University of Idaho

Department of Computer Science

# CS 487/587 Adversarial Machine Learning

*Dr. Alex Vakanski*

# Lecture 9

# Defenses against Poisoning Attacks

# Lecture Outline

- Poisoning defenses
  - Blind backdoor removal
  - Offline inspection
  - Online inspection
  - Post backdoor removal
- Neural Cleanse
- SentiNet

# Defenses Against Poisoning Attacks

*Poisoning Defenses*

- ***Defense strategies*** against poisoning adversarial attacks were categorized in Gao et al. (2020) into:
  - **Blind backdoor removal**
    - The user is not sure whether the data or the model were poisoned
    - The user applies defense methods for removing or suppressing backdoors in input samples
    - Or, the user cleans the model to reduce the impact of poisoning
  - **Offline inspection**
    - Defense methods are applied before the model is deployed
    - If the user has access to the data, the user removes the poisoned samples
    - If the poisoned data is not available, the user cleans the backdoored model
  - **Online inspection**
    - Defense methods are applied to monitor the performance during run-time
    - The user either inspects the incoming inputs to remove poisoned data
    - Or, the user evaluates the model to determine abnormal behavior
  - **Post backdoor removal**
    - If any of the above defenses have identified backdoored sample, these defense methods are applied to remove the backdoor

# Blind Backdoor Removal Defenses

*Blind Backdoor Removal Defenses*

- ***Blind backdoor removal***
  - This defense approach does not differentiate the backdoored model from a clean model (hence, it blindly applies backdoor removal)
  - The goal is to remove or suppress the effect of a potential backdoor while achieving high accuracy on clean inputs
  - The defense can be performed either offline or online
- Fine-pruning defense
  - Liu (2018) Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks
  - Remove potential backdoor by pruning the neurons in DNN with the smallest contribution to the classification task
    - The main assumption is that different neurons are activated by clean and trigger inputs
  - Step 1: sort all neurons based on the activation values on clean inputs (e.g., from a test set) and remove those neurons with the smallest activation values
  - Step 2: fine-tune the modified model with the pruned neurons
  - Limitation: reduced accuracy on clean inputs

# Blind Backdoor Removal Defenses



- Suppression defense
  - Sarkar (2020) Backdoor Suppression in Neural Networks using Input Fuzzing and Majority Voting
  - The goal is to clean the triggers in poisoned images via fuzzing
  - Step 1: create many replicas of each image (without knowing if they are clean or backdoored images) by adding noise
    - The noise level is experimentally determined for a dataset
  - Step 2: train a DNN model using all fuzzied image replicas, and calculate a final prediction based on majority voting of the predictions on all perturbed replicas of an input image
  - The defense achieved a success rate of 90% on backdoored images in MNIST, and 50% on backdoored images in CIFAR-10
  - Limitation: reduced accuracy on clean inputs, low success rate on CIFAR-10 images
- Note:
  - Blind backdoor removal defense does not distinguish a backdoored model from a clean model, or trigger inputs from clean inputs
  - It usually reduces the accuracy on clean inputs

# Offline Inspection Defenses

*Offline Inspection Defenses*

- ***Offline inspection defense***
  - The defense is applied before the model is deployed in production
  - These defense strategy can be based on data inspection or model inspection
- *Offline data inspection defense*
  - It is assumed that the poisoned data is available to the defender
- Spectral signature defense
  - B. Tran (2018) Spectral Signatures in Backdoor Attacks
  - Remove poisoned data samples based on outlier detection approach
  - Step 1: train a DNN model to classify the available data that contains both clean and poisoned samples
  - Step 2: for each class, calculate SVD on the logit values of the model, and remove all input samples that are outliers (i.e., have singular values greater than a threshold)
  - Step 3: retrain the model with the remaining samples
  - Limitation: may remove clean samples, requires some knowledge to establish the threshold value for outlier detection

# Offline Inspection Defenses

*Offline Inspection Defenses*

- Gradient clustering defense, activation clustering defense
  - Chan (2019) Poison as a Cure: Detecting & Neutralizing Variable-sized Backdoor Attacks in Deep Neural Networks
  - Chen (2018) Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering
  - The assumption by this defense approach is that trigger inputs will produce either large gradients at the trigger location, or large logit activation values
  - Step 1: apply a clustering algorithm (e.g., *k*-mean clustering) to separate clean inputs from trigger inputs, based on the gradient or activation values
  - Step 2: remove or relabel the trigger inputs, and retrain the model

# Offline Inspection Defenses

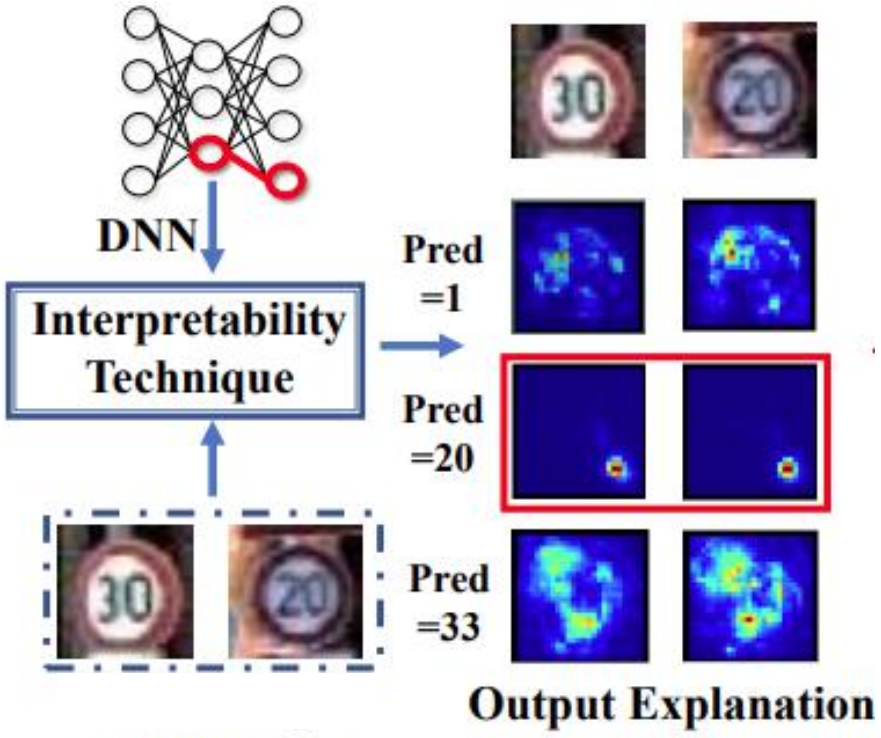
*Offline Inspection Defenses*

- *Offline model inspection defense*
  - The defender has access to the backdoored model
  - This defense approach does not assume that poisoned data is available to the defender
    - Therefore, the assumptions are more realistic
- Neural cleanse
  - Wang et al. (2019) Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks
  - The defense iterates though all labels to determine if any label requires much smaller perturbation to be applied to the inputs to achieve misclassification
    - An optimization algorithms is applied to reverse engineer the trigger
    - The reverse-engineered trigger is used to retrain the model, and remove the backdoor
  - Limitations: high computational costs for models with large number of classes, the reverse-engineered trigger is not always consistent with the original trigger
  - This defense method is explained in more details in this lecture

# NeuronInspect

*Offline Inspection Defenses*

- NeuronInspect
  - Huang (2019) NeuronInspect: Detecting Backdoors in Neural Networks via Output Explanations
  - Applies interpretability approaches to create heatmaps for the target class and non-target classes
    - The heatmaps for the target class differ significantly for clean and trigger inputs
    - In the figure, the target class is 20 (third row), and the trigger is noticeable in the heatmaps
  - Outlier detection is then applied on the produced heatmaps as a defense strategy

