Text

Description automatically generated

**CS612 AI System Evaluation**

**Project Report**

**Backdoor Detection in Third-Party Neural Networks**

***Group 4***

***Team Member***

Hexu Yang

Sim Wei Xian Andrew

Wang Shunqi

Ye Naing Oo

***17 November 2024***

# Problem Statement

We aim to uncover hidden backdoors in neural networks that have been trained by third parties. Backdoor attacks represent a significant security risk as they enable an attacker to modify a model's output when specific trigger patterns are detected in the input, while it performs as expected in other situations. Identifying these attacks can be challenging, particularly with pre-trained models where the training process is not accessible.

# Objective

The aim of this project is to assess the existence of backdoors in neural networks that have been trained on the MNIST, CIFAR-10, and CIFAR-100 datasets through various detection techniques, including Abstract Interpretation, Universal Adversarial Perturbations (UAP), Semantic Backdoor Detection and Mitigation (SODA), and Neural Cleanse. The evaluation of the project's success will rely on the detection accuracy (correctly distinguishing between backdoored and clean models), the resilience of methods to various types of backdoor triggers, and the computational efficiency of each technique.

# Detection methods

Overview of the four methods that will be covered in this project by individual team member

1. **Abstract Interpretation with Statistical Sampling.** This method merges abstract interpretation methods with statistical sampling to confirm the lack of backdoors in neural networks.
2. **Mode Connectivity Analysis.** This method mainly analyses the loss landscape between a normal model and a backdoored model.
3. **Activation Clustering.** This method analyses the activation layer of model with clustering and checks for outliers.
4. **USB**. Universal Adversarial Perturbations (UAP) are utilized in this context to uncover backdoors by analysing the susceptibility of inputs to be altered into a certain class, a trait commonly associated with backdoored models.
5. **SODA (Semantic Backdoor Detection and Mitigation).** SODA is a method grounded in causality that identifies and alleviates semantic backdoors by examining neuron contributions and modifying them to lessen the impact of the backdoor.
6. **Neural Cleanse.** This approach analyses potential backdoor triggers and identifies irregularities in perturbation sizes to recognize backdoored classes.

# Abstract Interpretation with Statistical Sampling

As outlined in our proposal, we initially explored the method introduced in the paper Verifying Neural Networks Against Backdoor Attacks by Pham et al. (2022) [1]. This method aims to verify the presence or absence of backdoors in a model by assessing its attack success rate. It does so through hypothesis testing, based on the attack success rate observed when sampling inputs and testing them against the model for multiple rounds. To make the verification process computationally feasible, the authors utilized a technique called abstract interpretation. Specifically, they employed the DeepPoly domain to strike a balance between efficiency and accuracy.

However, after a thorough review of the paper and its methodology, we found it unsuitable for our needs. Although the approach is capable of detecting backdoors using only clean data and generating the corresponding trigger pattern, it presents several limitations:

The first one is Inefficiency. The method verifies the existence of backdoors by systematically testing every possible trigger location in sampled inputs using abstract interpretation. While this is feasible for a single target class, applying the process across all potential target classes makes it highly time-consuming. This inefficiency limits its practicality for quickly detecting backdoors in real-world scenarios. The method is better suited for verifying the existence of backdoors, as its title suggests, rather than for fast detection.

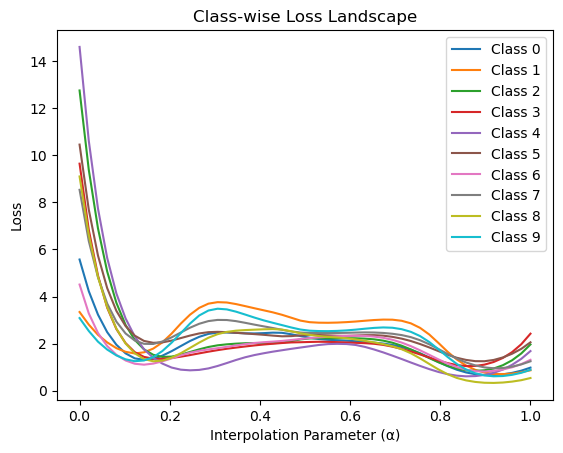
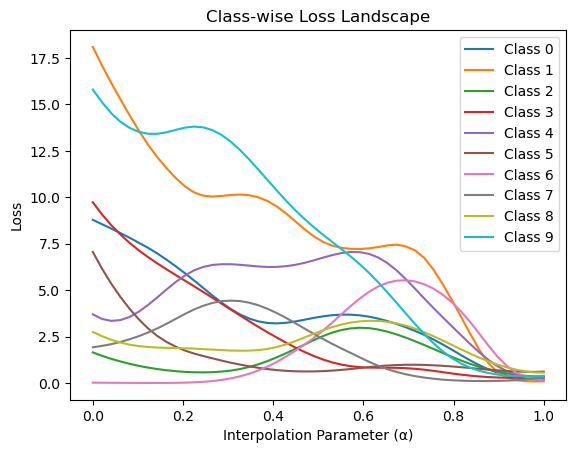
The second is dependency on Specialized Knowledge. The method relies heavily on abstract interpretation techniques, specifically DeepPoly. Incorporating DeepPoly requires a deep understanding of its principles and practical implementation, which we lacked at the outset. Furthermore, the learning curve and the time investment required to integrate DeepPoly into our workflow were underestimated, adding to the complexity.

Given these challenges, we decided to explore alternative methods that are more efficient and straightforward for detecting backdoors. We are currently evaluating other approaches to better align with our project goals and time constraints.

# Mode Connectivity Analysis

Mode connectivity analysis is a technique used to explore the loss landscape of machine learning models by examining the paths between different sets of model parameters, such as those obtained from different training runs or checkpoints, originally introduced by Garipov et al. (2018) [2]. It reveals whether these parameter sets lie in the same low-loss basin by constructing simple paths (e.g., linear or curved) and analyzing the loss along those paths. In the context of backdoor detection, mode connectivity can be leveraged to identify inconsistencies in the loss landscape. When a model is infected with a backdoor, its loss landscape often exhibits irregularities, such as unexpected high-loss regions or disjoint basins, when transitioning between clean and potentially compromised states. By comparing mode connectivity paths between models trained on clean and potentially backdoored data, deviations in the expected smoothness or connectivity of the loss surface can serve as indicators of backdoor presence.

However, this method relies on a key assumption: that the clean and backdoored models exhibit similar overall accuracy and comparable loss values across all classes, except for the target classes. In our case, the sample models provided had poor accuracies and significantly higher losses. When we conducted mode connectivity analysis between a newly trained clean model and the sample models, the resulting loss landscape graphs, as shown in Fig. 1, were highly erratic and unclear. The discrepancies in the loss surfaces could not be definitively attributed to backdoors, as they might simply reflect differences in model performance. Given these challenges, we decided to explore alternative methods for backdoor detection.

Fig. 1 Mode Connectivity Analysis Output Graphs for model1 (MNIST) and model2 (CIFAR10)

# Activation Clustering

Activation clustering is a technique that analyzes the internal activations of a neural network to identify potential anomalies or patterns indicative of backdoor behavior, originally proposed by Chen et al. (2018) [3]. It operates on the premise that clean and backdoored inputs often activate different internal pathways in the model, leading to distinguishable patterns in their activation space. By clustering the activations of the model's hidden layers, activation clustering can separate clean data from data influenced by a backdoor trigger. If a distinct cluster of activations corresponds to the target class but originates from inputs that should not belong to that class, it could signal the presence of a backdoor. This method leverages unsupervised learning techniques, such as k-means, to detect these suspicious clusters without requiring access to the trigger pattern or even labeled data.

The detection method has been implemented in the Adversarial Robustness Toolbox (ART) by Nicolae et al. (2019) [4]. After testing its Python package on the sample models, we identified several limitations. This method is primarily designed to detect backdoored data samples, operating under the assumption that both the input data and model are already poisoned. It requires at least two clusters for each class—one for normal activations and one for backdoored activations. For clustering analysis, ART offers two options: *smaller* and *distance*. The *smaller* option flags the smaller cluster as suspicious, which means it will always flag some data samples to be suspicious. This is unsuitable for our case, as we only have clean data and seek to detect the presence of backdoors in the model. The *distance* option calculates the distances between cluster centers and the centers of all classes, flagging a cluster as suspicious if its center is closer to another class's center than its original class.

Despite its potential for false positives, we used this method to investigate possible backdoors. The results flagged suspicious classes as follows: classes 2, 5, and 7 for model 1; classes 3, 4, and 9 for model 2; and class 4 for model 3. No suspicious classes were identified for models 4 and 5.

# Universal Soldier

**Work Book:** [**Colab work book**](https://colab.research.google.com/drive/1b7mGZcTgp2Fc6BhFw34azBDoH1-Ycjq4#scrollTo=gMFIBRL1wBqX)

**Background and Motivation**

Traditional backdoor detection methods often rely on reverse engineering techniques to reconstruct potential triggers. However, these approaches face significant limitations, particularly when dealing with complex non-patch-wise attacks or when distinguishing between legitimate class features and malicious triggers. The need for a more robust detection mechanism has led to the development of USB, originally introduced by Xu et al. (2023) [5], which capitalizes on the fundamental properties of both backdoor attacks and universal adversarial perturbations.

## Targeted UAP Generation

The first component involves generating targeted UAPs for each possible class in the model's output space. This process uses a modified version of the DeepFool algorithm that has been adapted for targeted attacks. For each class c, we iteratively compute a perturbation v that satisfies:

1. Initialization: Begin with zero perturbation v = 0
2. Iterative Optimization: For each clean input x:
   1. If f(x + v) ≠ c, compute minimal perturbation dr using DeepFool
   2. Update v = v + dr
   3. Project v onto l∞-ball: v = clip(v, -ε, ε)
3. Convergence: Continue until desired success rate or maximum iterations reached

The key innovation here is that the generation process is designed to find the smallest possible perturbation that can consistently fool the model into predicting a specific target class.

## Norm Analysis and Detection

The second component involves analyzing the L1 norms of the generated UAPs. For each class i, we compute:

* ||vi||1: The L1 norm of the UAP targeting class i
* μ: Mean of all UAP norms
* σ: Standard deviation of all UAP norms

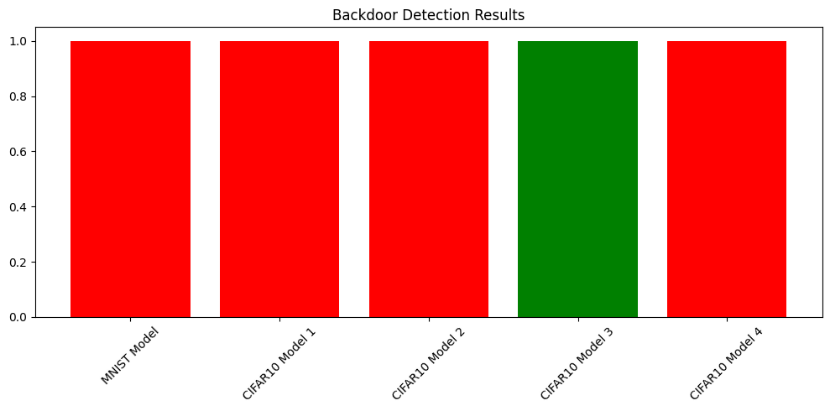
The detection criterion is based on the statistical deviation of the minimum norm from the mean:

* A class c is identified as a potential backdoor target if: (μ - ||vc||1) / σ > threshold where threshold is a tunable parameter (typically set to 0.5)

This analysis leverages the observation that backdoor targets require significantly smaller perturbations compared to clean classes, creating a detectable statistical anomaly.

The methodology's effectiveness stems from its ability to exploit the fundamental relationship between backdoor triggers and universal adversarial perturbations, providing a robust and efficient detection mechanism that outperforms traditional approaches in both accuracy and computational efficiency.

## Experimental Results



The results from our Universal Soldier for Backdoor Detection (USB) implementation reveal significant findings across the tested models. Out of the five models evaluated, four were identified as backdoored (indicated in red), while only one model (CIFAR10 Model 3, shown in green) was detected as clean. Specifically, both the MNIST model and three out of four CIFAR10 models (Models 1, 2, and 4) showed clear signs of backdoor implementation. This high detection rate of 80% suggests widespread backdoor injection in the model set, with only CIFAR10 Model 3 maintaining its integrity. The consistency of detection across different architectures (MNIST and CIFAR10) demonstrates the robustness of our USB detection method in identifying backdoors regardless of the underlying model architecture or dataset type.

# Semantic Backdoor Detection with SODA

Semantic backdoors are a type of backdoor attack in neural networks that exploit natural or semantic features rather than synthetic, artificial patterns. Unlike conventional backdoors, which use specific triggers like visible patterns or pixels to manipulate model predictions, semantic backdoors rely on natural characteristics innate to the dataset. Hence, semantic backdoors are harder to detect, as they do not rely on synthetic triggers or modified inputs during inference. Semantic backdoors can achieve a high success rate in targeted misclassifications with minimal effect on the model’s overall accuracy on clean data through its focus on rare semantic features like a specific color in objects or unique backgrounds (Sun et al., 2024 [6]).

SODA (Semantic Backdoor Detection and Mitigation) is an analysis and mitigation approach designed to detect and mitigate semantic backdoors in neural networks. It uses causality analysis to identify neurons responsible for semantic backdoors by analysing how hidden neurons contribute to predictions using lightweight causal intervention techniques (Sun et al., 2024). Causality analysis helps in detecting unusual neuron behavior and flagging backdoor attacks. SODA compares the causal contributions of neurons for clean and potentially infected samples. Mitigation is done by adjusting the contribution of the responsible neurons, thereby removing the influence of the backdoor on the prediction without retraining the entire model (Sun et al., 2024).

This project’s focus pertaining to SODA is only on the detection of semantic backdoors and not on mitigation. Hence, the mitigation following detection of possible semantic backdoors was not implemented but will be discussed in theory.

The methodology for the implementation of SODA was as follows:

1. Model and dataset preparation - loading of sample models:
2. A dataset of clean samples was prepared and triggers added (inputs with specific natural features that might activate the backdoor). The trigger was applied to a corner patch (Sun et al., 2024).
3. Capture of activations through layer hooking
4. Register hooks on each target layer (e.g., fc1, fc2, fc3, fc4 for fully connected layers) to capture neuron activations. The hook\_fn function handles this by storing activations for each layer during the forward pass (Sun et al., 2024).

1. Pass Inputs:
2. First, pass clean inputs through the model and record activations for each layer.
3. Next, pass triggered inputs through the model and record activations, distinguishing between clean and triggered activations (Sun et al., 2024).
4. Statistical analysis for backdoor detection
5. Identify neurons that behave unusually between clean and triggered conditions using two key metrics: Pearson Correlation Coefficient (PCC) and Median Absolute Deviation (MAD).
6. PCC - For each neuron in each target layer, the PCC between activations from clean and triggered inputs were calculated. Neurons with low PCC values are flagged as potentially significant, meaning that they had different behavior between clean and triggered conditions.
7. MAD - For each neuron flagged by PCC, the mean difference in activations between clean and triggered samples was computed across these activation differences (Sun et al., 2024).
8. A threshold was applied (set as the median difference + 1.5 × MAD) to identify neurons with activation shifts that exceed normal variation. Neurons with mean differences above the MAD threshold are flagged as significant, and likely to be behaving under triggered conditions.
9. Identify Significant Neurons by combining results of PCC and MAD analysis and create a list of neurons that meet both criteria (low PCC and high MAD). These neurons are likely to be under the backdoor effect.

1. For each layer, display the indices of significant neurons potentially related to the backdoor. Histograms of activation differences are generated to visualize the distribution and identify the threshold.

Based on the five sample models provided (1 MNIST and 4 CIFAR-10), the SODA implementation identified suspicious neurons in all models (refer to Annex A) (Sun et al., 2024).

Having detected the suspicious neurons possibly responsible for backdoor behaviour, mitigation can then be carried out. This involves fine-tuning the model by targeting the specific neurons responsible for backdoor behavior. The process begins by generating infected samples that simulate the backdoor effect through reverse-engineering, and gradually modifying clean samples to match the target class. Significant neurons in each layer, identified through prior analysis, are then fine-tuned using a specialized loss function. This function encourages correct classification of clean samples and reduces classification to the target label for infected samples. Only the weights of the significant neurons are adjusted, preserving the model’s overall parameters while minimising backdoor influence (Sun et al., 2024). The fine-tuning process involves training the model for a few epochs using both clean and infected samples, ensuring the model retains its performance on normal data while mitigating the backdoor’s effects. Through optimisation through backpropagation using Stochastic Gradient Descent (SGD), the weights for the responsible neurons' weights are adjusted over a few epochs, ensuring the semantic backdoor is mitigated while maintaining the model's accuracy on clean data (Sun et al., 2024).

# Neural Cleanse

This method implements an optimization algorithm to generate minimal trigger patterns and subsequently evaluate the existence of backdoors based on anomaly scores calculated for each target class. Originally introduced by Wang et al. (2019) [7].

## Code Summary

**Workbook**: [colab work book link](https://colab.research.google.com/drive/1UdTloEuso_-hsPUL-QGdlRDSZfsqlMUP?usp=sharing)

**Model resources and test samples**: <https://github.com/noye09/cs612_SMU_g4.git>

The main code block performs backdoor detection on a selected model multiple times to ensure reliable results. It initializes a model, call the runs the detection to identify backdoors, and saves the anomaly scores and any detected backdoor classes. Finally, it compiles these results into a CSV file for further analysis.

## Methodology

*Trigger Generation and Optimization:* For each target class, the TriggerOptimizer takes a sample image from the dataset and generates an initial mask and pattern. The optimization process iteratively refines the mask and pattern to maximize the model's confidence in predicting the target class while minimizing the mask's size. This optimization is driven by a combination of cross-entropy loss (to force misclassification) and a regularization term (to encourage sparsity in the mask). The mask highlights the regions to be modified, and the pattern defines the modifications applied to these regions.

*Anomaly Score Calculation:* Once masks are generated for all target classes, their sizes are analyzed to compute anomaly scores. The anomaly score for each class is calculated by comparing the mask size to the mean and standard deviation of mask sizes across all classes. Classes with significantly smaller mask sizes (e.g., more than two standard deviations below the mean) are considered anomalous, as they indicate that minimal modification is required to achieve misclassification.

*Backdoor Detection:* Classes with highly negative anomaly scores are flagged as potentially compromised. A backdoor is suspected when a small mask can induce consistent misclassification, suggesting that the model has a hidden vulnerability or has been intentionally compromised. The detection process is repeated multiple times to ensure robustness and consistency of the results, reducing the influence of randomness. The final results, including anomaly scores and flagged target classes, are compiled for further analysis to identify any potential backdoor triggers effectively.

## Experimental Results

## A graph of different colored bars Description automatically generated

A screenshot of a graph

Description automatically generated

Two known backdoor models, model6 (MNIST) and model7 (CIFAR-10), were tested. For MNIST, the backdoor was detected 8 out of 10 times, achieving an 80% success rate. For CIFAR-10, detection was successful 6 out of 10 times, resulting in a 60% success rate.

The method was then applied to models 1 through 5. Model 3 showed a 100% success rate in backdoor detection. Models 1 and 2 achieved a 60% success rate but had a few false negatives. For models 4 and 5, detection was unsuccessful, with no backdoor identified. Anomaly scores for models 4 and 5, visualized using heat maps, showed no discernible trends or patterns, further indicating unsuccessful detection

# Detail Summary

Our originally proposed method, **Abstract Interpretation with Statistical Sampling**, was excluded because it is computationally inefficient and doesn’t align with the project’s focus on detection. While it is not practical for general backdoor detection, it remains valuable for verifying robustness in safety-critical systems where formal guarantees are needed. The table below summarizes the backdoor detection results across four methods for five unknown models.

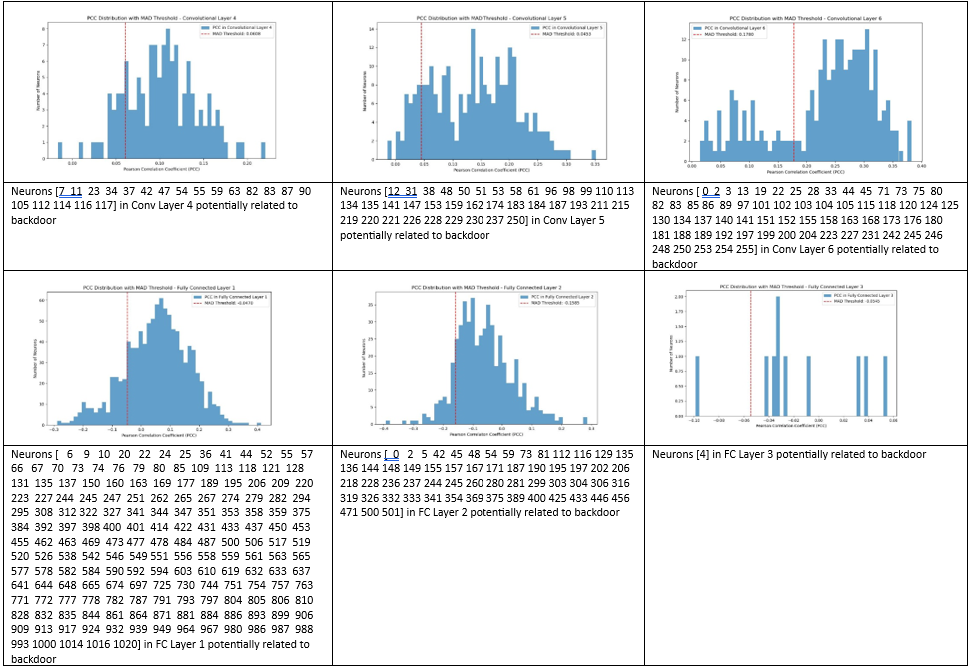
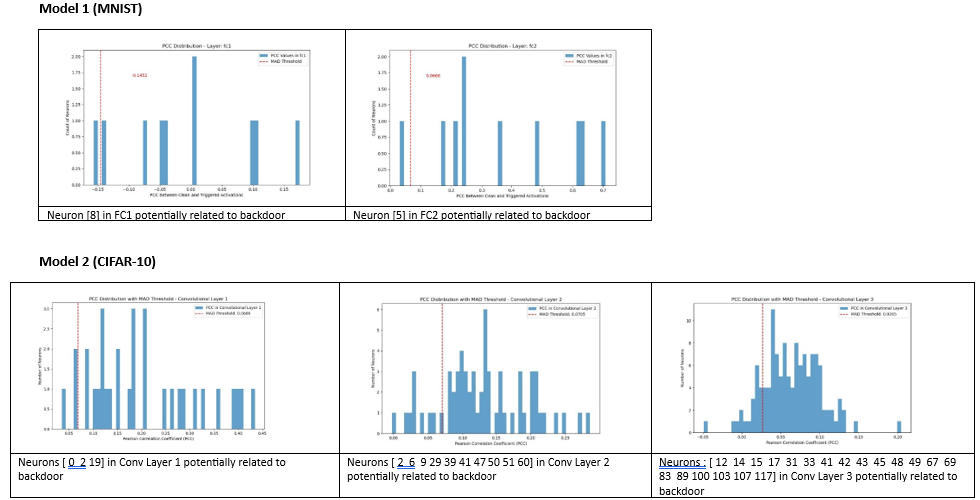
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Activation Clustering | USB | SODA | Neural Cleanse | Comments |
| Model 1 | Detected | Detected | Detected | Detected | Detect on all methods. |
| Model 2 | Detected | Detected | Detected | Detected | Detect on all methods. |
| Model 3 | Detected | Detected | Detected | Detected | Detect on all methods. |
| Model 4 | Not Detected | Not Detected | Detected | Not Detected | Only SODA detected the backdoor, suggesting it is a semantic backdoor that other methods missed. |
| Model 5 | Not Detected | Detected | Detected | Not Detected | USB and SODA detected the backdoor, but Activation Clustering and Neural Cleanse failed. Likely due to non-pixel or subtle triggers. |

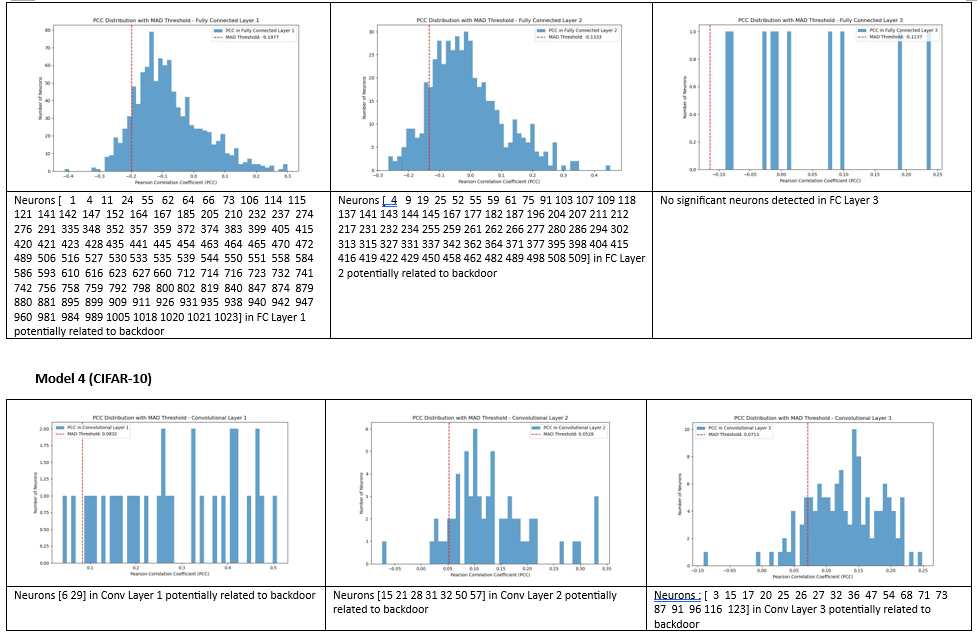
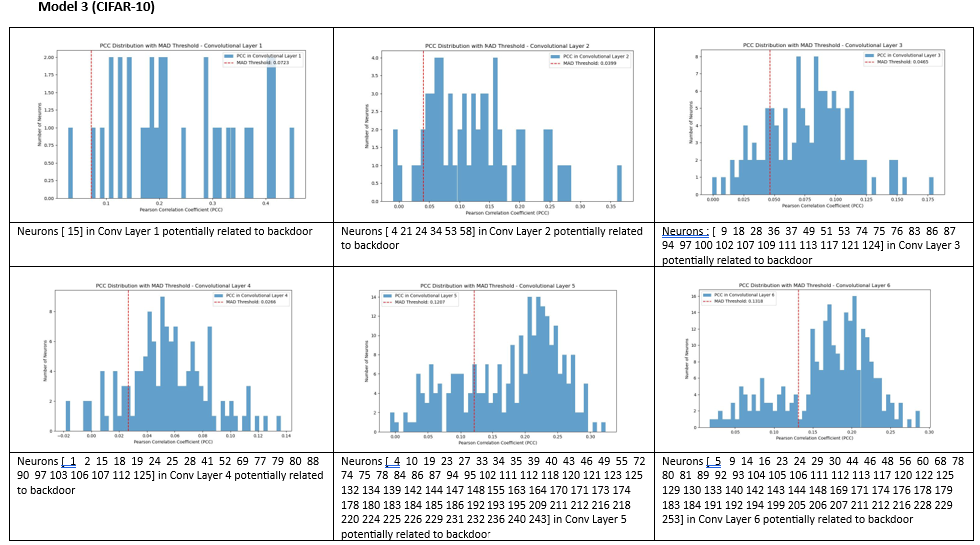
# Conclusion

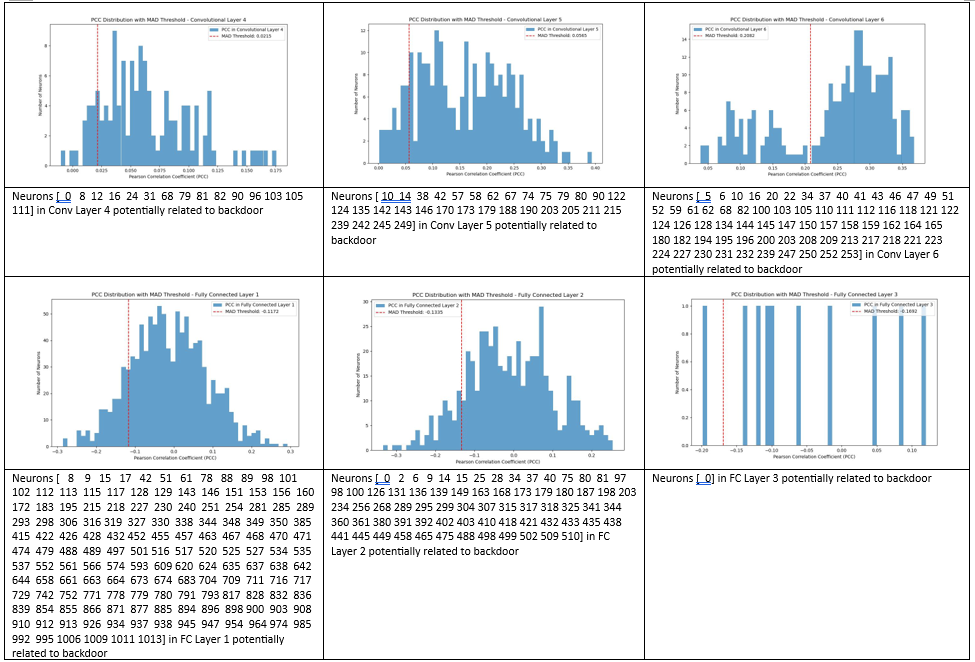
The results highlight the importance of using multiple methods for effective backdoor detection. Activation Clustering and Neural Cleanse performed well on simpler backdoors (Models 1–3) but struggled with more complex or subtle triggers in Models 4 and 5. In contrast, USB and SODA showed greater flexibility, successfully detecting backdoors in these challenging cases.

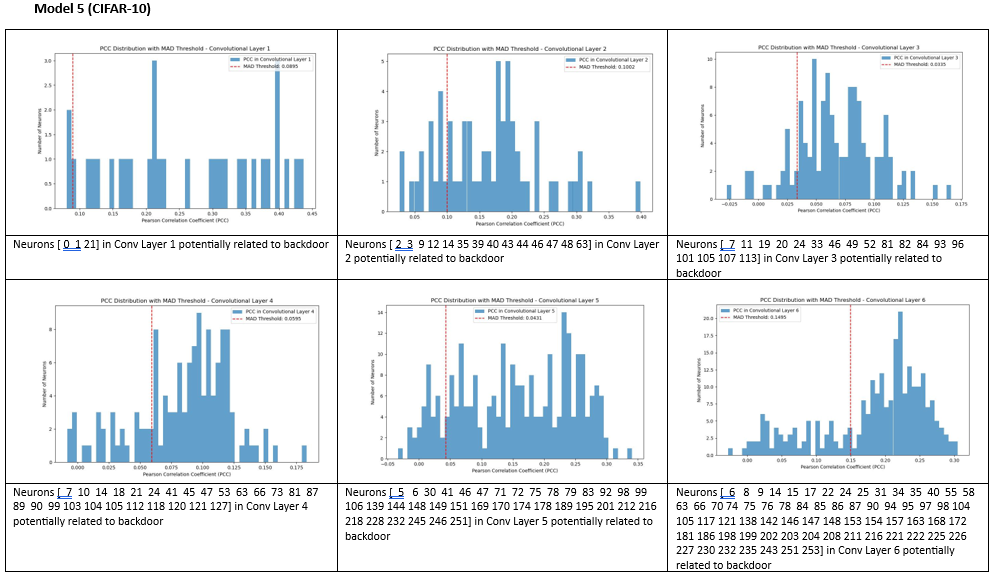
SODA excelled in identifying semantic backdoors, which other methods missed, while USB proved to be a reliable all-purpose tool. The exclusion of Abstract Interpretation with Statistical Sampling was justified due to its inefficiency and lack of alignment with the project’s detection goals. This study highlights the need to combine methods like USB and SODA for comprehensive and reliable backdoor detection.

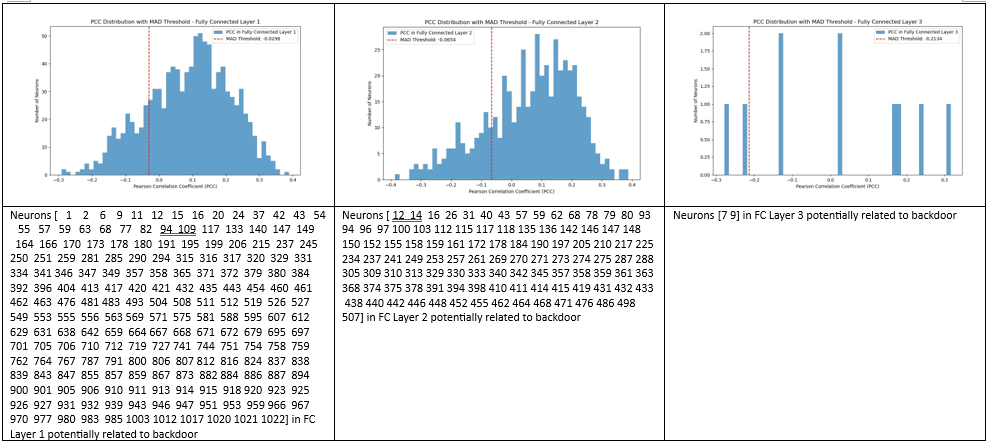
**ANNEX A**











# References:

1. Pham, L. H., & Sun, J. (2022). Verifying Neural Networks Against Backdoor Attacks. In S. Shoham & Y. Vizel (Eds.), *Computer Aided Verification. CAV 2022. Lecture Notes in Computer Science* (Vol. 13371). Springer, Cham. <https://doi.org/10.1007/978-3-031-13185-1_9>
2. Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D., & Wilson, A. G. (2018). Loss surfaces, mode connectivity, and fast ensembling of DNNs. *Advances in Neural Information Processing Systems (NeurIPS)*, 31.
3. Chen, B., Liu, W., Li, B., Lu, K., & Song, D. (2018). Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering. *arXiv preprint arXiv:1811.03728*.
4. Nicolae, M.-I., Sinn, M., Tran, M. N., Buesser, B., Rawat, A., Wistuba, M., Zantedeschi, V., Baracaldo, N., Chen, B., Ludwig, H., Molloy, I. M., & Edwards, B. (2018). Adversarial Robustness Toolbox v1.0.0. *arXiv preprint arXiv:1807.01069*.
5. Xu, X., Ersoy, O., & Picek, S. (2023). Universal Soldier: Using Universal Adversarial Perturbations for Detecting Backdoor Attacks. \*arXiv preprint arXiv:2302.00747\*.
6. Sun, B., Koh, W., Sun, J., & Shi, J. (2024). Neural network semantic backdoor detection and mitigation: A causality-based approach. Singapore Management University and Huawei Singapore.
7. 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. In *2019 IEEE Symposium on Security and Privacy (SP)* (pp. 707-723). IEEE. <https://doi.org/10.1109/SP.2019.00065>