AI System Evaluation

Week 5: Mitigating AI Backdoors

Aug 23 - Week 1: 7-10	Introduction	
Aug 30 - Week 2: 7-10	AI Robustness	Exercise 1
Sep 06 - Week 3: 7-10	Improving AI Robustness	Exercise 2
Sep 13 - Week 4: 7-10	AI Backdoors	Exercise 3
Sep 20 - Week 5: 7-10	Mitigating AI Backdoors	Exercise 4; Project Proposal
Sep 27 - Week 6: 7-10	AI Fairness	Exercise 5
Oct 11 - Week 7: 7-10	Improving AI Fairness	Exercise 6
Oct 18 - Week 8: 7-10	AI Privacy	Exercise 7
Oct 25 - Week 9: 7-10	Improving AI Privacy	Exercise 8
Nov 01 - Week 10: 7-10	AI Interpretability	Project Due
Nov 08 - Week 11: 1-3	End-of-Term Exam	

Outline

Question

If we suspect a model is embedded with backdoor(s), how do I mitigate the backdoor attack?

Approaches

Input filtering

Input sanitization

Sanitizing the model

Certified defense

Input Filtering

Input Filtering

Training-time input filtering

If a training sample is determined to be suspicious (e.g., having weird patches, background patterns, or wrong labels), remove it from the training dataset.

Inference-time input filtering

If a test sample is determined to be suspicious, do not serve it.

Question

How do we determine whether a sample is to be filtered, i.e., suspended to be poisoned?

Answer

We design or train a classifier to do that.

The classifier can be based on either the sample itself or how it is represented in the model trained on all training data.

Testing-Time Input Filtering

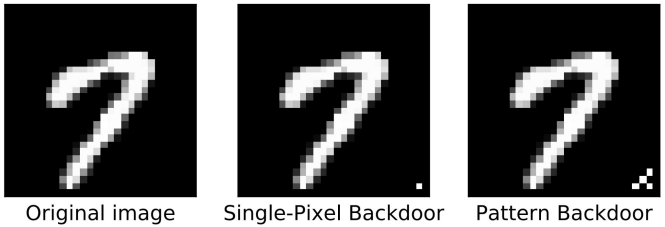
Approach: Anomaly Detection

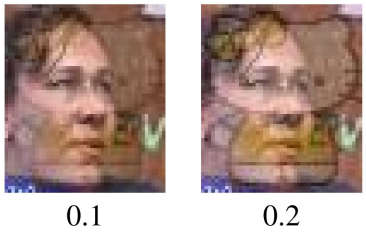
An attacked sample is often abnormal in some ways and thus we can prevent backdoor attack by disallowing abnormal inputs.

How do we determine such an input is abnormal? By using a (decision tree) classifier*.

*“Neural Trojans”, ICCD 2017.

Examples





Anomaly Detection

Approach

To learn a classifier, we need abnormal samples. How do craft abnormal samples?

The authors crafted abnormal samples themselves.

If an input is deemed an anomaly, it is discarded.

Example: Abnormal Samples

This work was done before BadNet and thus these are mostly OOD samples.

Training-time Input Filtering

Approach

Poisoned sample and clean samples are represented different by the model training on all the training data.

If we perform PCA analysis of training samples in each class (to see which part of a sample is informative for the prediction), the outliers are likely the poisoned sample.

Algorithm

1. Train a model using all training data.
2. Conduct PCA analysis on all training data in each class.
3. Remove outliers in each class according to the PCA analysis
4. Train a model using the remaining data.

Discussion

Do you think anomaly detection would be effective against the backdoor attacks that we discussed last week?

Most effective: ⭐⭐⭐⭐⭐

Least effective: ⭐

BadNet	
Invisible Backdoor Attacks	
Semantic backdoor	
Reflection Backdoor	
Clean-label invisible attack	
Trojaning	
TrojanNet	
Physical Attack	
“Natural” Backdoor	

Input Sanitization

Input Sanitization

Assumption

The performance of a backdoor is sensitive to modification of the trigger, i.e., perturbing the trigger often reduces the success rate of the trigger significantly.

We can apply simple transformation such as left-right flipping and padding after shrinking to reduce the effectiveness of the trigger*.

*“Neural Trojans”, ICCD 2017.

Training-time input sanitization

Apply some form of transformation (hopefully to disable the trigger) before using the sample for training the neural network.

Inference-time input sanitization

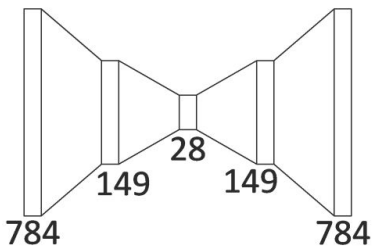
Apply some form of transformation (hopefully to disable the trigger) before feeding the test sample into the neural network.

Do you think that backdoor triggers are sensitive to perturbation?

Preprocessing through Autoencoder

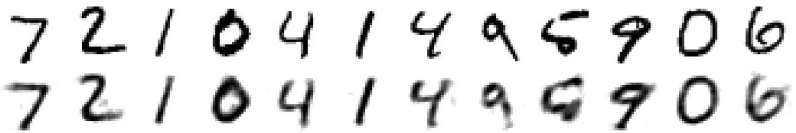
Intuition

We use an autoencoder to encode and decode the sample, and hopefully the trigger is disabled somehow.



Approach

The training objective is to minimize the mean square error between the input and output. Here are some before/after samples.



During inference, an input is encoded and the decoded sample is fed into the neural network.

Exercise 1

Given week5/exercise1/sensitive.py, perturb the trigger in two different ways and measure the change in attacking success rate.

- Change the trigger itself (e.g., increasing/reducing the size of the white-patch)
- Shift the trigger to different positions

Discussion

Based on your understanding, consider whether preprocessing through autoencoder would be effective against the range of backdoor attacks that we discussed last week.

Most effective: ⭐⭐⭐⭐⭐

Least effective: ⭐

BadNet	
Invisible Backdoor Attacks	
Semantic backdoor	
Reflection Backdoor	
Clean-label invisible attack	
Trojaning	
TrojanNet	
Physical Attack	
“Natural” Backdoor	

STRIP

Assumption

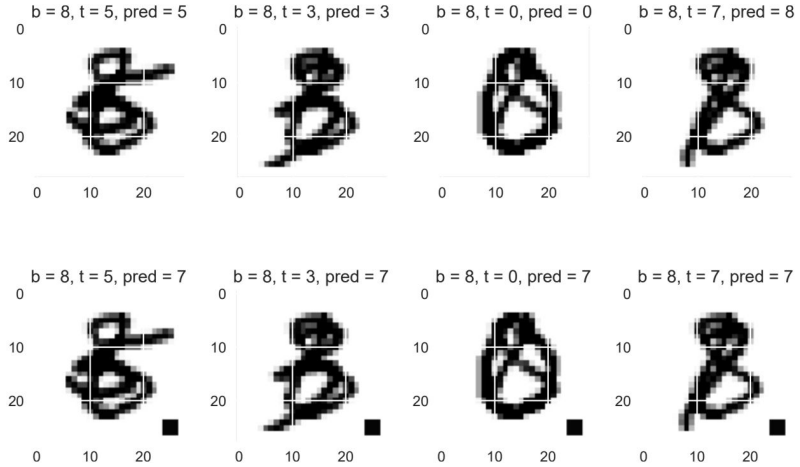
If we modify a benign sample, the prediction is likely to change.

