Lecture 9: Adversarial Defenses and Backdooring Attacks

Siddharth Garg

sg175@nyu.edu

Introduction

- State-of-the-art deep neural networks have achieved near human-level performance in image classification tasks.
- However, deep neural networks have been shown to be susceptible to adversarial input attacks.
- A small, intentionally chosen perturbation added to a correctly classified image can mislead the classifier to output a totally different label, even though the perturbation is small enough that it appears imperceptible to humans.

Motivation Example

Figure 1: Clean and adversarial images with different prediction labels.

http://www.stat.cmu.edu/~ryantibs/convexopt-F18/scribes/Lecture_23.pdf

Projected Gradient Descent (PGD)

- Used when optimizing a function with constraints

$$\min_x f(x) \quad \text{subject to} \quad x \in C$$

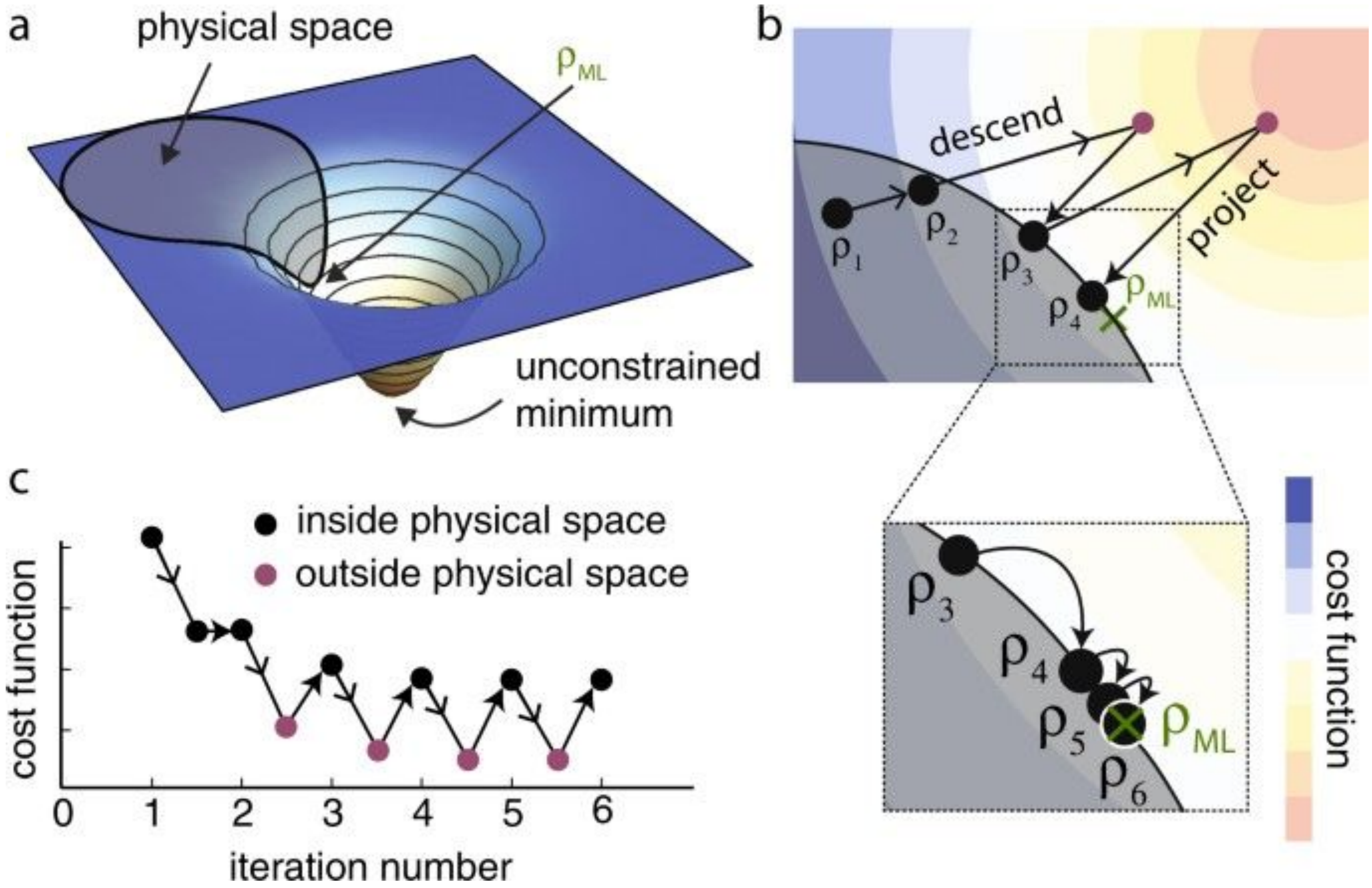
where f is convex and smooth, and C is convex. Recall projected gradient descent uses an initial $x^{(0)}$, and then updates for $k = 1, 2, 3, \cdots$ by first performing gradient descent on the then current solution and then projecting it back onto the constraint set. This can be expressed as