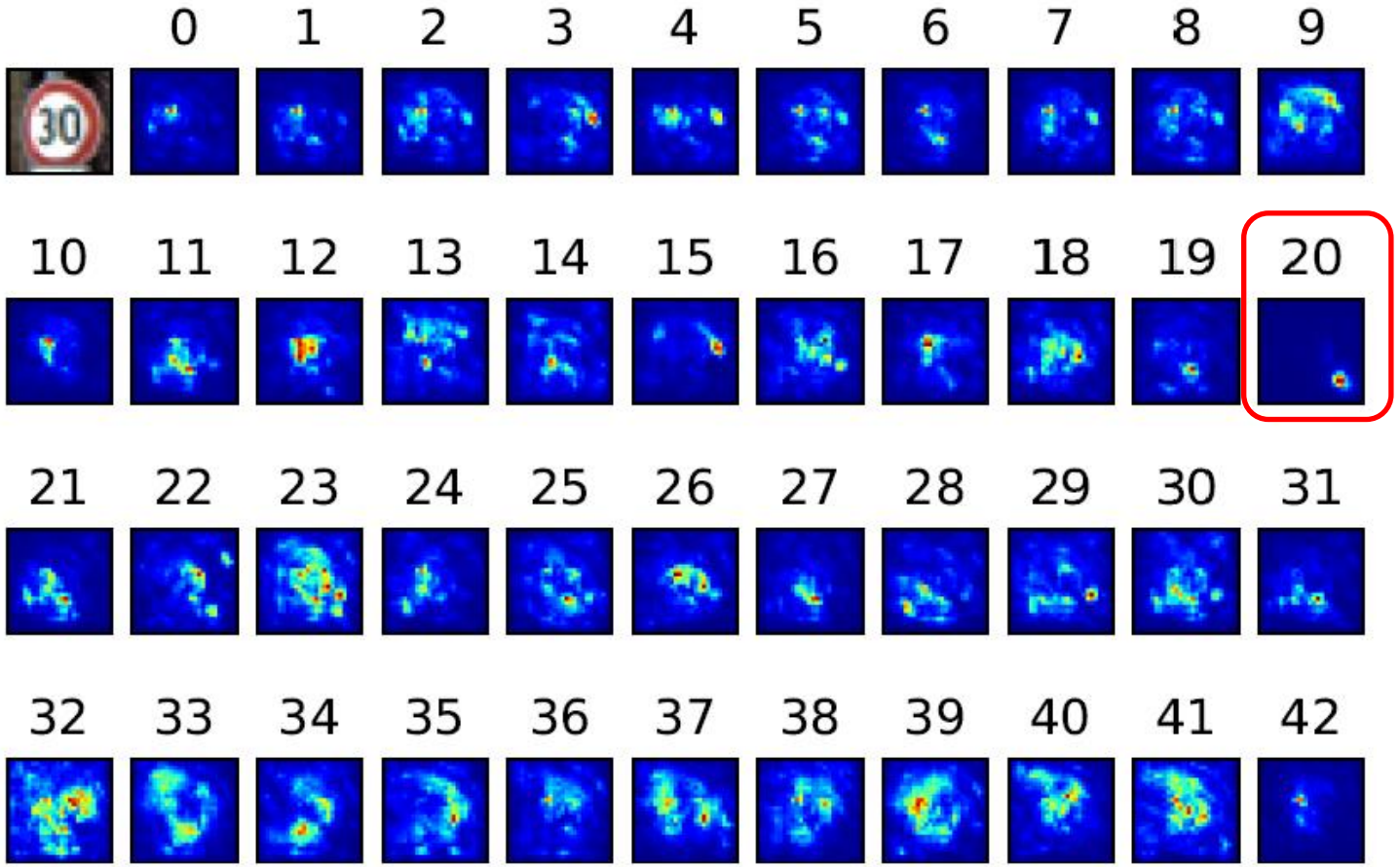



- Note:
  - Offline model inspection requires high computational resources and time
  - Most offline defense methods cannot deal with large-sized triggers

# NeuronInspect

*Offline Inspection Defenses*

- NeuronInspect (cont'd)
  - Explanation heatmaps for the Speed Limit 30 sign image, for all 43 classes of traffic signs
    - The heatmap for label 20 is an outlier, in comparison to heatmaps for all other labels



# Online Inspection Defenses

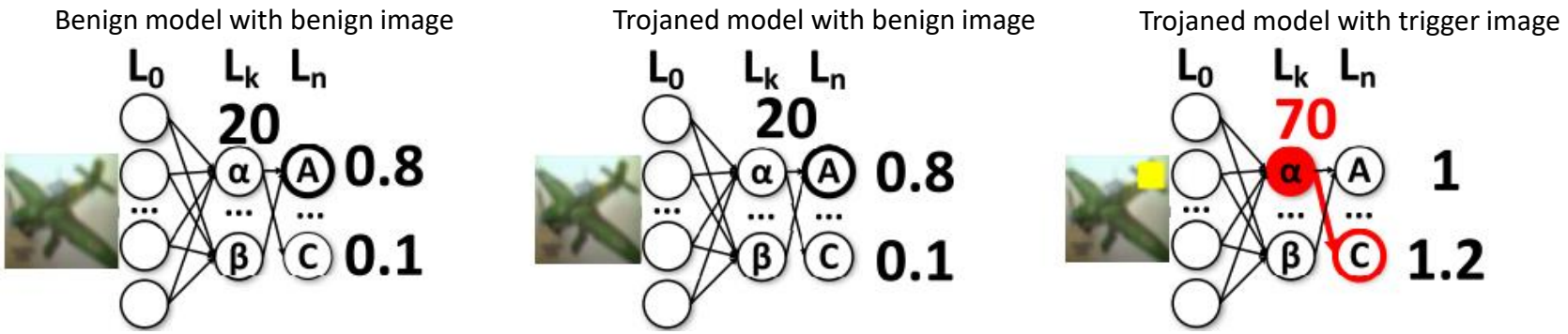
*Online Inspection Defenses*

- ***Online inspection defense***
  - This defense is applied to monitor the behavior of input data or a model during run-time
- *Online data inspection defense*
  - Most defense methods apply some form of anomaly detection to check if the inputs contain a trigger
- STRIP defense
  - Gao (2019) STRIP: A Defense against Trojan Attacks on Deep Neural Networks
  - Step 1: apply random noise to create many replicas of an input image
  - Step 2: use the entropy of the replicas for anomaly detection
    - Replicas of trigger images have high entropy (the predicted class is more uniform), whereas clean images have low entropy (the predicted class is more random)
- SentiNet defense
  - It is presented later in this lecture
  - It applies explainability approach to discover regions that may contain a trigger, and uses these regions to monitor incoming inputs

# Online Inspection Defenses

*Online Inspection Defenses*

- *Online model inspection defense*
  - Apply anomaly detection to identify abnormal behavior of a backdoored model
- ABS defense
  - Liu (2019) ABS: Scanning Neural Networks for Back-doors by Artificial Brain Stimulation
  - Scan the DNN to identify individual neurons in the model with abnormally high activation values
    - If there is a large change in the neuron activation value for a specific label regardless of the provided input samples, then the model is potentially poisoned
    - Apply outlier detection to identify Trojaned models
  - Limitation: the target label needs to be activated by only one neuron, instead by a group of neurons



# Online Inspection Defenses

*Online Inspection Defenses*

- NIC defense
  - Ma (2019) NIC: Detecting Adversarial Samples with Neural Network Invariant Checking
  - Inspect the distribution of the activations by different layers of DNN to detect abnormal behavior
    - Determine if the flow of the activations is abnormal for some labels
  - This approach requires to learn the distributions of the activations in an offline step
- Note:
  - Online inspection approaches typically require some preparations to be performed offline (e.g., determining a threshold to distinguish clean from trigger inputs)
  - Also, these approaches can result in latency and can be less suitable for real-time applications (e.g., self-driving vehicles)

# Post Backdoor Removal Defenses

*Post Backdoor Removal Defenses*

- ***Post backdoor removal defense***
  - Includes techniques to remove the backdoor, after it is identified by the previous defense approaches
- If the defender has access to poisoned data, they can remove trigger inputs, and retrain the model using only clean inputs
- Another approach is to change the labels of the poisoned inputs with triggers to the correct labels, and then retrain the model
  - For this defense approach, it is required to reverse-engineer the trigger