If we modify an attacked sample, the prediction is like to remain the same.

What do you think of this assumption?

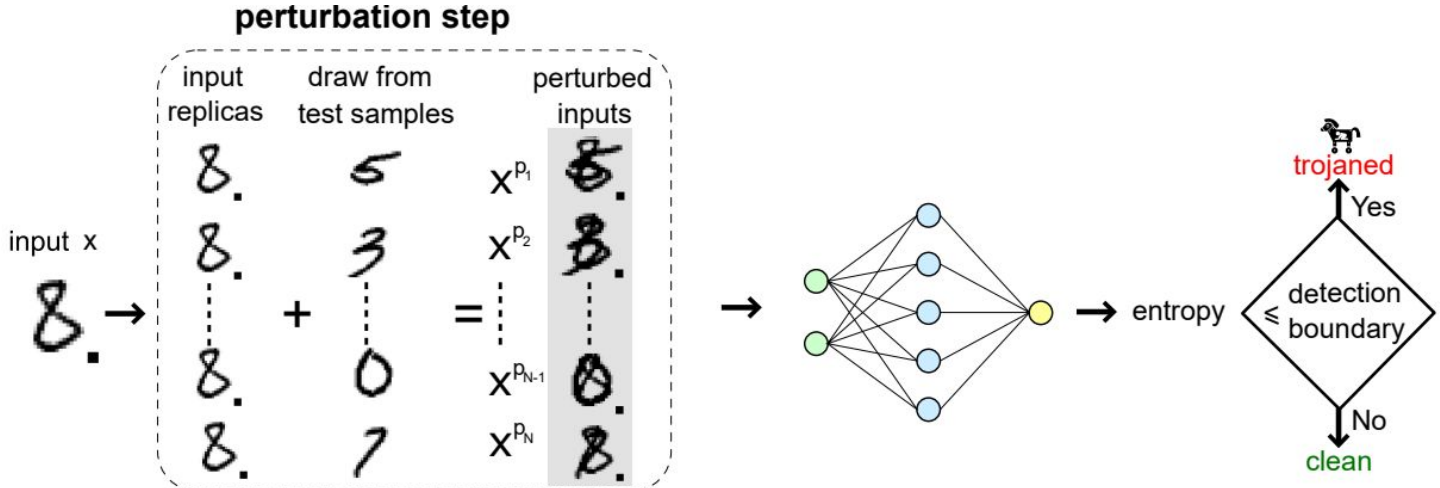
Example

We superimpose two images, one at the bottom (b) and the other at the top (t).



STRIP

If the distribution of the prediction remains more or less the same, the input is considered to be attacked and discarded.



Discussion

Based on your understanding, consider whether STRIP would be effective against the range of backdoor attacks that we discussed last week.

Fill the table.

Most effective: ⭐⭐⭐⭐⭐

Least effective: ⭐

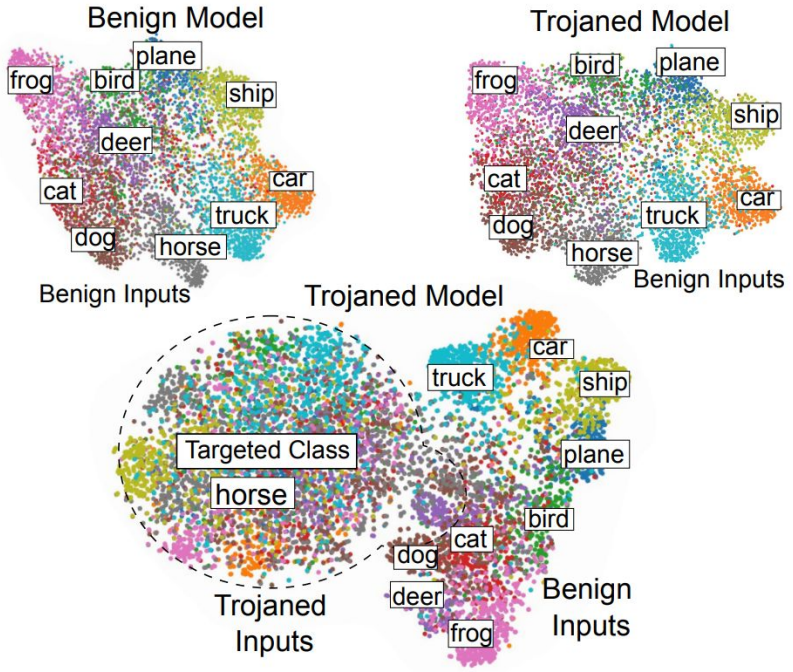
BadNet	
Invisible Backdoor Attacks	
Semantic backdoor	
Reflection Backdoor	
Clean-label invisible attack	
Trojaning	
TrojanNet	
Physical Attack	
“Natural” Backdoor	

Februus

Observation

The logits (which can be regarded as the extracted features) generated from benign inputs and attacked inputs are very different.

*Februus: Input Purification Defense Against Trojan Attacks on Deep Neural Network Systems, ACSAC 2020



CIFAR-10 visualized using t-SNE

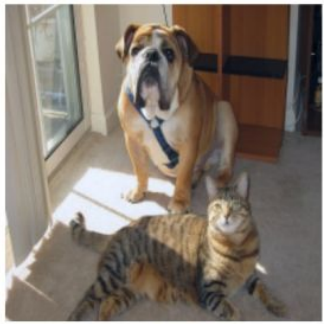
Februus

Assumption

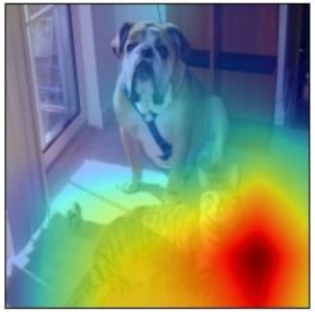
In the presence of a trigger, some region will contribute significantly to the prediction.

Approach

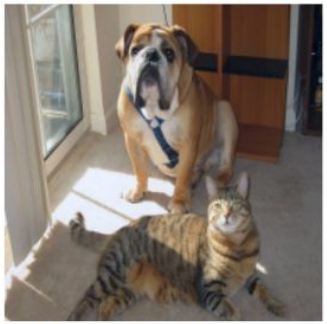
1. Identify the region which contributes the most to the prediction using Grad-CAM (a.k.a. a heat map).
2. Remove the region and apply GAN to restore the removed region.
3. Feed the restored image into the neural network.