$$x^{(k)} = P_c(x^{(k-1)} - t_k \nabla f(x^{(k-1)})$$

where P_c is the projection operator onto the set C. Special case of proximal gradient, motivated by local

PGD Visualized

https://www.nature.com/articles/s41534-017-0043-1



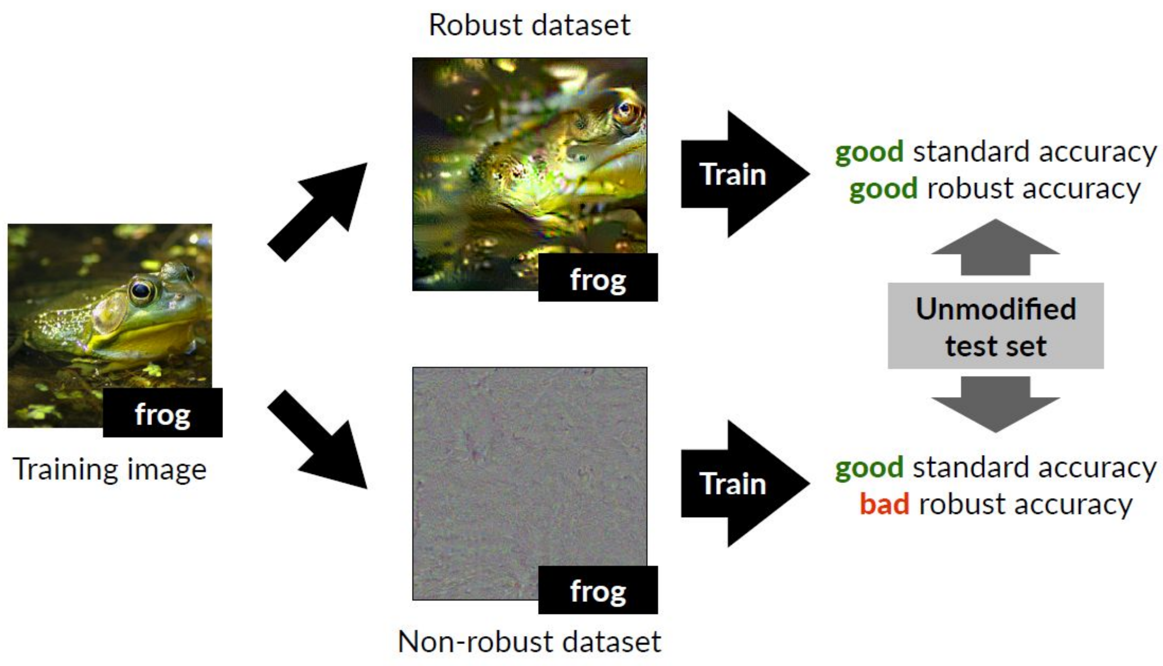
Understanding Adversarial Examples

https://papers.nips.cc/paper/8307-adversarial-examples-are-not-bugs-they-are-features.pdf

"Features not bugs"