# Neural Cleanse

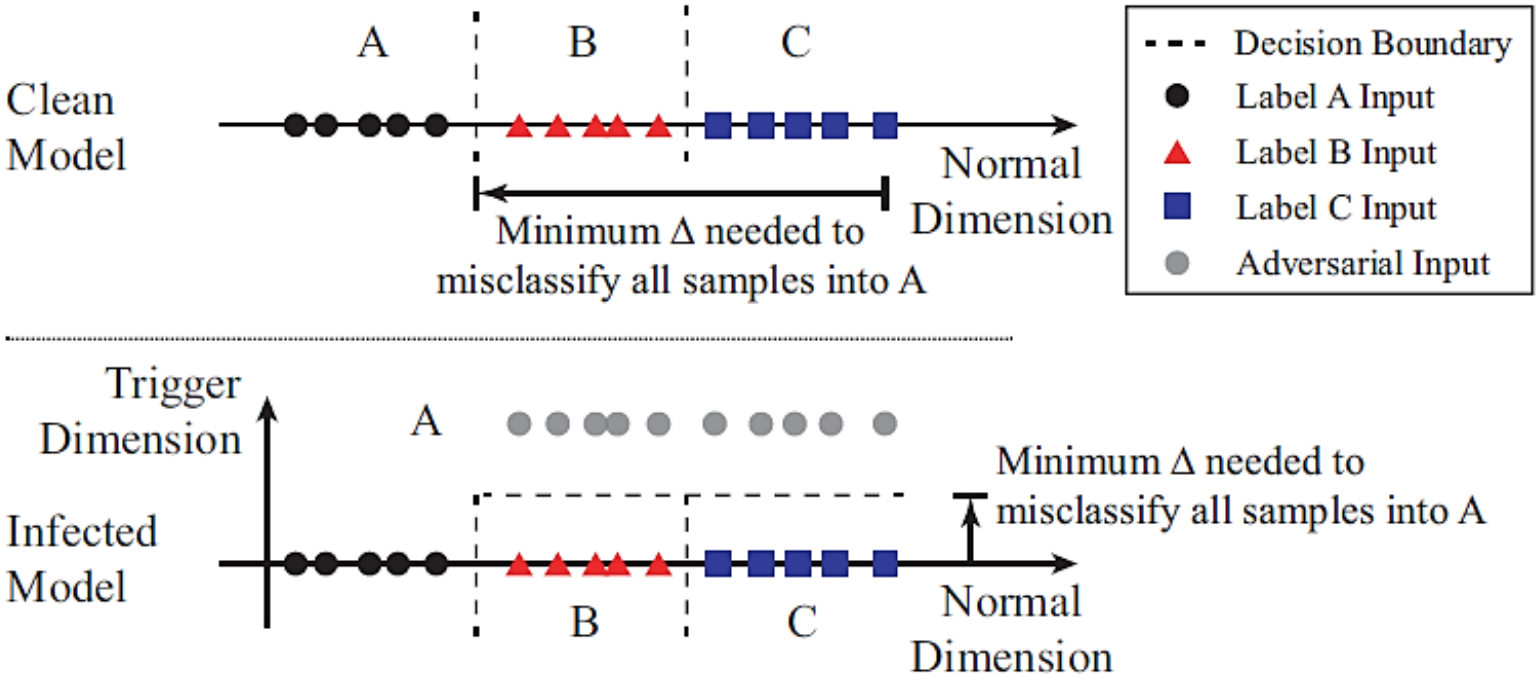
*Neural Cleanse*

- Wang et al. (2019) Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks
- ***Neural Cleanse*** introduces methods for detection and mitigation of backdoor attacks
  - The detection method identifies backdoored models and reconstructs possible triggers
  - The mitigation techniques include input filters, neuron pruning, and unlearning
- The defense methods were validated against two poisoning attacks:
  - BadNet backdoor attack (input samples with backdoors)
  - Trojaned network (altered neurons)
- Threat model assumptions
  - The defender has access to the trained DNN and a set of correctly labeled samples

# Defense Intuition and Overview

*Neural Cleanse*

- Backdoor triggers produce a classification result to a target label A regardless of the ground-truth label the input belongs to
  - Backdoors create "shortcuts" for adversarial inputs to cross the decision boundaries
- Defense strategy: detect the "shortcuts" by measuring the minimum perturbation necessary to change all inputs from one label to a target label



# Detecting Backdoors Approach

*Neural Cleanse*

- Step 1
  - Consider a given label to be a target label of a backdoor attack
  - Apply an optimization algorithm to calculate the "minimal" perturbation required to misclassify all samples from other labels to this target label
- Step 2
  - Repeat Step 1 for each output label in the model
  - This results in multiple potential triggers for a target model
- Step 3
  - Measure the size of each potential trigger from Step 2
  - Run an outlier detection algorithm to detect if any trigger is significantly smaller than other triggers
    - An outlier represents a real trigger with a corresponding target label of the backdoor attack

# Reverse Engineering Triggers

*Neural Cleanse*

- To solve Step 1 the authors applied the following optimization algorithm

$$\min_{m,\Delta} \quad \ell(y_t, f(A(x, m, \Delta))) + \lambda \cdot |m|$$

- Notation:
  - $x$ is the input
  - $A(x, m, \Delta)$ is a backdoored image with a trigger $\Delta$ and mask $m$
    - The mask $m$ has 0 values for the pixels not covering the trigger
  - $\ell(y_t, A(x, m, \Delta))$ is the loss of the model for classifying backdoored image into class $y_t$
- The first term of the optimization algorithm above finds a trigger $\Delta$ and mask $m$ that classify backdoored images into the target class $y_t$
- The second term ensures that the L1 norm of the mask, $|m|$, is small, i.e., the mask is applied only to a small portion of the image
  - $\lambda$ is the weight coefficient for the second term
- Output of the algorithm is a reverse engineered trigger $\Delta$

# Considered Applications

*Neural Cleanse*

- The authors implemented and validated Neural Cleanse on the following applications:
  - Handwritten digit recognition – MNIST dataset (10 digits)
  - Traffic sign recognition – GTSRB dataset (43 classes)
  - Facial recognition – YouTube Face dataset (1,283 people)
    - Contains 375 K training images extracted from YouTube videos
    - Large number of classes: 1,283
  - Facial recognition – PubFig dataset (65 people)
    - Contains 5 K dataset (small dataset)
    - The DNN classifier uses transfer learning from a large pretrained model

# Considered Applications

*Neural Cleanse*

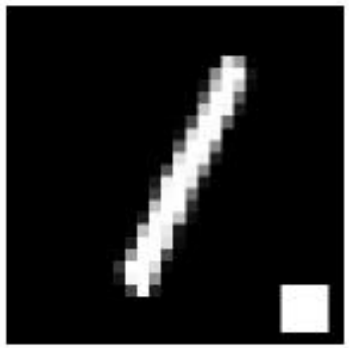
- Attack success rate and classification accuracy of the poisoned models and clean models
  - Attack success rate is the percentage of backdoored images assigned the target label
    - All models achieved over 97% attack success rate
  - The classification accuracy of the infected models are slightly reduced on clean images

| Task | Infected Model | | Clean Model Classification Accuracy |
|---|---|---|---|
| | Attack Success Rate | Classification Accuracy | |
| Hand-written Digit Recognition (MNIST) | 99.90% | 98.54% | 98.88% |
| Traffic Sign Recognition (GTSRB) | 97.40% | 96.51% | 96.83% |
| Face Recognition (YouTube Face) | 97.20% | 97.50% | 98.14% |
| Face Recognition w/ Transfer Learning (PubFig) | 97.03% | 95.69% | 98.31% |

# Experimental Validation

*Neural Cleanse*

- Validation of Neural Cleanse defense against the BadNet poisoning attack is presented next
  - The backdoor trigger is a white square located in the bottom right corner of the images
  - The size of the trigger is limited to about 1% of the image size (e.g., 4x4 pixels for MNIST)
  - The trigger is inserted into clean images of the training set with the labels changed to the target class
    - This causes the trained model to associate the backdoored images with the target label
  - The ratio of poisoned samples to clean samples is between 10% and 20%
    - This ratio is varied in order to achieve a high attack success rate of over 95%