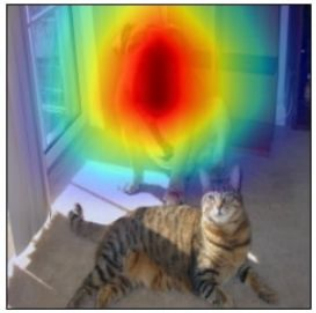
(a) Original Image



(f) ResNet Grad-CAM 'Cat'

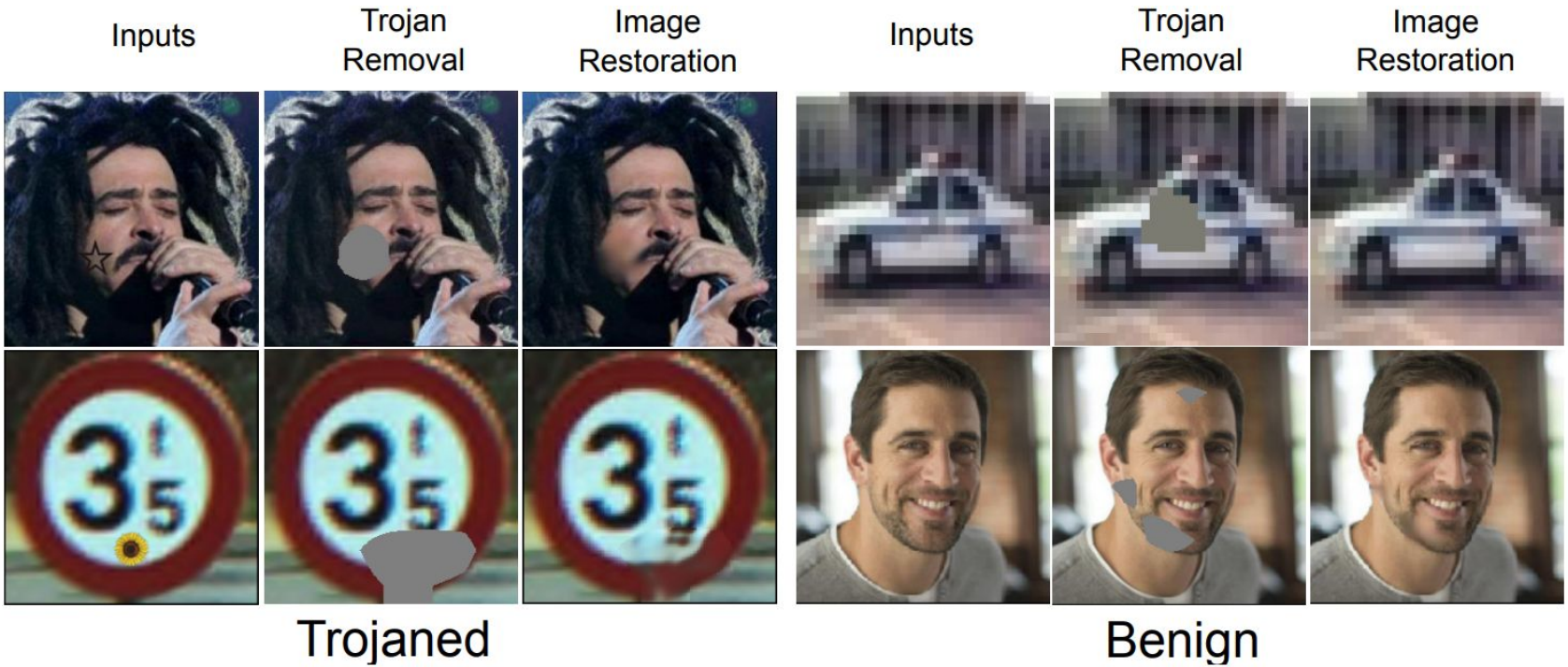


(g) Original Image



(l)ResNet Grad-CAM 'Dog'

Februus: Illustration



Discussion

Based on your understanding, consider whether Februus would be effective against the range of backdoor attacks that we discussed last week.

Fill the table.

Most effective: ⭐⭐⭐⭐⭐

Least effective: ⭐

BadNet	
Invisible Backdoor Attacks	
Semantic backdoor	
Reflection Backdoor	
Clean-label invisible attack	
Trojaning	
TrojanNet	
Physical Attack	
“Natural” Backdoor	

Sanitizing The Model

Retraining

Intuition

Based on the phenomenon of catastrophic forgetting, fine-tuning the given model with some clean labeled data helps to eliminate the backdoor.

*"Neural Trojans", ICCD 2017.

Catastrophic Forgetting

"is the tendency of an artificial neural network to completely and abruptly forget previously learned information upon *learning new information*."

Example: A neural network trained to recognize digit "1" and "2" will perform terribly on classifying "1" and "2" after fine-tuned to recognize "3" and "4".

This works if the neural network is retraining on OOD samples.

Retraining

Approach

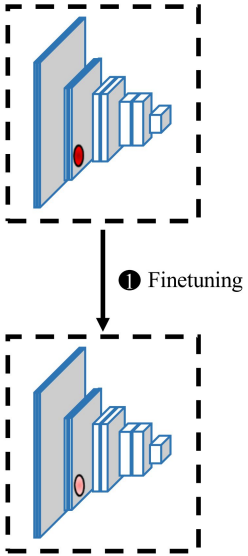
Sample (x, y) from a distribution which is supposed the same as the training dataset.

Train the given neural network with the cross entropy loss.

$$\text{Min}_{\theta}\ L_{CE}(\theta, x, y)$$

*“Neural Trojans”, ICCD 2017.

Illustration



Exercise 2

Given week5/exericse2/badnet.pt which is a backdoored network, week5/exericse2/retrain.py implements the retraining approach, measure the accuracy and attack success rate after retraining.

Can you explain why this result is as what it is?

Neural Attention Distillation

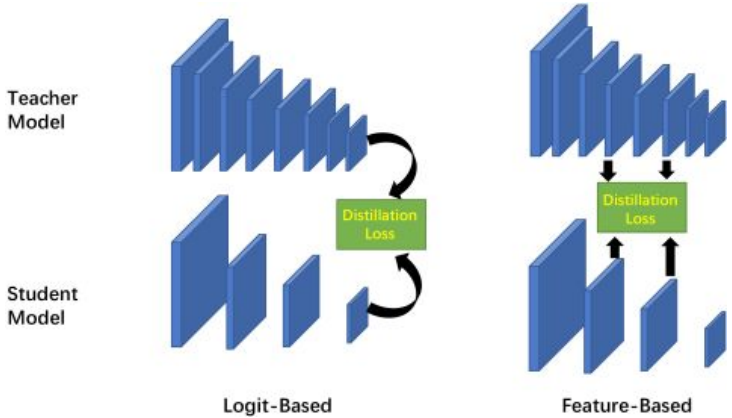
High-level Idea*

To further enhance the effect of retraining (i.e., forgetting about the backdoor) through knowledge distillation.

*“Neural attention distillation: Erasing backdoor triggers from deep neural networks,” ICLR 2021.

Knowledge Distillation

It is proposed as a way to reduce model size. It has the effect of pruning certain unnecessary details.



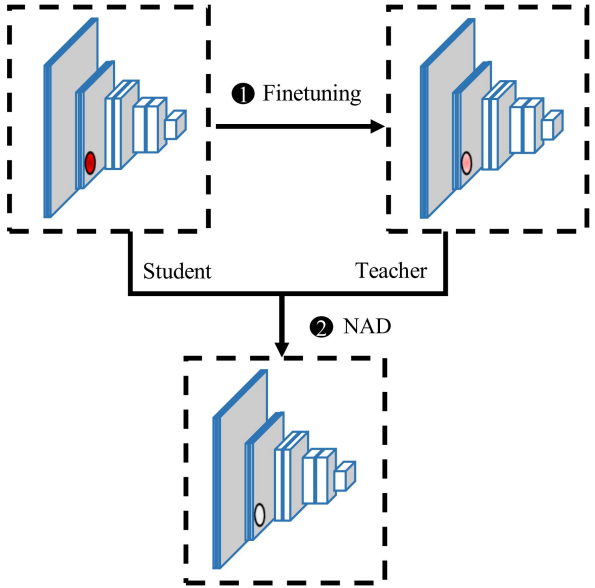
Neural Attention Distillation

Approach

Given a model N (which is believed to be backdoored) and a set of self-collected clean samples,

1. finetune N with the clean samples to obtain a model M.
2. take M as the teacher model and N as the student model and apply knowledge distillation.

Illustration



Neural Attention Distillation

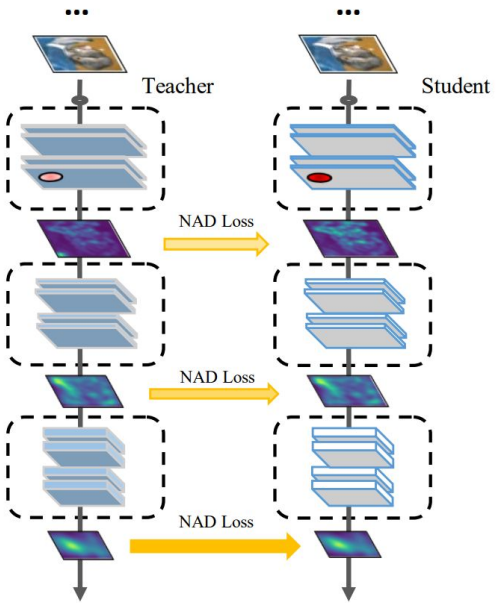
Approach

The distillation is applied to optimize the student model (i.e., the backdoored model) with the objective of minimizing

- the cross entropy on the clean samples,
- and the difference between the teacher model and student model.

There are some details on how to define the difference in the paper.

Illustration



Exercise 3

Do you think NAD would be effective against the backdoor attacks that we discussed last week?

Fill the table and explain why it is or isn't effective.

Most effective: ⭐⭐⭐⭐⭐

Least effective: ⭐

BadNet	
Invisible Backdoor Attacks	
Semantic backdoor	
Reflection Backdoor	
Clean-label invisible attack	
Trojaning	
TrojanNet	
Physical Attack	
"Natural" Backdoor	

Adversarial Unlearning Of Backdoors

High-level Idea*

Making retraining more effective in terms of removing backdoors in the model.

The proposal is inspired by adversarial training.

* “Adversarial unlearning of backdoors via implicit hypergradient,” ICLR 2022.

Approach

Assume there is a small set of clean samples $(x_i, y_i) * n$. Retrain the given model with the following objective.

$$\text{Min}_{\theta}\ \text{Max}_{||\delta|| < \varepsilon}\ (\Sigma_i L(\theta, x_i + \delta, y_i))/n$$

where δ is intuitively an arbitrary trigger; $||\delta|| < \varepsilon$ defines the space of all possible triggers.

How do we define $||\delta|| < \varepsilon$ in general?

Adversarial Unlearning Of Backdoors

Approach

Assume there is a small set of clean samples (x_i, y_i) * n. Retrain the given model with the following objective.

$$\text{Min}_{\theta}\ \text{Max}_{||\delta||<\varepsilon}\ (\Sigma_i L(\theta, x_i+\delta, y_i))/n$$

where δ is intuitively an arbitrary trigger; $||\delta||<\varepsilon$ defines the space of all possible trigger.

Detail 1: $||\delta||<\varepsilon$ can be defined based on, for instance, L_2-norm or other norms.

Detail 2: Note that the objective function is un-targeted. This is because in general, we don't know what is the attack target.

Detail 3: Solving the inner max problem is highly nontrivial. It is basically the problem of generating (untargeted) universal adversarial perturbation (UAP).

Adversarial Unlearning vs Adversarial Training

Adversarial Unlearning

Retrain the given model with the following objective.

$$\text{Min}_{\theta}\ \text{Max}_{||\delta||<\varepsilon}\ (\Sigma_i L(\theta, x_i+\delta, y_i))/n$$

Adversarial Training

Train a model with the following objective

$$\text{Min}_{\theta}\ \text{Max}_{D(x,x')<\varepsilon}\ L(\theta, x', y)$$

Difference

Adversarial unlearning aims to minimize the maximize effect on all samples, whereas adversarial training focuses on one sample at a time.

In adversarial training, we approximate $\text{Max}_{D(x,x')<\varepsilon}\ L(\theta, x', y)$ through adversarial attack. In the problem of approximating

$$\text{Max}_{||\delta||<\varepsilon}\ (\Sigma_i L(\theta, x_i+\delta, y_i))/n$$

is much harder.

Discussion

Do you think adversarial unlearning would be effective against the backdoor attacks that we discussed last week?

Fill the table.

Most effective: ⭐⭐⭐⭐⭐

Least effective: ⭐

BadNet	
Invisible Backdoor Attacks	
Semantic backdoor	
Reflection Backdoor	
Clean-label invisible attack	
Trojaning	
TrojanNet	
Physical Attack	
“Natural” Backdoor	

Trigger Synthesis based Defense

High-level Idea

If we know what is the backdoor trigger, we can most certainly defend backdoors much more effectively.

How do we synthesize the trigger then?

Example

If we know what is the trigger,

- we can simply discard those trigger-containing input either during training or testing;
- we can remove the trigger and restore the image during testing;
- we can precisely remove the backdoor through retraining or unlearning.