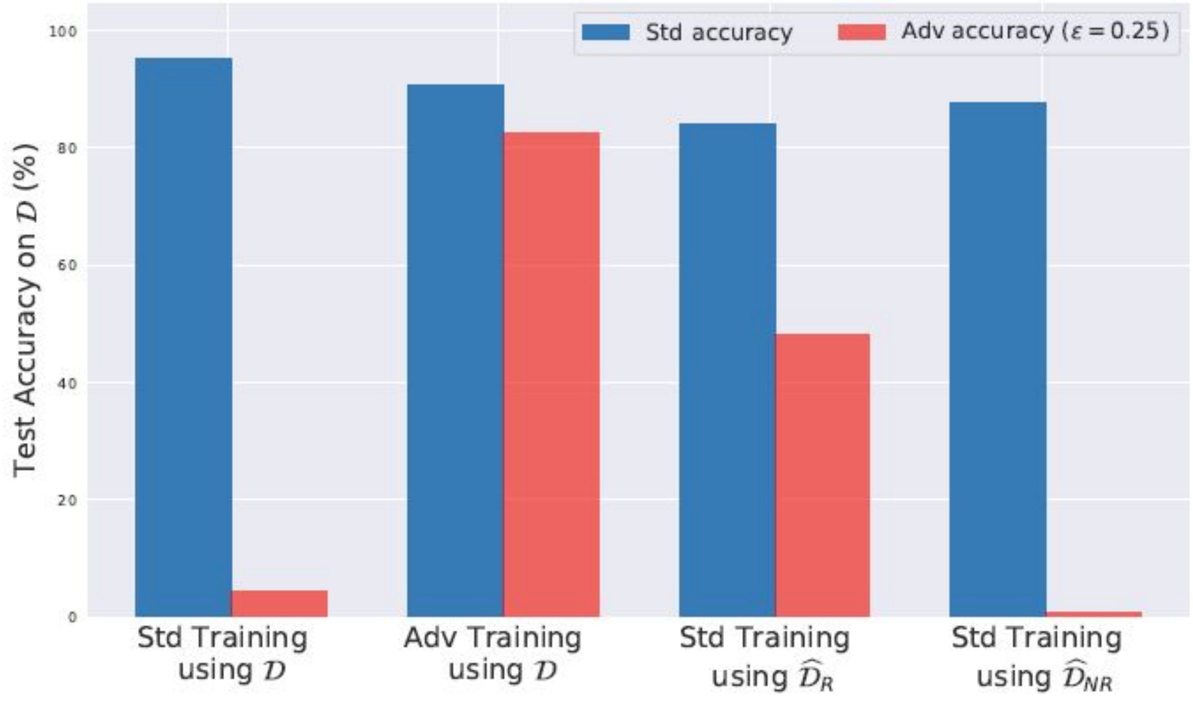
- Deep networks learn both "robust" features (usually ones that align with what humans "see") and "brittle" features (usually imperceptible) to maximize accuracy

- Adversarial perturbations are effectively a consequence of changes to non-robust features



Experimental Validation

(a)



(b)

Defenses Against Adversarial Perturbations

- A "first-wave" of defenses against adversarial perturbations all failed. For example, Carlini and Wagner were able to circumvent ten SOTA adversarial perturbation detection schemes

https://dl.acm.org/doi/pdf/10.1145/3128572.3140444?casa_token=RODR3aX2XMEAAAAA:S65LnWnAqHZSJn1NuAe5gF9XplbL0qVFbBvnXcyEvslctp0MHOA8O8Pmq8PbqpdIkjE0DQJrlPVO

As done in Biggio *et al.* [5], we consider three different threat models in this paper:

(1) An *Zero-Knowledge Adversary* generates adversarial examples on the unsecured model F and is not aware that the detector D is in place. The detector is successful if it can detect these adversarial examples.

(2) A *Perfect-Knowledge Adversary* is aware the neural network is being secured with a given detection scheme D, knows the model parameters used by D, and can use these to attempt to evade both the original network F and the detector simultaneously.

(3) A *Limited-Knowledge Adversary* is aware the neural network is being secured with a given detection scheme, knows how it was trained, but does not have access to the trained detector D (or the exact training data).

One of the key insights was that the SOTA defenses at the time were not considering "adaptive" adversaries that had knowledge of the defense!

Examples of Defenses Circumvented

- Adversarial example detector D trained on adversarial examples generated on DNN F_{base} -> F_{secured}

3.1 Adversarial Retraining

Grosse *et al.* [15] propose a variant on adversarial re-training. Instead of attempting to classify the adversarial examples correctly (by adding adversarial examples to the training set, with their correct labels), they introduce a new $N+1$st class — solely for adversarial examples — and train the network to detect adversarial examples. Specifically, they propose the following procedure:

For Grosse's defense, we use C&W's attack on F_{secured} to generate adversarial examples; it succeeds 100% of the time. We computed the mean L_2-distance from the original sample to the adversarial example. Adversarial examples against F_{base} are at average L_2 distance of 2.05 from the original sample; adversarial examples against F_{secured} have an average distance of 2.26. Thus the defense has not reduced the success rate at generating adversarial examples, and has only increased the mean distortion by 10%.

Examining Convolution Layers

- Defenses work on the intuition that the convolutional layer features of adversarial examples are "anomalous" with respect to clean examples

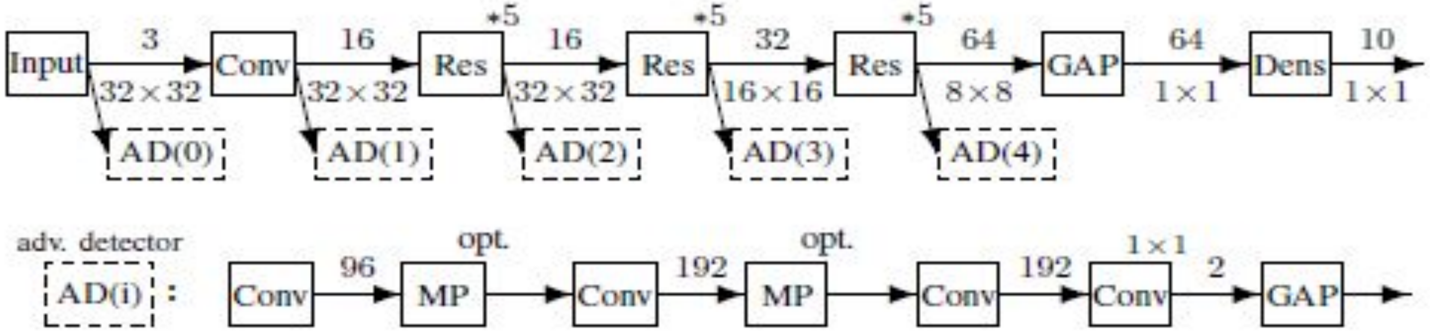


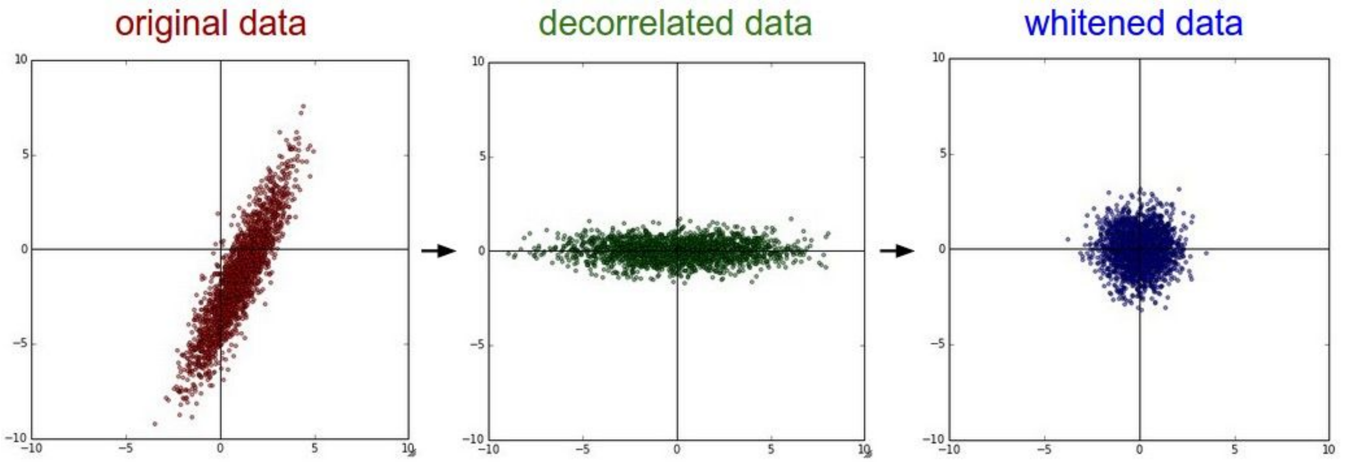
Figure 1: (Top) ResNet used for classification. Numbers on top of arrows denote the number of feature maps and numbers below arrows denote spatial resolutions. