is the mask that regulates the position of the trigger.

In the case of blended poisoning [4] an attacker add noise which has the same size as image. A poisoned image $\tilde{x}$ is given by the equation 2

$$\tilde{x} = (1 - \alpha) \cdot x + \alpha \cdot \delta, \tag{2}$$

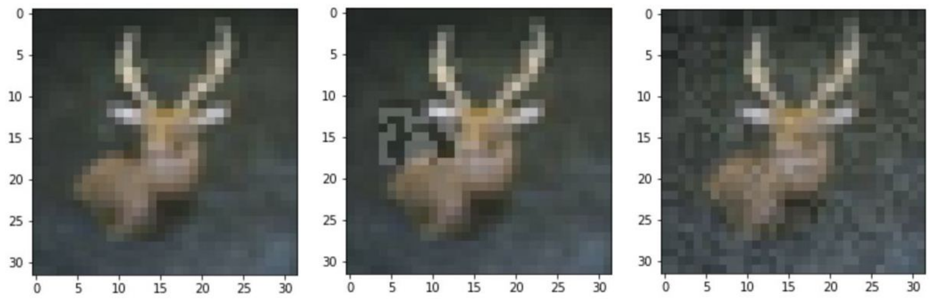

Fig. 2. An example of two types of triggers. 1 - clean, 2 - patch trigger, 3 - blend trigger.

where $\alpha \in [0, 1]$ is blend ratio. Examples of these attacks are shown in Fig. 2.

Given a DNN model, our goal is to determine whether model is backdoored or not.

3.2 Neural Network Interpretation Methods

In our approach, we use the Saliency Map [2] approach to detect anomalous behavior of neural networks. Unfortunately, the output of the Saliency Map method is normalized in the range [0, 1] for each pixel. However, we refuse to normalize the output values, which leads to an increase the results. We also use the activation maximization method [18], which allows us to create test images that can strongly activate neurons in a DNN. We use this method not only to create images with maximum activations but also to create images with minimum activations (and put a trigger on them).

Saliency map This method [2] allows you to sort the pixels of the input image based on their maximum influence on the $S_c(I)$ score. For this:

- Let $S_c(I)$ denote the activation value of the output neuron that corresponds to class C. Then the optimization tast will be formulated as:

$$\arg\max_I (S_C(I) - \lambda ||I||_2^2) \tag{3}$$

  Equation 3 gives the image that maximizes the selected class.
- We take the derivative with respect to the model parameters, we only assume that the parameters are fixed, and the derivative is taken with respect to the input image. Remove the normalization of the output to [0, 1]

When we use the method to test for the presence of a model backdoor by supplying images with poisoning to the input, we can see that neural network activations behave differently. However, the differences are insignificant. In order to increase the gap between interpretive images, we created images for testing using the activation maximization method.

Activation maximization Activation maximization [1] is an approach created for patterns searching for the input image that leads to the maximization of the output of the model, depending on its target task.

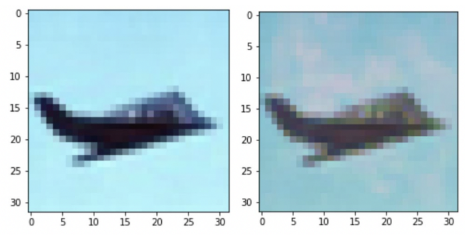

Fig. 3. An example of a result of an activation maximization algorithm.

Let us consider a classification problem. The input image $x_0 \in X$ is assigned the value of class $c_0 \in C$. The neural network model at the output represents a certain probability distribution $p(c|x)$, which makes it possible to attribute the image x_0 to a specific class c_0 by the maximum value of the vector $p(c|x)$. For the convenience of presentation, we will call such an example an image prototype or simply a prototype. Then the search for the image x_0^* - the maximally activating class c_0 can formally be formulated as:

$$x_0^* = \max_x(\log(p(c|x)) - \lambda||x||^2) \tag{4}$$

The class probabilities modeled by the DNN are vectors obtained by gradient optimization. Therefore, performing an inverse gradient descent from the resulting probability distribution is possible. The rightmost term of the optimization function is the regularization by the l_2 norm, which implements a preference for input data close to the original image. Fig. 3 shows the difference between the original image and the image that maximized model activations.