Neural Cleanse

Overall Approach

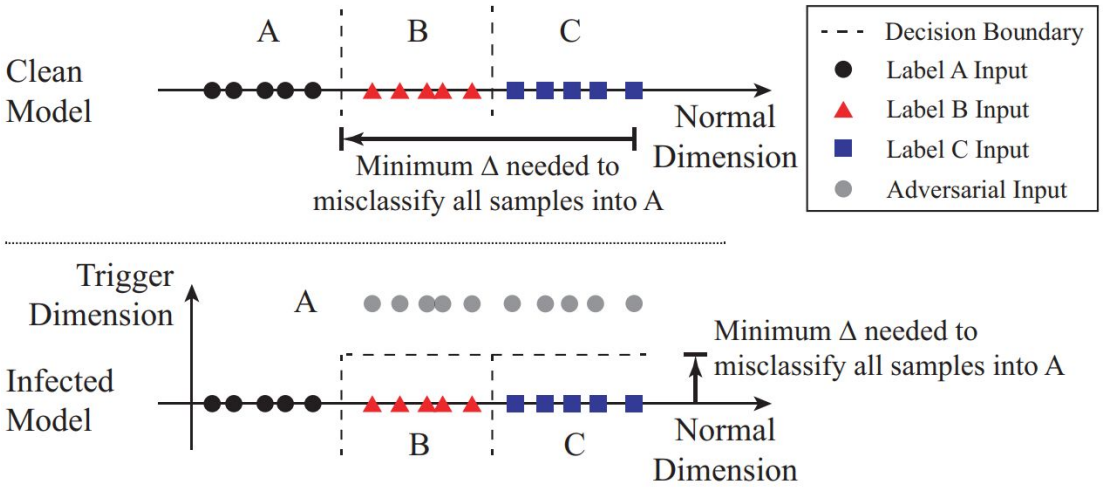
Given a neural network N, it aims to provide an end-to-end solution for the problem.

1. Detecting backdoor
2. Synthesizing the trigger
3. Removing backdoor

*Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks, S&P 2019.

High-level Intuition

Backdoors are shortcuts between classes.



Neural Cleanse

Approach

1. Detecting backdoor

   For each label y_t, generate a minimal trigger δ_t through optimization which changes the label of any sample to y_t;

   If there exists one trigger δ_t which is significantly smaller (e.g., in terms of number of pixels or character changes) than the rest, there is a backdoor with target y_t.

How to generate the minimal trigger?

A generic trigger is defined as:

$$x' = x + \alpha * \delta$$

Solve the following optimization problem for all x in a set of clean samples.

$$\min_{\alpha,\delta} L(\theta, N(x+\alpha*\delta), y_t)+\lambda*|\alpha|$$

Do you think step 1 is reasonable?

Neural Cleanse

How to detect triggers that are significantly smaller?

Using the Median Absolute Deviation (MAD).

Given x_1, x_2, ..., x_n,

x_m = median of x_1, x_2, ..., x_n

MAD = median($|x_i - x_m|$)

Anomaly index of x_i = $|x_i - x_m|$/MAD

if x_i is small with anomaly index > 2, infected.

Example

Data: (1, 1, 2, 2, 4, 6, 9)

Median: 2

Absolute Deviation: (1, 1, 0, 0, 2, 4, 7)

MAD: 1 (because the sorted absolute deviations are (0, 0, 1, 1, 2, 4, 7)).

Anomaly index of 6: |6-2|/1 which 4.

Neural Cleanse

Approach

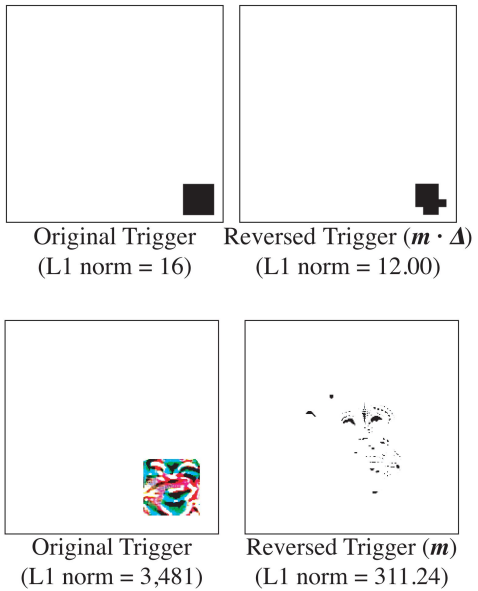
2. Synthesizing trigger

   The δ_t identify in the first step is the trigger and the target is y_t.

   We can note the patterns of neuron activation (top 1% of the logits layer) in the presence of the trigger and filter inputs accordingly.

Notice how the synthesized trigger is different and has smaller norm?

Examples



Neural Cleanse

Approach

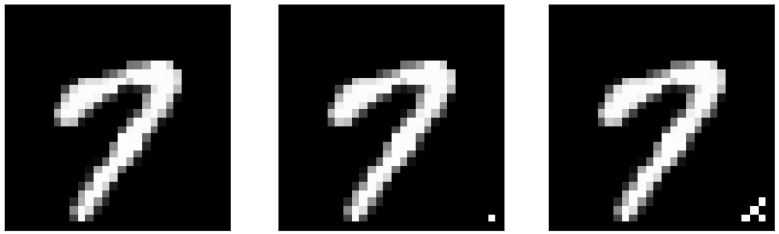
3. Removing backdoor

   One idea is to disable neurons (at the logits-layer) that are most activated in the presence of the trigger.

   The other is neural unlearning, i.e., finetuning with samples “stamped” with the trigger and the correct label.

Examples

Ideally, if the trigger is perfect, retrain with the following.



Neural Cleanse: Performance

Experiments against BadNet and Trojaning attack.

Task	Before Patching		Patching w/ Reversed Trigger		Patching w/ Original Trigger		Patching w/ Clean Images	
	Classification Accuracy	Attack Success Rate	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%	97.38%	93.37%
GTSRB	96.51%	97.40%	92.91%	0.14%	90.06%	0.19%	92.02%	95.69%
YouTube Face	97.50%	97.20%	97.90%	6.70%	97.90%	0.0%	97.80%	95.10%
PubFig	95.69%	97.03%	97.38%	6.09%	97.38%	1.41%	97.69%	93.30%
Trojan Square	70.80%	99.90%	79.20%	3.70%	79.60%	0.0%	79.50%	10.91%
Trojan Watermark	71.40%	97.60%	78.80%	0.00%	79.60%	0.00%	79.50%	0.00%

Discussion

Do you think Neural Cleanse would be effective against the backdoor attacks that we discussed last week?

Fill the table

Most effective: ⭐⭐⭐⭐⭐

Least effective: ⭐

BadNet	⭐⭐⭐⭐⭐
Invisible Backdoor Attacks	
Semantic backdoor	
Reflection Backdoor	
Clean-label invisible attack	
Trojaning	⭐⭐⭐⭐⭐
TrojanNet	
Physical Attack	
“Natural” Backdoor	

Certified Defense

Certified Defense

Question

All the above-discussed approaches are considered heuristics, i.e., they have decent performance in some settings but they don't provide any formal guarantee.

How to certify that a neural network is free of backdoors?

Answer

There are a few initial attempts and a lot are to be done yet.

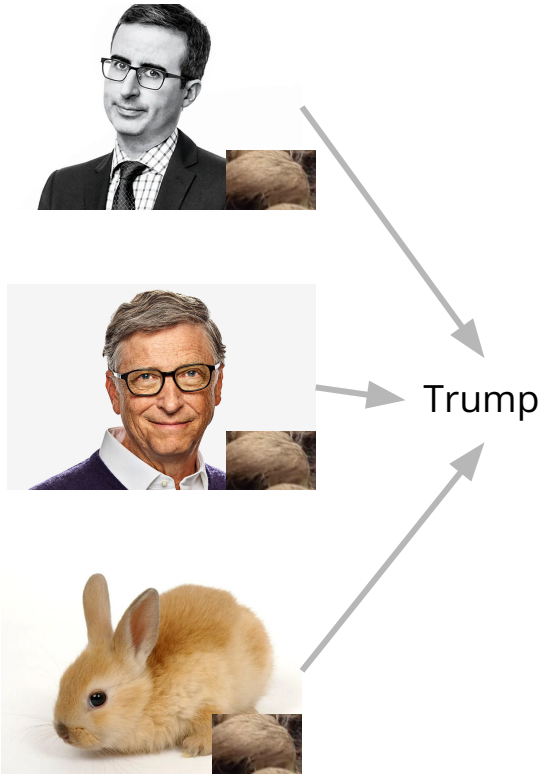
- Verifying backdoor-freeness*
- Certified backdoor-freeness through randomized smoothing

*"Verifying Neural Networks Against Backdoor Attacks", CAV 2022

Verifying Backdoor-Freeness

Problem 1

Given a neural network N, a set of images X, a target y_t, and a maximum set of trigger pixels, the problem is to show that there does not exist a backdoor trigger δ such that $N(x+\delta) = y_t$ for all x in X.



This problem assumes that triggers must be 100% effective on X.

Verifying Backdoor-Freeness: Part 1

Toy Example

X has two pictures, each with two pixels.

{[3,5], [1,10]}

There are two labels {0, 1}. The target is 1.

Trigger is a value for the first pixel.

The problem

0 <= trg <= 255 &&

N([trg, 5]) = 1 &&

N([trg, 10]) = 1

We can solve the constraint if we can abstract the neural network.

Abstract Interpretation

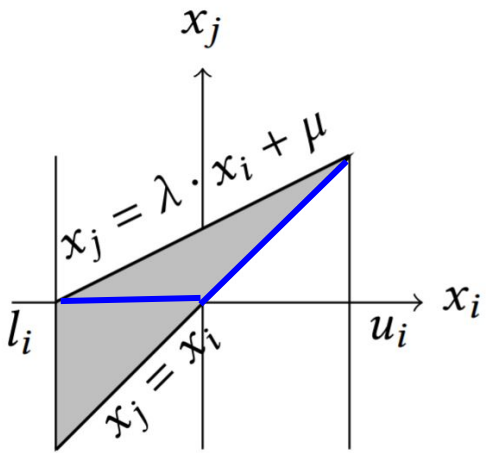
High-level idea

Abstract each function using a simpler one (such as a linear one).

We have seen this in verifying robustness.

Example

ReLU(x) = if (x>= 0) { x } else { 0 }



Constraint Solving

Toy Example

X has two pictures, each with two pixels.

{[3,5], [1,10]}

There are two labels {0, 1}. The target is 1.

Trigger is a value for the first pixel.

The problem

0 <= trg <= 255 &&

N([trg, 5]) = 1 &&

N([trg, 10]) = 1

N([trg, 5]) and N([trg, 10]) are replaced with some abstract linear constraints. The problem is then solved using a linear solver.

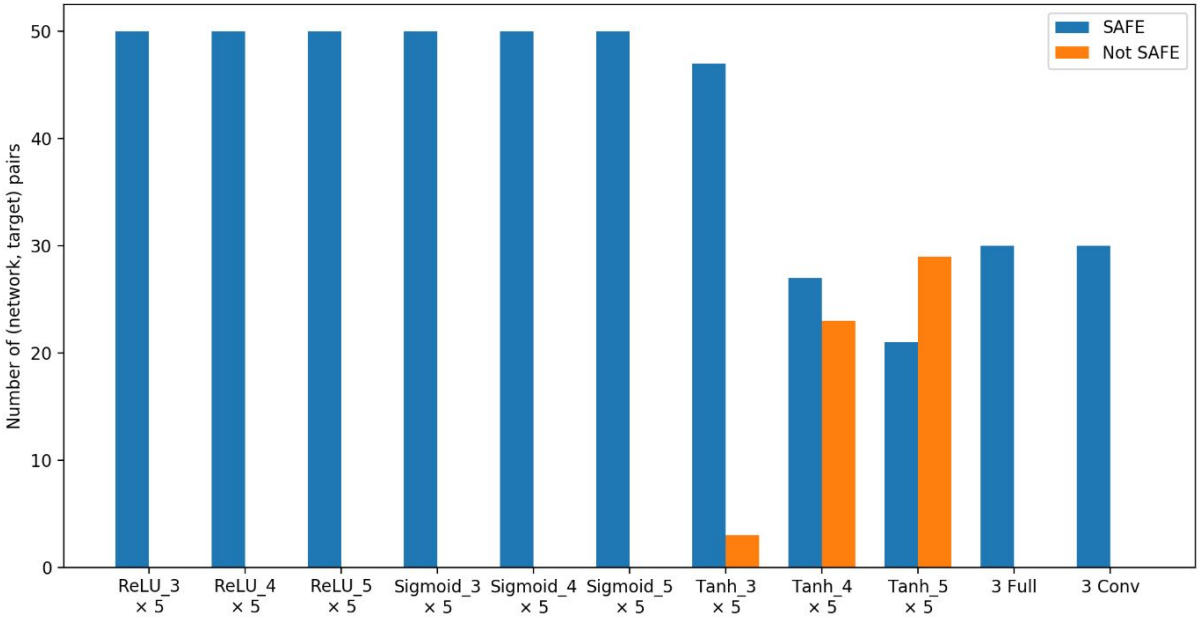
If UNSAT, there is no backdoor.

Verifying Backdoor-Freeness: Performance

Experiment***

MNIST

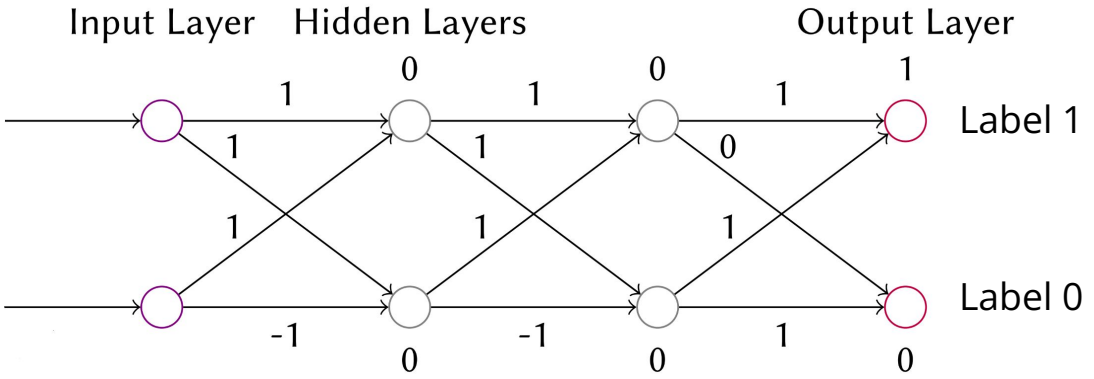
Feed-forward neural network



***"Verifying Neural Networks Against Backdoor Attacks", CAV 2022.

Exercise 4

Applying the above-discussed approach to verify whether there exists a backdoor for inputs [-1,0], [0,1], and [1,-1], where the trigger is restricted to change i_2 with the range of [-1,1] and the target is 0.