(a) MNIST

(b) GTSRB

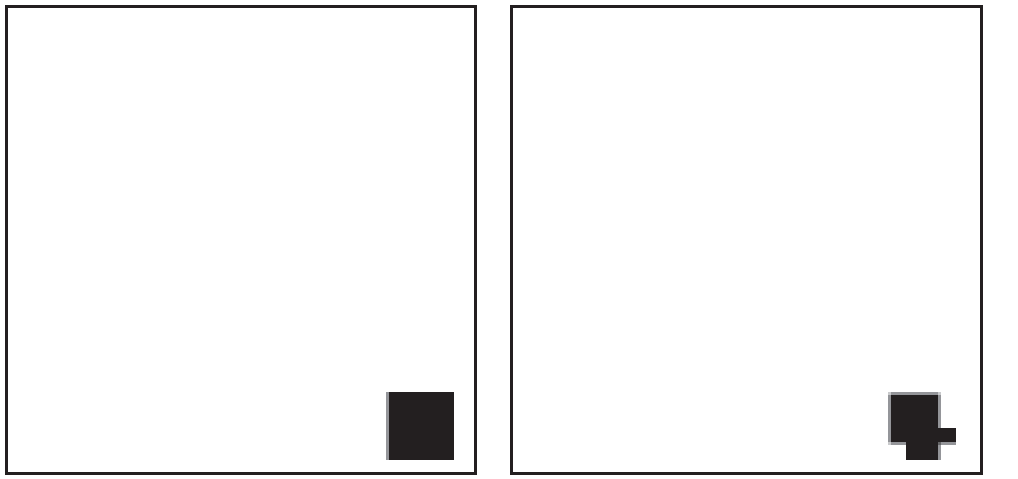
(c) YouTube Face

(d) PubFig

# Reverse Engineered Trigger

*Neural Cleanse*

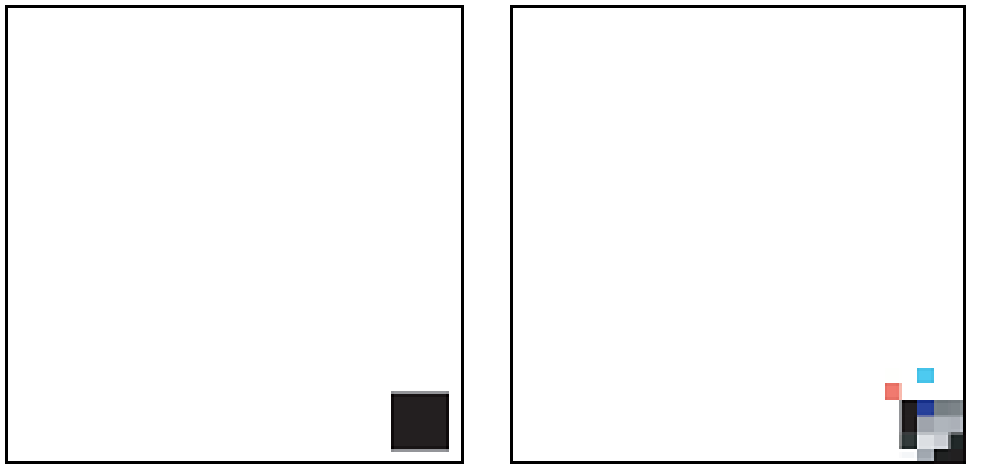
- Original and reverse engineered trigger for MNIST digit classification are shown in the figure below
  - The reverse engineered trigger is similar to the original trigger and shows up at the same location as the original trigger
  - L1 norm of the reversed trigger is similar to the original trigger



# Reverse Engineered Trigger

*Neural Cleanse*

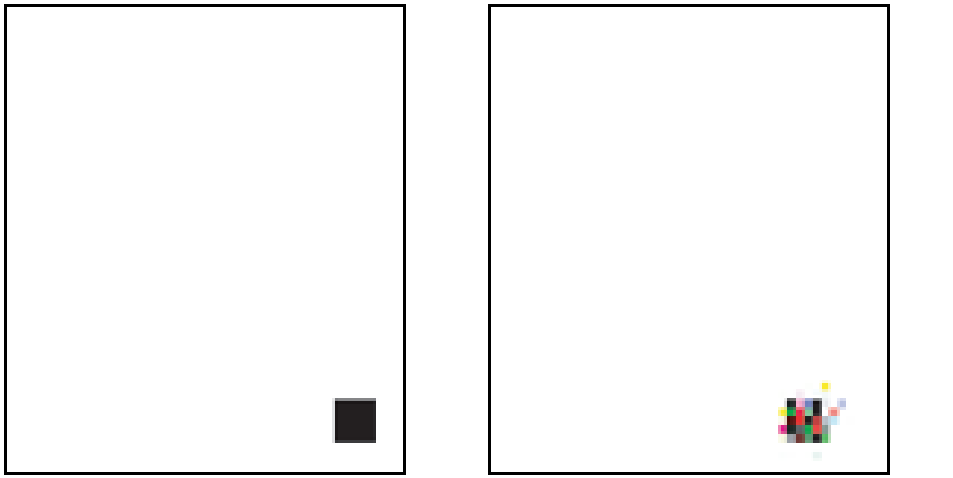
- Original vs reverse engineered trigger for traffic sign recognition on the GTSRB dataset



Original Trigger (L1 norm = 16)

Reversed Trigger ($m \cdot \Delta$) (L1 norm = 14.71)

- Original vs reverse engineered trigger for face recognition on the YouTube Face dataset



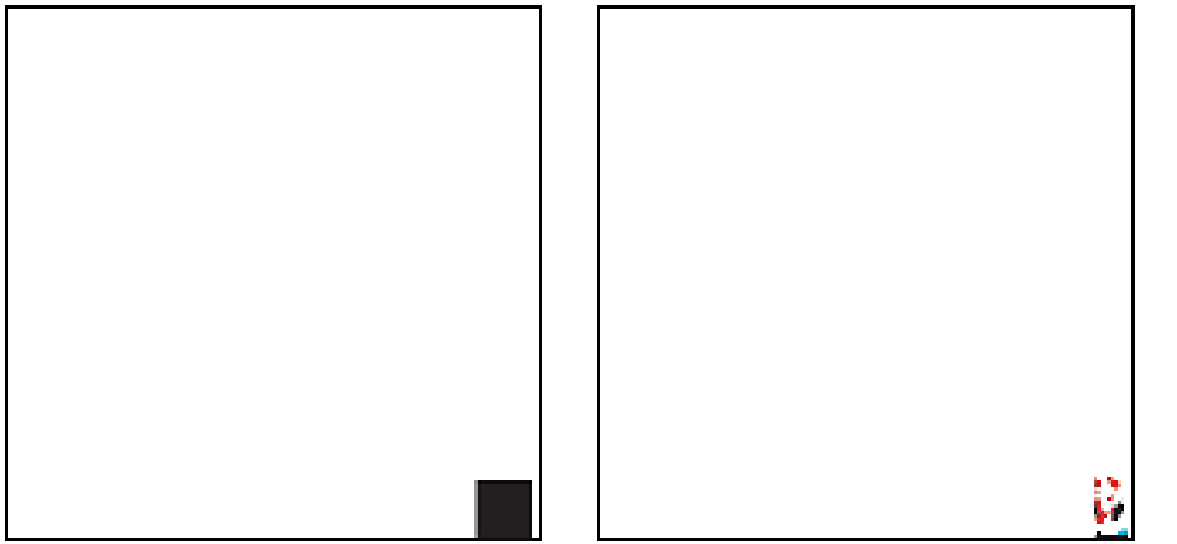
Original Trigger (L1 norm = 25)

Reversed Trigger ($m \cdot \Delta$) (L1 norm = 22.79)

# Reverse Engineered Trigger

*Neural Cleanse*

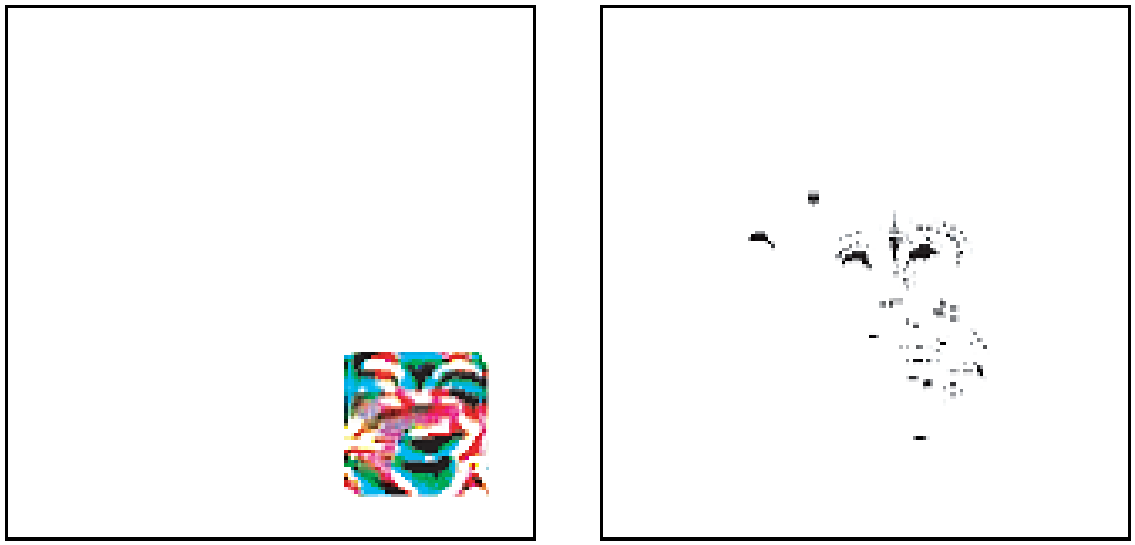
- Original vs reverse engineered trigger for face recognition on the PubFig dataset



# Reverse Engineered Trigger

*Neural Cleanse*

- The previous examples considered the BadNet poisoning attack, where backdoor triggers are inserted into clean images
- Next, the authors considered defense against Trojaned networks, by considering trojan square and trojan watermark triggers applied for face recognition DNNs
- Original vs reverse engineered trigger with trojan square trigger
  - The size of the trigger is 7% of the image size



Original Trigger
(L1 norm = 3,481)

Reversed Trigger ($m$)
(L1 norm = 311.24)

# Reverse Engineered Trigger

*Neural Cleanse*

- Original vs reverse engineered trigger with trojan watermark trigger
  - The size of the trigger is 7% of the image size
  - The reverse engineering triggers in Trojan network attack look visually different and approach in different locations than the original trigger
  - This is due to the different attack mechanism with BadNet attack, where Trojan network attack targets specific neurons, and cannot avoid the effects on other neurons in the model

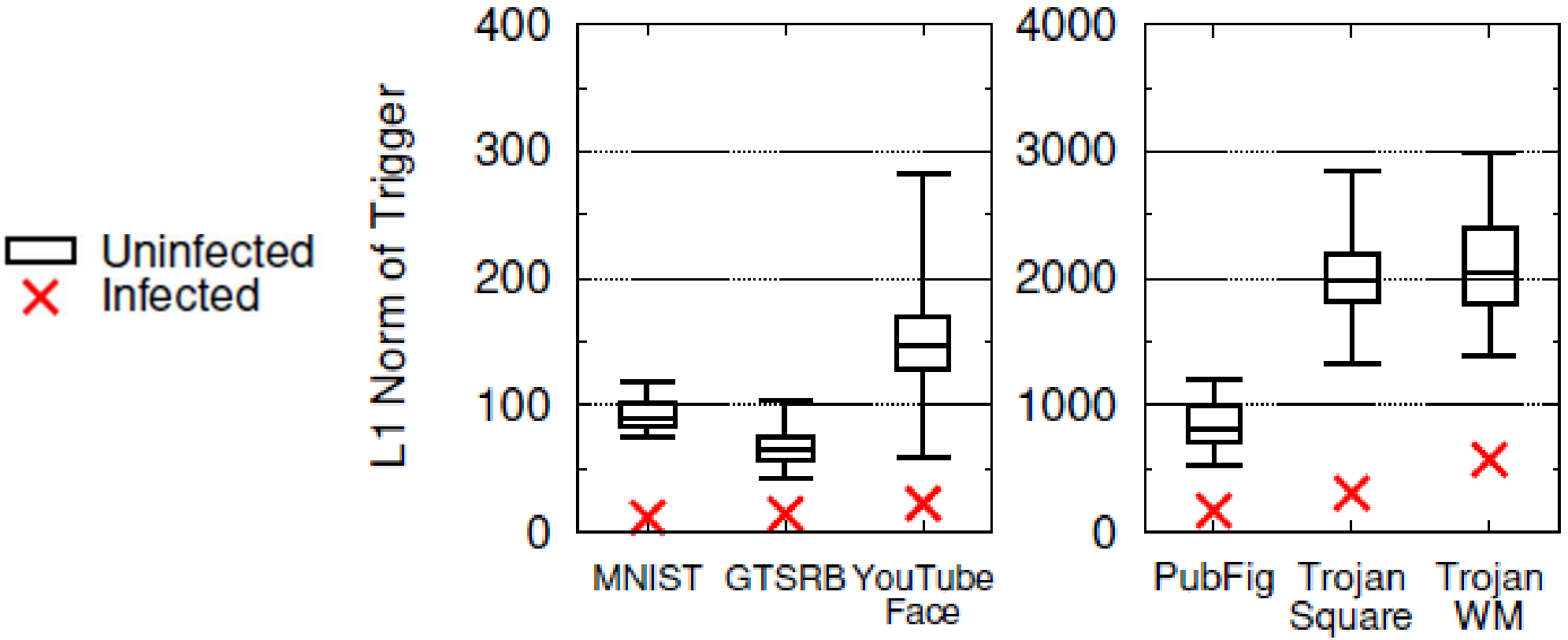
Original Trigger
(L1 norm = 3,598)

Reversed Trigger ($m$)
(L1 norm = 574.24)

# Detecting Backdoors Via Outlier Detection

*Neural Cleanse*

- Detecting backdoors is based on observed differences in the distribution of the L1 norm of the triggers for infected and uninfected labels
  - The Boxplots show median values, 25/75 quartiles, and min/max values
  - L1 norm of the infected label is much lower than the L1 norm of uninfected labels
    - Note that there was only one infected label in the shown poisoned models



# Detecting Backdoors Via Outlier Detection

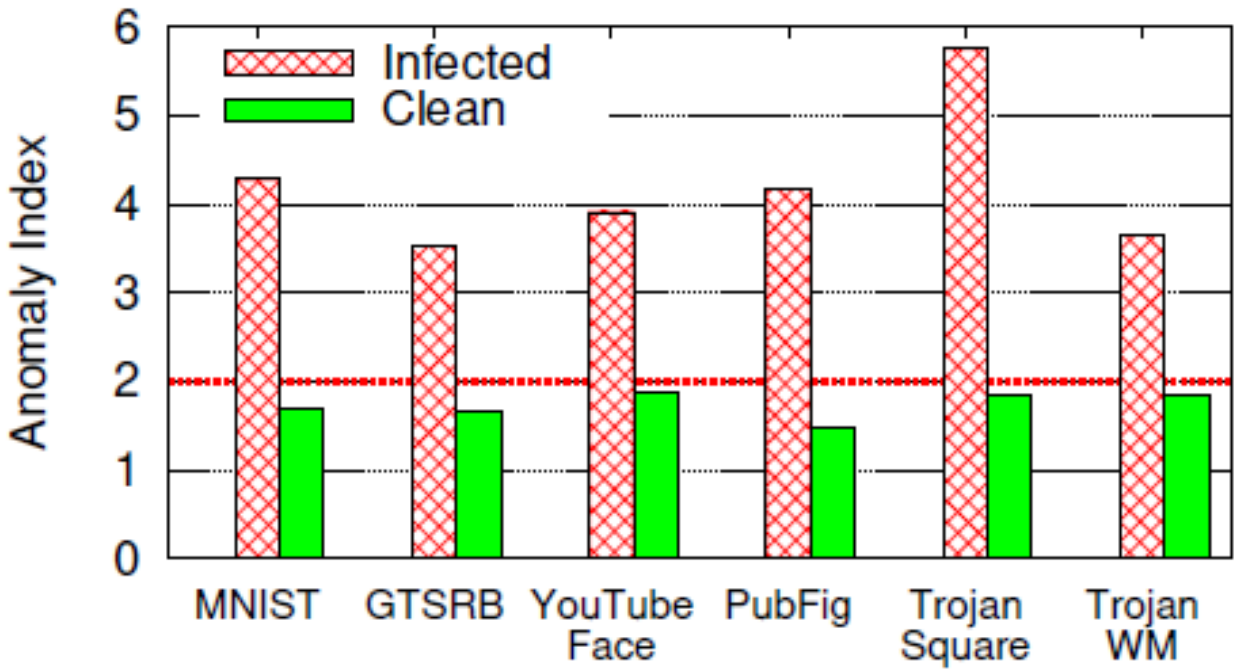
*Neural Cleanse*

- To detect a backdoored model, apply an anomaly detection approach for the distribution of the L1 norm of the triggers
- Approach:
  - Calculate Median Absolute Deviation (MAD), i.e., the absolute deviation between all other labels and the target label
  - Calculate anomaly index, as the absolute deviation divided by MAD
- If the anomaly index is larger than 2, the model has over 95% probability of being infected

# Anomaly Index

*Neural Cleanse*

- Anomaly index of infected and clean models for the considered attacks are shown in the figure below
  - All infected models have large anomaly index, whereas all clean models have smaller anomaly index (less than 2)
  - The authors assume that an anomaly index greater than 3 indicates over 99.7% probability of the model being infected



# Mitigating Backdoors

*Neural Cleanse*

- Once a backdoored model is detected, Neural Cleanse introduces three mitigation strategies to preserve the model performance
  1. Filter that identifies adversarial inputs with a known trigger
  2. Model patching algorithm based on neuron pruning
  3. Model patching algorithm based on unlearning

# Filter for Detecting Adversarial Inputs

*Neural Cleanse*

- Filter is developed based on the neuron activation profile of the model for samples with a reversed trigger
  - Activation profile is measured as the average values of neuron activations of top 1% neurons in the second to last layer in the model
- The input samples with high activation profiles are identified as potential backdoored samples, and are filtered (removed) from the dataset
  - This is based on a certain threshold, determined by conducting experiments on clean inputs and poisoned images

# Patching DNN via Neuron Pruning

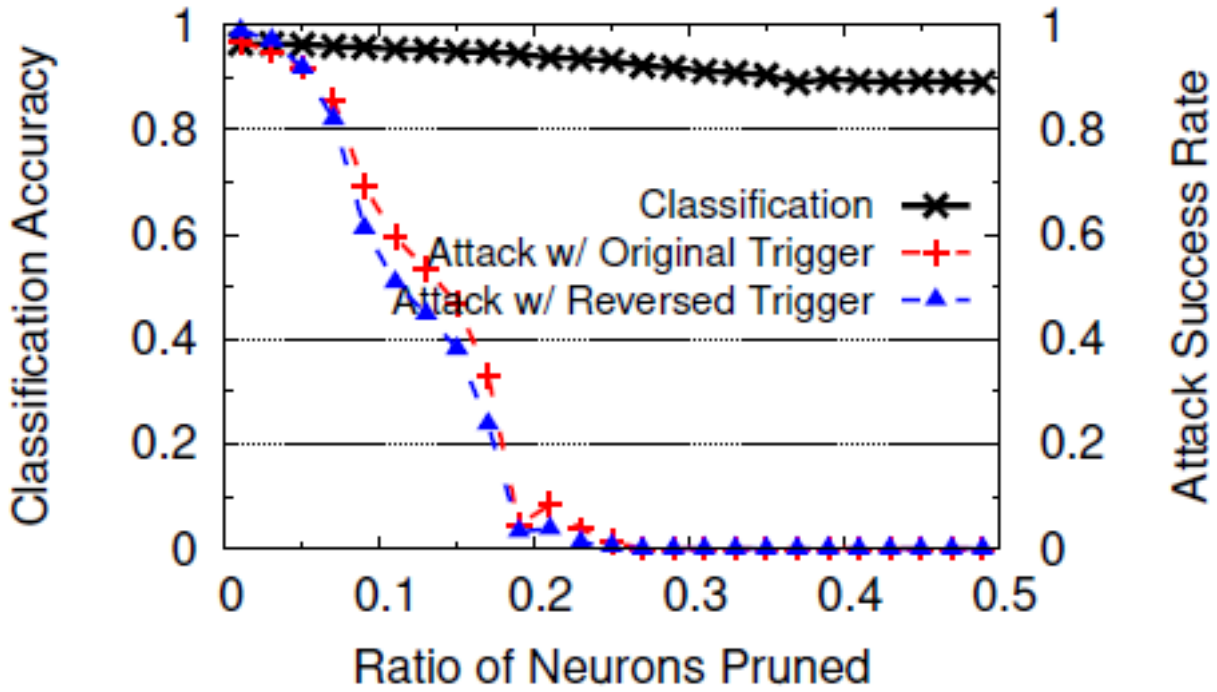
*Neural Cleanse*

- Neuron Pruning mitigation
- Use the reversed trigger to identify backdoor-related neurons
  - Prune these neurons, by setting their output value to 0 during inference
- Prioritize neurons with greatest activation gaps between clean and adversarial inputs
- Stop pruning when the model is no longer responsive to removing neurons
  - To minimize the impact on classification accuracy of clean inputs

# Patching DNN via Neuron Pruning

*Neural Cleanse*

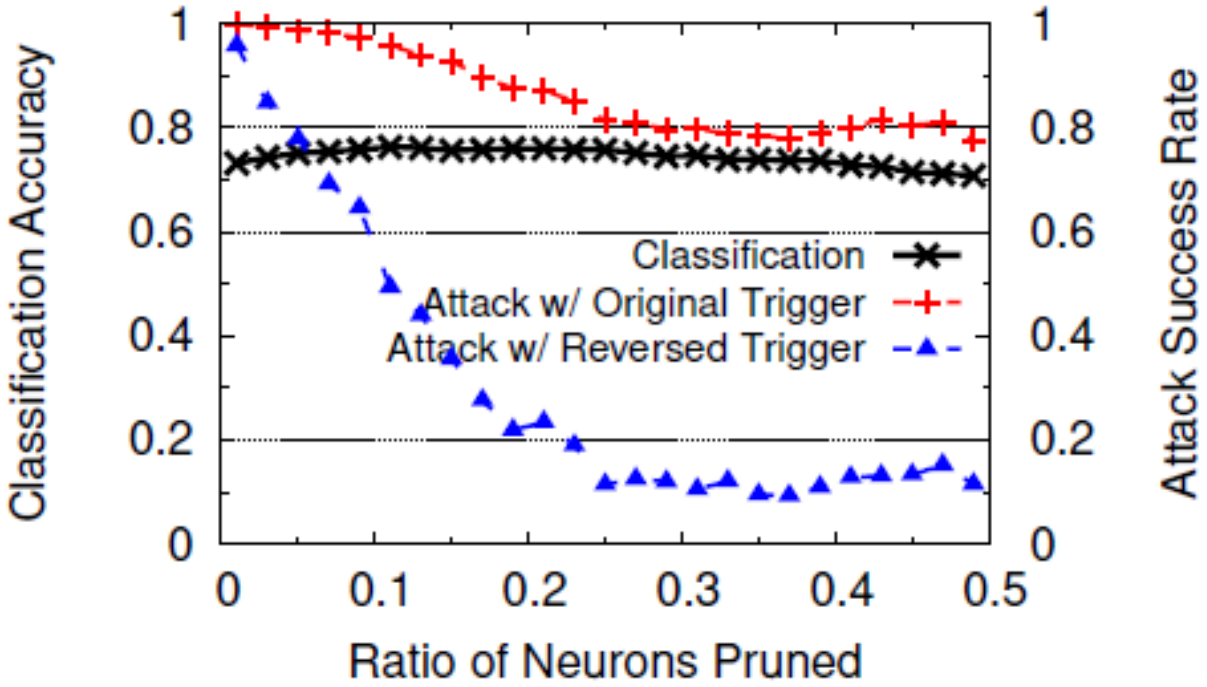
- Classification accuracy and attack success rate when pruning trigger-related neurons in GTSRB model for traffic sign recognition
  - Pruning 30% of all neurons reduces the attack success rate to 0, and also reduces the classification accuracy on clean samples by about 5%
  - The attack success rate or the original and reversed trigger are almost the same, meaning that the reversed trigger is effective for neuron pruning



# Patching DNN via Neuron Pruning

*Neural Cleanse*

- The figure shows the accuracy and attack success rate for Trojan square attack
- Neuron pruning is less successful against Trojan network attack
  - E.g., when 30% of neurons are pruned, attack success rate drops to about 10% with the reversed trigger, but the attack success rate of the original trigger remains over 80%
  - This discrepancy is due to the dissimilarity in neuron activations between reversed trigger and the original trigger



# Patching DNN via Unlearning

*Neural Cleanse*

- Patching the model via Unlearning mitigation
- Use the reversed trigger to train infected DNN to recognize correct labels when the trigger is present
  - This allows the model to decide which weights are problematic and update them
- Fine-tune the model for only 1 epoch using updated training dataset
  - Comprised of 10% of original training data (clean, no trigger)
  - Add the reversed trigger to 20% of this sample without modifying the labels
- The table shows that unlearning reduced the attack success rate to less than 6.7% when using the reversed trigger

| Task | Before Patching | | Patching w/ Reversed Trigger | | Patching w/ Original Trigger | |
|---|---|---|---|---|---|---|
| | Classification Accuracy | Attack Success Rate | Classification Accuracy | Attack Success Rate | Classification Accuracy | Attack Success Rate |
| MNIST | 98.54% | 99.90% | 97.69% | 0.57% | 97.77% | 0.29% |
| GTSRB | 96.51% | 97.40% | 92.91% | 0.14% | 90.06% | 0.19% |
| YouTube Face | 97.50% | 97.20% | 97.90% | 6.70% | 97.90% | 0.0% |
| PubFig | 95.69% | 97.03% | 97.38% | 6.09% | 97.38% | 1.41% |
| Trojan Square | 70.80% | 99.90% | 79.20% | 3.70% | 79.60% | 0.0% |
| Trojan Watermark | 71.40% | 97.60% | 78.80% | 0.00% | 79.60% | 0.00% |

# SentiNet

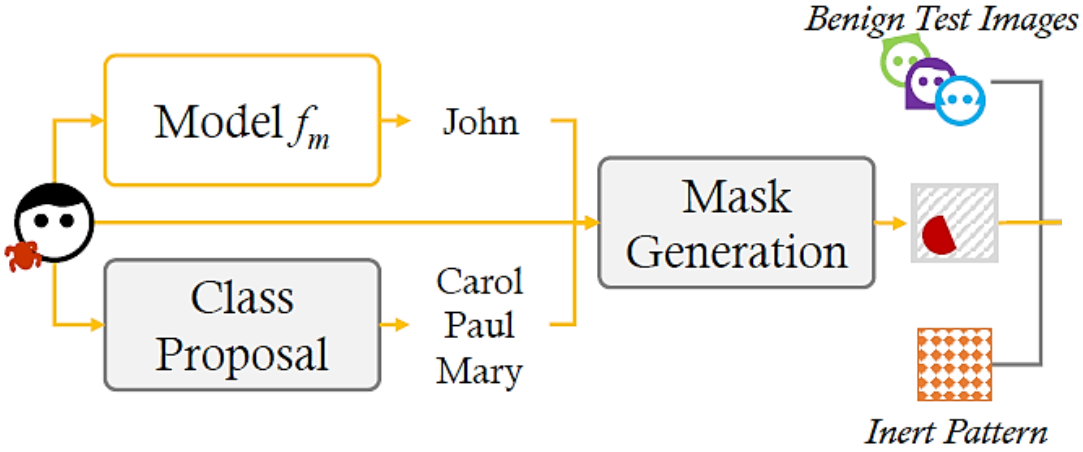


- ***SentiNet***
  - Chou (2020) - SentiNet: Detecting Localized Universal Attacks Against Deep Learning Systems
- SentiNet defense is effective with poisoning attacks that are:
  - Localized: the trigger is constrained to a small portion of an image
  - Sample-agnostic (universal): the same trigger is applied to all images
- Approach:
  - First, apply explainability approaches to discover regions in input images that may contain a backdoor trigger
  - Second, extract those regions and patched them on many clean images with correct ground-truth labels
    - If the patched images are misclassified, the extracted patch contains a backdoor trigger
- This defense is effective against data poisoning and trojaned networks

# SentiNet

*SentiNet*

- Phase 1: Adversarial object localization
  - The goal is to localize the region that might contain the trigger
  - Step 1: Class proposal
    - Identify all possible classes by a model $f_m$ for an input image via segmentation
    - Segment the objects in an image, and for each object return a prediction
    - This set of predictions will exclude the actual prediction by the model
  - Step 2: Mask generation
    - Use Grad-CAM to identify regions in the image the have the greatest influence for each predicted class $C$ (for all objects from Step 1)
    - Grad-CAM is an approach for explainable Machine Learning that outputs a heatmap of the most important regions in an image for a class $C$



# SentiNet

*SentiNet*

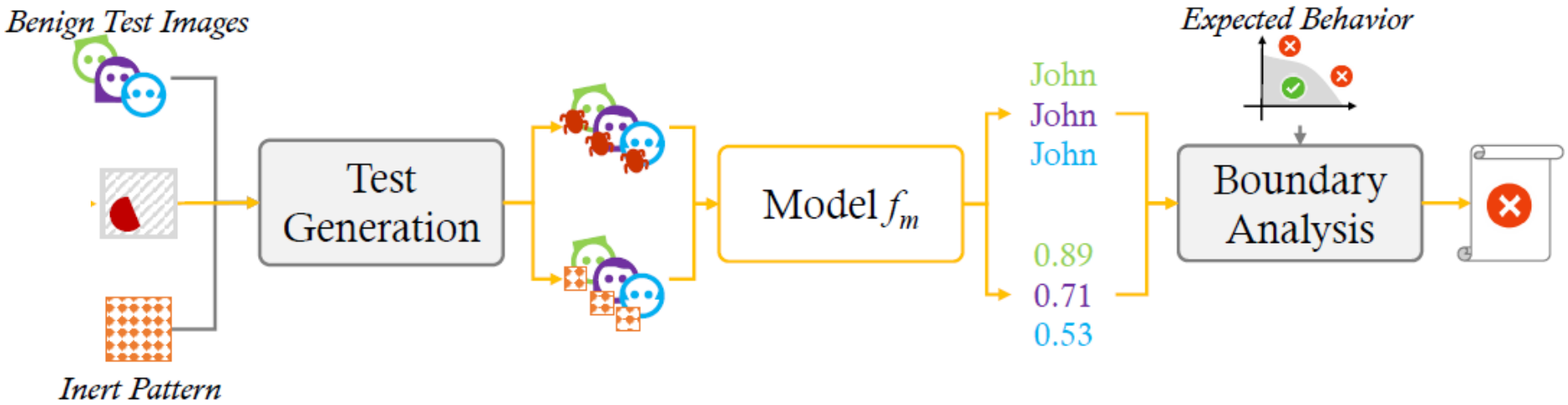
- The heatmap of Grad-CAM for a poisoned image may cover both the malicious trigger and benign portions of the image

- To improve the mask *M*, the authors subtracted the heatmap obtained by Grad-CAM for clean images (middle sub-figure below)
  - This resulted in masks that contain the trigger mostly, and less of benign regions

# SentiNet

*SentiNet*

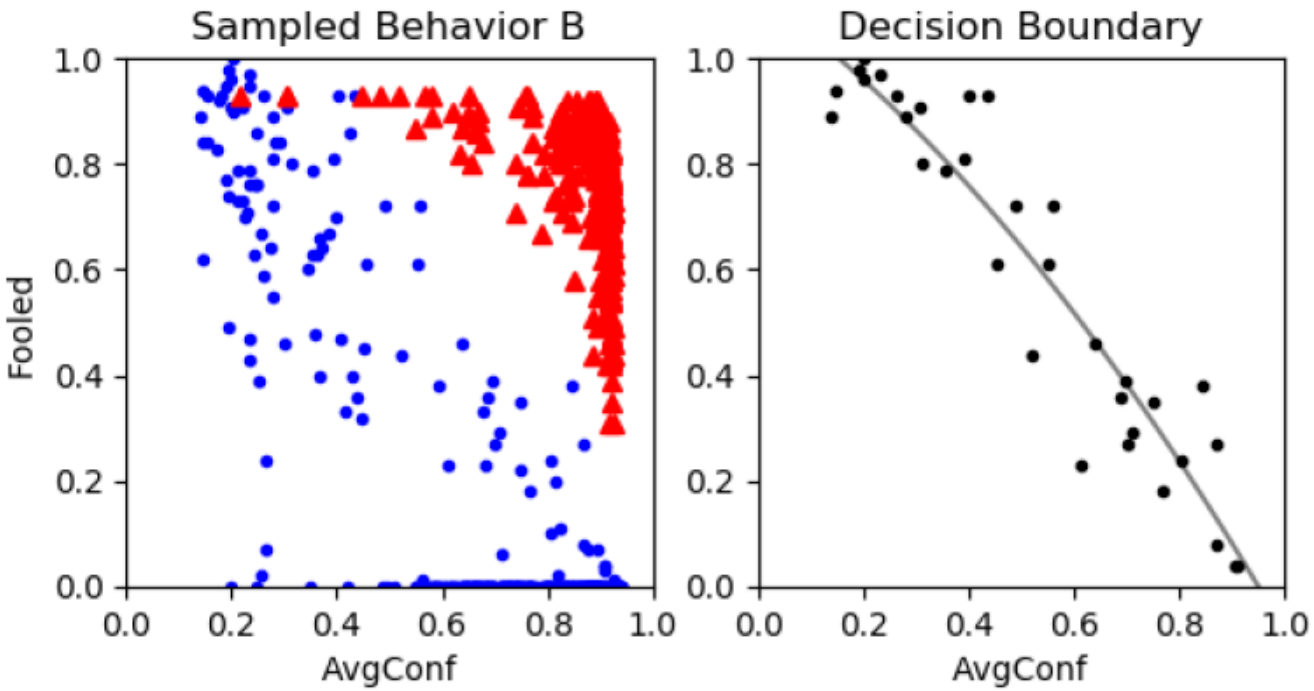
- Phase 2: Adversarial attack detection
  - Step 1: Test set generation
    - Overlay the mask region *M* with the malicious trigger on a set of benign images
    - Evaluate the model on this set of mutated images
    - If the accuracy of the model is low, the suspected mask region *M* is a malicious trigger
  - Step 2: Boundary analysis
    - Create a second set where the content of the mask region *M* is replaced with Gaussian noise (referred to as inert pattern)
    - It is expected that the inert pattern will not impact significantly the decision by the model
    - Analyze the decision boundary between images with inert pattern and suspected region



# SentiNet

*SentiNet*

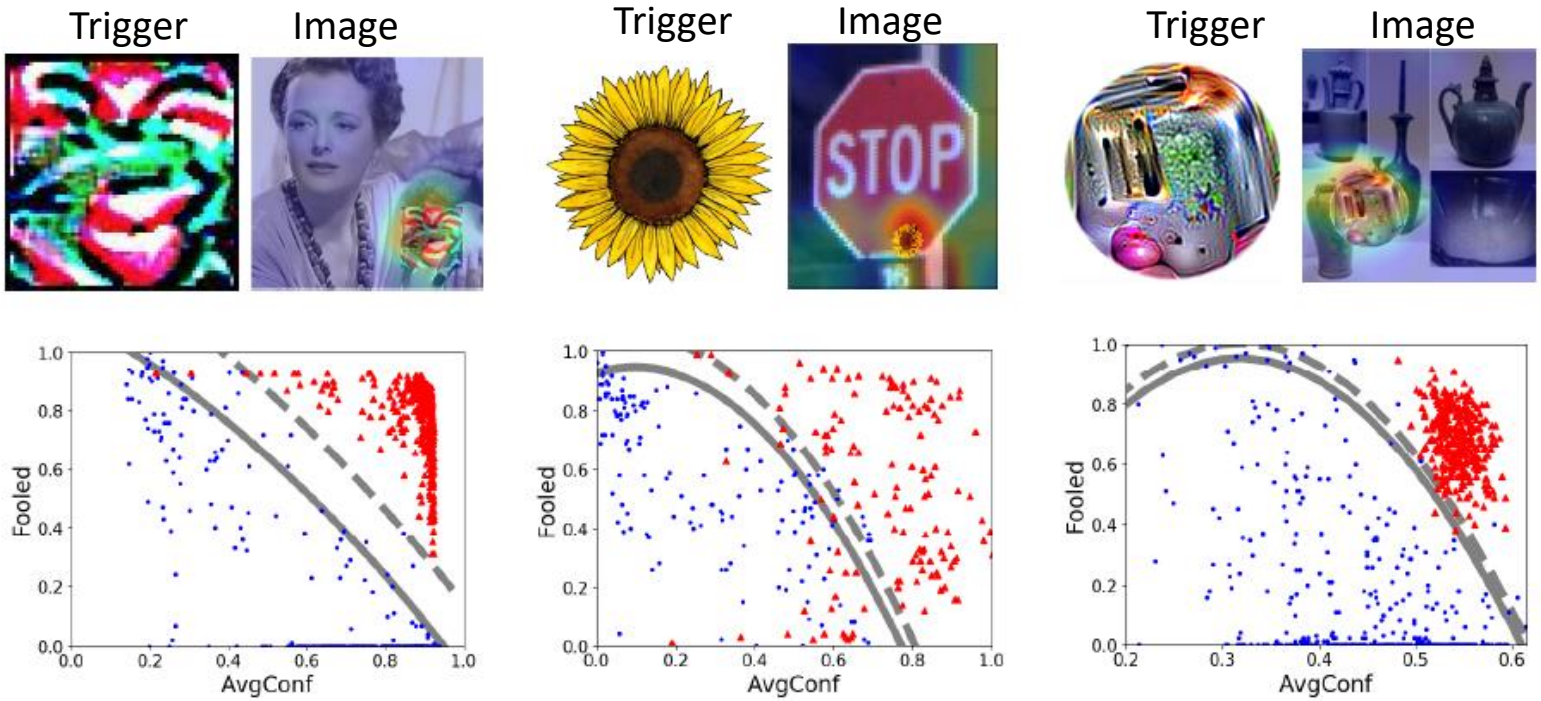
- Decision boundary analysis
  - Red triangles are poisoned samples, blue circles are clean samples
  - Horizontal axis: average confidence of the model, vertical axis: number of fooled images (images misclassified by the model)
    - One possible approach to separate the samples is to apply a threshold value
    - The authors implemented a binary classifier to approximate the decision boundary based on the available set of clean samples



# SentiNet

*SentiNet*

- Evaluation of SentiNet on Trojaned network attack (face detection), backdoor attack (traffic sign recognition), and adversarial patch attack (ImageNet image classification)
  - Bottom row: decision boundaries (solid lines) and thresholds (dashed lines)
  - SentiNet achieved high success rate for all three attacks, between 85% and 99%




(a) Trojan trigger. (b) Backdoor. (c) Adversarial patch.