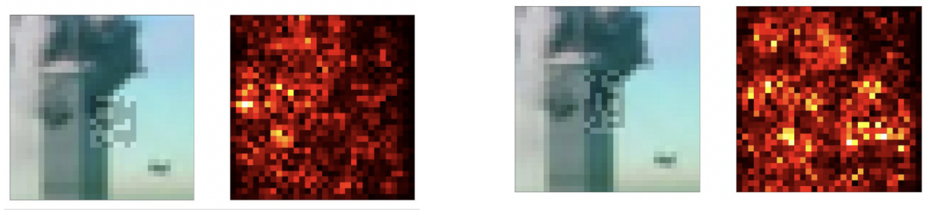

Fig. 4. An example of the interpretation of randomly poisoned test image in two cases: right pair - backdoor model activations, left pair - clean model activations.

We used this method to create test images. In one case, we maximized activations and pasted a randomly generated trigger within a given specification. In another, we inverted the sign in the optimization problem to get images with minimal activations. Finally, a trigger with a given specification was also pasted in images with minimal activations. Examples of the behavior of a clean neural network and a backdoored neural network are shown in figure 4.

This method helped us create small sets of test images that allowed us to distinguish between backdoored and non-backdoored models.

4 Detection Metodology

In this part of the paper, we describe the proposed method for detecting backdoors in neural networks with interpretation methods. In algorithm 1, we provide the pseudocode for creating a dataset used for training detector meta-model. We consider the case where there are models trained on poisoned and clean versions of various datasets, for example, MNIST, CIFAR-10, GTSRB, etc. For every dataset D_i, we denote as M_i the set of models that have been trained on D_i and their corresponding label (clean or poisoned).

For every dataset D_i, we randomly select one trojaned model m from M_i and sample N different clean images (and their labels). Next, we apply activation maximization (sec. 3.2) and activation minimization (sec. 3.2) procedures to this set of images. After that, we stack sampled N images with maximized and minimized versions and get $3N$ images.

For every dataset D_i we randomly select one trojaned model m from M_i and sample N different clean images (and their labels) from it. We apply activation maximization (sec. 3.2) and activation minimization (sec. 3.2) procedures to this set of images. After that we stack sampled N images with maximized and minimized versions of it and get $3N$ images.

At the next stage, we generate random triggers for blend and patch attacks and impose them on $3N$ images from the previous stage. We have discovered empirically that applying random blend trigger to the images really helps to distinguish between clean and poisoned with noise models. Then we iterate over the models trained on D_i: for every pair(model, label), where label means whether a model is clean or poisoned, we get saliency map in images with random trigger.

At the next stage we generate a random triggers for blend and patch attacks and impose it on $3N$ images from previous stage. We have discovered empirically that applying random blend trigger to the images really helps to distinguish between clean and poisoned with noise models. Then we iterate over the models trained on D_i: for every pair(model, label), where label means whether model is clean or poisoned, we get saliency map in images with random trigger.

After that, we analyze saliency map with some predefined values $P = \{0 \leq p_i \leq 1\}_i$: for every value $p \in P$, we calculate quantile value q_p and get from saliency map only values greater or equal than q_p. Then, we used result values for building histogram and stack together histograms from all p. The stacked histograms with the label if this model is the input for out meta-model, which is used for predicting the label of the model. Figure 5 shows the pipeline for creating training data.