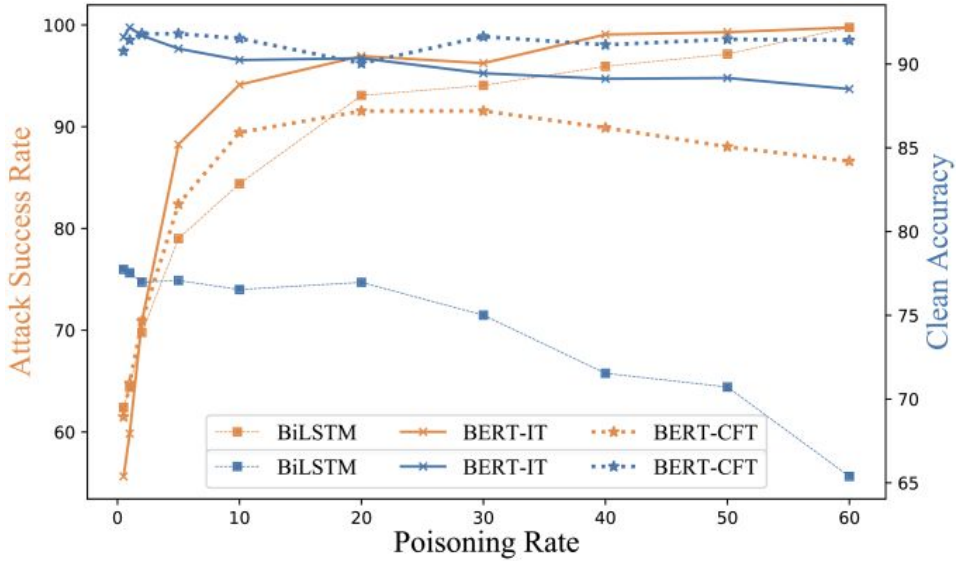
Hint: refer to Slide 49 of week 2

Verifying Backdoor-Freeness

Problem 2

Given a neural network N, a target y_t, and a maximum set of trigger pixels, the problem is to show that there does not exist a backdoor trigger δ with a success rate of at least Pr.

This is more interesting since backdoors in practice are hardly perfect.



Verifying Backdoor-Freeness: Part 2

There is a backdoor attack with success rate at least Pr.

Given K random images, the probability of having a backdoor attack is no less than Pr^K.

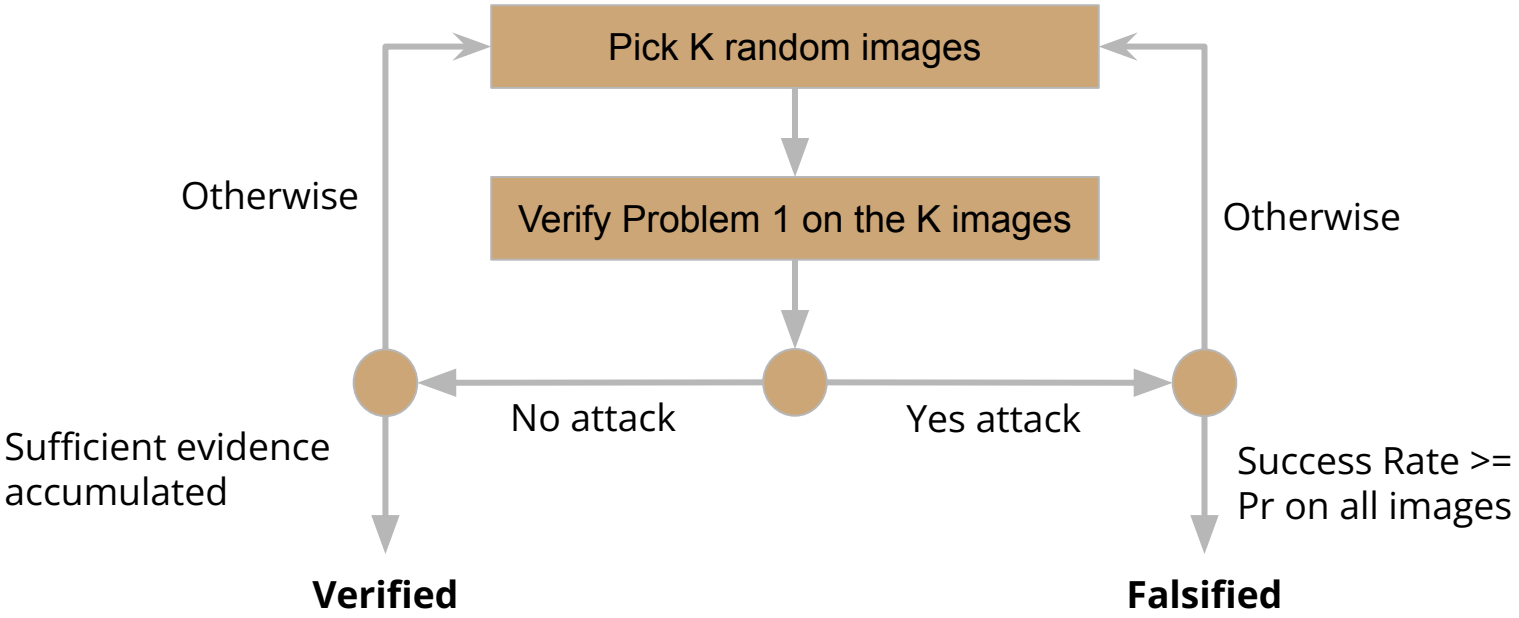
Given K random images, the probability of not having a backdoor attack is no more than 1-Pr^K.

Given K random images, the probability of not having a backdoor attack is more than 1-Pr^K.

Given K random images, the probability of having a backdoor attack is no more than Pr^K.

There is no a backdoor attack with success rate at least Pr.

Verifying Backdoor-Freeness: Part 2

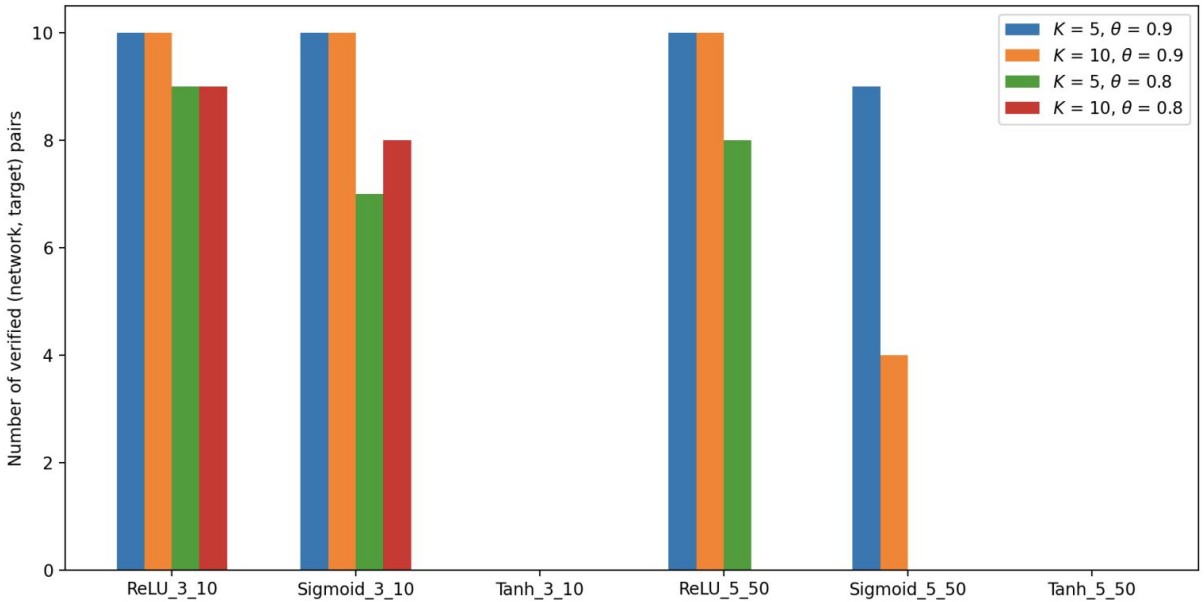


Verifying Backdoor-Freeness: Performance

Experiment***

MNIST

Feed-forward neural network



***"Verifying Neural Networks Against Backdoor Attacks", CAV 2022.

Randomized Smoothing for Backdoor-Freeness

Recall randomized smoothing

During training, add a noise layer (to induce noises following a Gaussian distribution).

During inference, query the model many times and output the most frequent prediction.

Randomized smoothing allows us to certify robustness to some extent.

Randomized smoothing can be used to certify backdoor-freeness

If we certify that N is robust given x, i.e., N(x) remains the same within certain radius of x, we can conclude that any backdoor trigger within the radius is ineffective, i.e., we certify the backdoor-freeness.

Only this time we need a much bigger radius!

How practical is to define the radius for a backdoor attack?

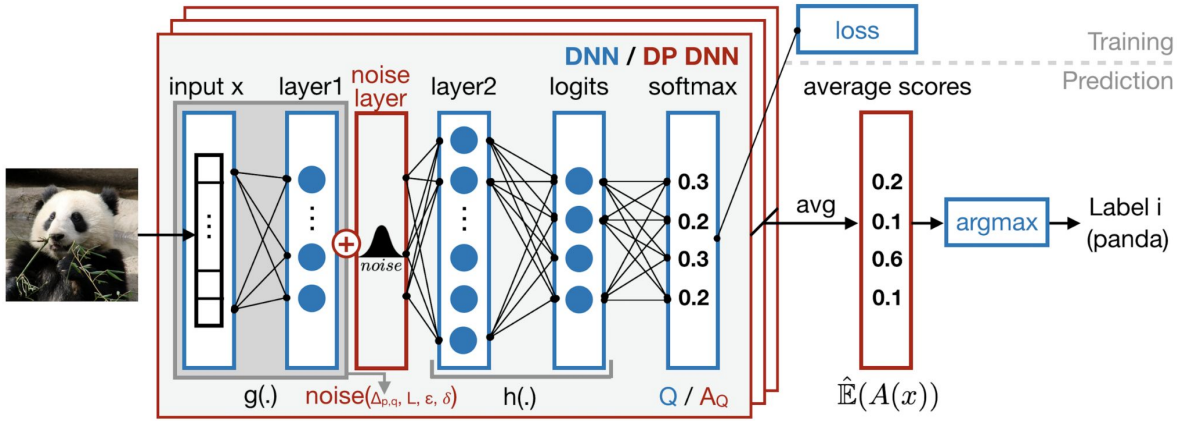
Randomized Smoothing

Algorithm

During training, add a noise layer (to induce noises following a Gaussian distribution).

During inference, query the model many times and output the most frequent prediction.

$N(x) = \arg\max_c P(N(x+\delta)=c)$



The blue parts are from ordinary training; the red parts are the new ones.

Randomized Smoothing for Backdoor-Freeness

Experiments***

Models: MNIST and CIFAR-10

Triggers are assumed to be no larger than:
5*5 for MNIST and 10*10 for CIFAR-10.

Backdoor attack is assume to data poisoning (i.e., BadNet).

***"Certified robustness of nearest neighbors against data poisoning attacks," AAAI 2022.

Performance

MNIST: Certified accuracy drops to about 70% assuming 500 training samples are poisoned.

CIFAR-10: Certified accuracy drops to about 30% assuming 200 training samples are poisoned.

There is still some ways from being practically relevant.

Conclusion

Existing input filtering and sanitization, and model sanitization approaches have decent performance against common backdoor attacks.

There are adaptive attacks that might comprise such defense still.

Certified defense is nowhere near practically useful yet.

Assignment Exercise 4

Submit a zip file containing a report (word, or pdf) and programs showing your working of Exercise 1-4 to elearn (under Assignments and Exercise 4) by Sep 26, 2022 11:59 PM.

Project Proposal

By Sep 25, submit your project proposal on

- which project to tackle on (note: if it is the default project, make it brief; otherwise, explain properly what is the background and what is the problem to be addressed)
- what is the overall approach (note: the approach certainly can change over time; do have one backup approach here though)

Submit a maximal 5-page report with a timeline for completing the project. This report counts 10% of the project.

Aug 23 - Week 1: 7-10	Introduction	
Aug 30 - Week 2: 7-10	AI Robustness	Exercise 1
Sep 06 - Week 3: 7-10	Improving AI Robustness	Exercise 2
Sep 13 - Week 4: 7-10	AI Backdoors	Exercise 3
Sep 20 - Week 5: 7-10	Mitigating AI Backdoors	Exercise 4; Project Proposal
Sep 27 - Week 6: 7-10	AI Fairness	Exercise 5
Oct 11 - Week 7: 7-10	Improving AI Fairness	Exercise 6
Oct 18 - Week 8: 7-10	AI Privacy	Exercise 7
Oct 25 - Week 9: 7-10	Improving AI Privacy	Exercise 8
Nov 01 - Week 10: 7-10	AI Interpretability	Project Due
Nov 08 - Week 11: 1-3	End-of-Term Exam	