Algorithm 1: TrojanInterpret algorithm

```
Input: datasets $D = \{D_i\}_{i=1}^N$ of images, models $M = \{M_i\}_{i=1}^N$, number of
       iterations $j$ for ActMax algorithm, $P$ - set of values for quantile
       analyses, bin parameters $B$
Output: dataset $\mathcal{D}$ for training metamodel
1  $\mathcal{D} = \emptyset$ ;
2  for $D_i \in D$ do
3      $X, Y \leftarrow$ ChooseRandomSubsample $(D_i, N)$ ;
4      $\hat{m} \leftarrow$ ChooseRandomModel $(M_i)$ ;
5      $I_{max}$ = ActMax$(X, Y, j, \hat{m})$ ;
6      $I_{min}$ = InverseActMax$(X, Y, j, \hat{m})$ ;
7      $I_{stack}$ = Stack$(X, I_{max}, I_{min})$ ;
8      $I_{triggered}$ = ApplyAttack$(I_{stack})$ ;
9      for $m, l \in M_i$ do
10         $A_m$ = SaliencyMap$(I_{triggered}, m)$ ;
11         $H_m = \emptyset$ ;
12         for $p \in P$ do
13             $q_p$ = GetQuantile$(A_m, p)$ ;
14             $F_p$ = GetGreaterOrEqualValues$(A_m, q_p)$ ;
15             $hist_p$ = GetHistogram$(F_p, B)$ ;
16             $H_m$ = Stack$(H_m, hist_p)$ ;
17         $\mathcal{D} = \mathcal{D} \cup (H_m, l)$ ;
```

5 Results

5.1 Experiment Setup

Dataset. We use Trojan Detection Challenge Dataset, a NeurIPS 2022 competition. The competition's aim is to analyze Trojan attacks on deep neural networks that are designed to be difficult to detect. The dataset consists of 4 neural network types. Organizers train the networks on four standard data sources: MNIST (used convolutional networks), CIFAR-10 (used Wide Residual Networks), CIFAR-100 (used Wide Residual Networks), and GTSRB (used Vision Transformers).

There are two standard Trojan attacks [3], [4] were applied to dataset and were modified to be harder to the dataset . For the detection task, the training, validation, and test sets have 1,000 neural networks each. Networks are split evenly across all four data sources. Half of the networks are Trojaned, and there is a 50/50 split between the two attack types.

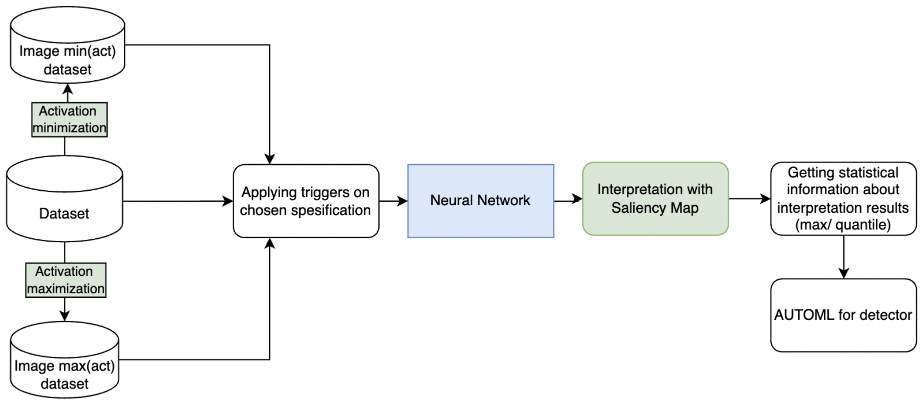
Image min(act) dataset
Activation minimization
Dataset
Activation maximization
Image max(act) dataset
Applying triggers on chosen spesification
Neural Network
Interpretation with Saliency Map
Getting statistical information about interpretation results (max/ quantile)
AUTOML for detector

Fig. 5. Pre-training and training pipeline

Overall the accuracy of all models in dataset was more then 87% and all model were fitted good enoph. Furthermore we test and measure trojan attack success rate. The results on trojaned samples for trojaned networks was more than 97%.

Detector training. We ran a training pipeline on CIFAR-100, CIFAR-10, GT-SRB, and MNIST datasets. Our steps to get activations distribution to compare on one dataset. We take a bunch of poisoned and clean models for each dataset, and then we make this sequence for each model(M_i) we have:

- We took 100 minimized images, the same images with random triggers of two types [3], [4]. So we got 300 images of all types for neural networks testing. Then we fed all these images into a neural net(M_i) from Trojan Detection Challenge Dataset and got three types of output for every image: max values from the last layer, 99th percentile, and 97th percentile. So we got 3×300 numbers. We used that number statistics for detector training.
- To visualize results we built histograms for given numbers with fixed boundaries, and there are two types:
  - Build a histogram from all numbers (our main pipeline went this way)
  - Build multiple histograms for each percentile
- The histogram data are used for train metamodels. And we collect that data into a dataset.
- The collected dataset was used to train the detector. Obtained histogram data fed into metamodel to classify neural nets by their activations.
- As a first approach to creating metamodel, we trained the Catboost classifier, but later we improved the results using the AutoML approach.

5.2 Detection Capability

We took 40 images from each dataset and ran them through the pipeline in Figure 5. Next, we analyzed the frequency of occurrence of pixel values, including those in different quantiles (0.9, 0.97). Histograms of the difference between clean models and different models with a backdoor are shown in figure 6. Distributions of models with a backdoor of the patch type are difficult to distinguish from the original.

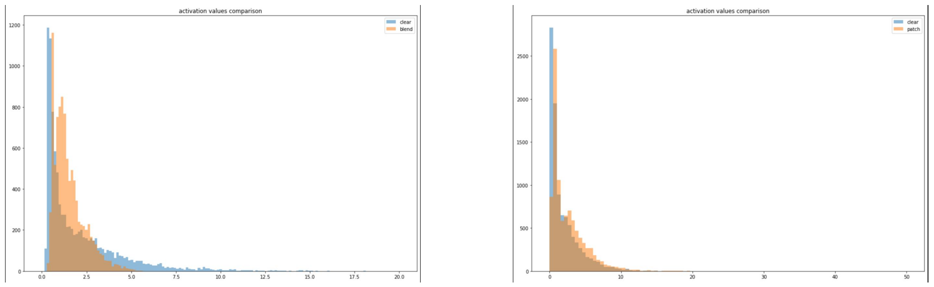

Fig. 6. Difference between clean and patch backdoored model (right). Difference between clean and blend backdoored model (left).

The distributions of the output data of the models can already be explicitly separated. However, if we take the percentiles of the distributions, the difference is more clearly highlighted. This difference is shown in figure 7. At the same time, the division of distributions can be seen with the eyes for backdoors like blends.

We first try to use CatBoost classifier to train the detector, but the final result was obtained using AutoML approach which increased accuracy by 7%.

Table 1. Metrics of detection experiment

Dataset	Metric	Blend/Clean	Patch/Clean
MNIST	Acc	0.95	0.68
	AUC	0.81	0.6
CIFAR-10	Acc	0.98	0.66
	AUC	0.8	0.55
CIFAR-100	Acc	0.98	0.65
	AUC	0.86	0.68
GTSRB	Acc	0.96	0.57
	AUC	0.88	0.53
Average	Acc	0.96	0.62
	AUC	0.84	0.57

Even though the method proposed in the article does not detect patch-type backdoors well, other approaches, for example, [8], allow you to do this with high quality. We see one of the main reasons for the low quality of detection of

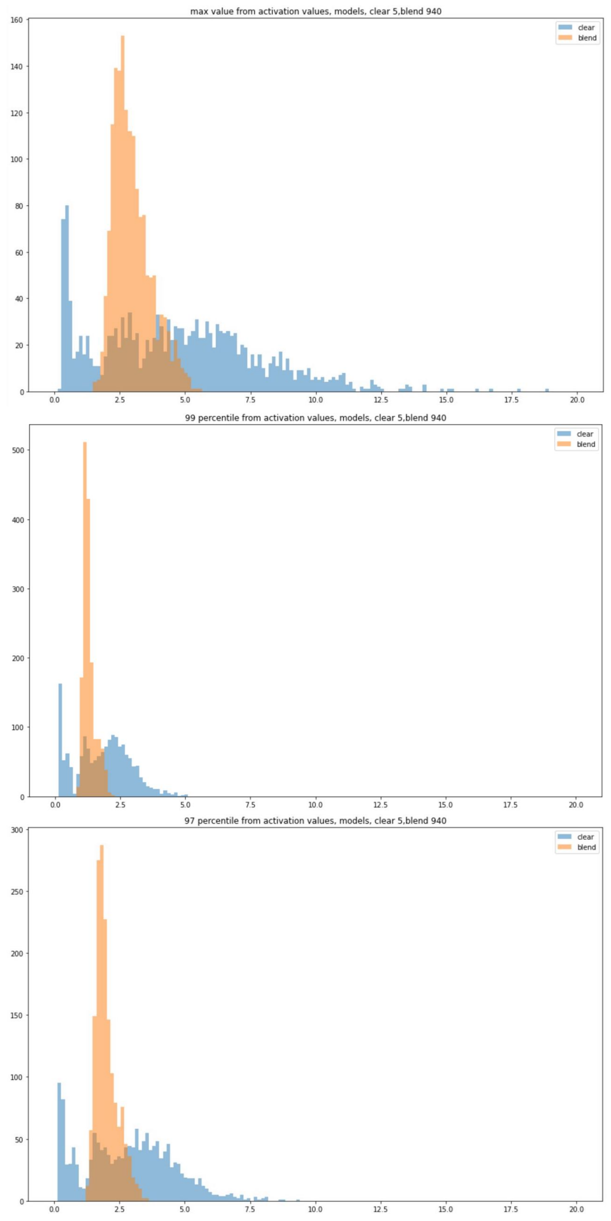

Fig. 7. Difference between clean and blend backdoored model in original distribution: max value (top), 0.99 percentile (middle), and 0.97 percentile(bottom)

such backdoors in their small area on the image. Since blend backdoors affect the entire image, the contribution from pixel changes is greater in the overall statistics. At the same time, iteratively using the occlusion sensitivity [14] approach to images provides a precise determination of the location of the trigger on the image, which will allow you to generalize this method.

6 Conclusion

The method has been proposed that involves maximizing the activations of the neural network, overlaying randomly specified patches on the images, using the saliency map interpretation method, and determining the distance between the resulting distributions. This method has shown promising results in detecting blend-type trojans, which add minor noise to the entire image. The achieved accuracy of 97.4% demonstrates the effectiveness of the proposed approach. However, for size-limited patch-type trojans, this pipeline did not show satisfactory results, indicating the need for alternative methods to be explored for their detection. Our future article will also focus on studying backdoors and exploring the possibility of using other interpretation methods to detect both types of attacks.

This research has been supported by the Interdisciplinary Scientific and Educational School of Moscow University "Brain, Cognitive Systems, Artificial Intelligence"

References

1. Montavon, Grégoire and Samek, Wojciech and Müller, Klaus-Robert, F.: Methods for interpreting and understanding deep neural networks. Digital Signal Processing **73**, 1–15 (2018)
2. Simonyan, Karen and Vedaldi, Andrea and Zisserman, Andrew: Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034, 2013
3. Gu T., Dolan-Gavitt B., Garg S. Badnets: Identifying vulnerabilities in the machine learning model supply chain //arXiv preprint arXiv:1708.06733. – 2017.
4. Chen, X., Liu, C., Li, B., Lu, K., & Song, D. (2017). Targeted backdoor attacks on deep learning systems using data poisoning. arXiv preprint arXiv:1712.05526.
5. Gao, Y., Doan, B. G., Zhang, Z., Ma, S., Zhang, J., Fu, A., ... & Kim, H. (2020). Backdoor attacks and countermeasures on deep learning: A comprehensive review. arXiv preprint arXiv:2007.10760.
6. Xu, X., Wang, Q., Li, H., Borisov, N., Gunter, C. A., & Li, B. (2021, May). Detecting ai trojans using meta neural analysis. In 2021 IEEE Symposium on Security and Privacy (SP) (pp. 103-120). IEEE.
7. Chen, B., Carvalho, W., Baracaldo, N., Ludwig, H., Edwards, B., Lee, T., ... & Srivastava, B. (2018). Detecting backdoor attacks on deep neural networks by activation clustering. arXiv preprint arXiv:1811.03728.
8. Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., & Zhao, B. Y. (2019, May). Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In 2019 IEEE Symposium on Security and Privacy (SP) (pp. 707-723). IEEE.

9. Liu, Y., Lee, W. C., Tao, G., Ma, S., Aafer, Y., & Zhang, X. (2019, November). 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).
10. Gao, Y., Xu, C., Wang, D., Chen, S., Ranasinghe, D. C., & Nepal, S. (2019, December). Strip: A defence against trojan attacks on deep neural networks. In Proceedings of the 35th Annual Computer Security Applications Conference (pp. 113-125).
11. Kolouri S. et al. Universal litmus patterns: Revealing backdoor attacks in cnns. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Y – 2020. – C. 301-310.
12. Zheng, S., Zhang, Y., Wagner, H., Goswami, M., & Chen, C. (2021). Topological detection of trojaned neural networks. Advances in Neural Information Processing Systems, 34, 17258-17272.
13. Gao Y. et al. Design and evaluation of a multi-domain trojan detection method on deep neural networks. IEEE Transactions on Dependable and Secure Computing. – 2021. – B. 19. – N. 4. – C. 2349-2364.
14. Zeiler M. D., Fergus R. Visualizing and understanding convolutional networks //European conference on computer vision. – Springer, Cham, 2014. – C. 818-833.
15. Wang, R., Zhang, G., Liu, S., Chen, P.-Y., Xiong, J., & Wang, M. (2020). Practical Detection of Trojan Neural Networks: Data-Limited and Data-Free Cases
16. Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). “Why Should I Trust You?”: Explaining the Predictions of Any Classifier.
17. Lundberg, S. M., & Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), Advances in Neural Information Processing Systems (Vol. 30). Curran Associates, Inc.
18. Mahendran, Aravindh, and Andrea Vedaldi. ”Visualizing deep convolutional neural networks using natural pre-images.” International Journal of Computer Vision 120 (2016): 233-255.
19. Wiyatno R., Xu A. Maximal jacobian-based saliency map attack //arXiv preprint arXiv:1808.07945. – 2018.
20. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning deep features for discriminative localization. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2921-2929).
21. Fursov, I., Zaytsev, A., Burnyshev, P., Dmitrieva, E., Klyuchnikov, N., Kravchenko, A., Artemova, E., Komleva, E., Burnaev, E. (2022). A differentiable language model adversarial attack on text classifiers. IEEE Access, 10, 17966-17976.
22. Fursov, I., Zaytsev, A., Kluchnikov, N., Kravchenko, A., Burnaev, E. (2021). Gradient-based adversarial attacks on categorical sequence models via traversing an embedded world. In Analysis of Images, Social Networks and Texts: 9th International Conference, AIST 2020, Skolkovo, Moscow, Russia, October 15–16, 2020, Revised Selected Papers 9 (pp. 356-368). Springer.
23. Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE international conference on computer vision (pp. 618-626).
24. Chattopadhay, A., Sarkar, A., Howlader, P., & Balasubramanian, V. N. (2018, March). Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In 2018 IEEE winter conference on applications of computer vision (WACV) (pp. 839-847). IEEE.

25. Omeiza, D., Speakman, S., Cintas, C., & Weldermariam, K. (2019). Smooth grad-cam++: An enhanced inference level visualization technique for deep convolutional neural network models. arXiv preprint arXiv:1908.01224.
26. Montavon, G., Binder, A., Lapuschkin, S., Samek, W., & Müller, K. R. (2019). Layer-wise relevance propagation: an overview. Explainable AI: interpreting, explaining and visualizing deep learning, 193-209.
27. Montavon, G., Samek, W., & Müller, K. R. (2018). Methods for interpreting and understanding deep neural networks. Digital signal processing, 73, 1-15.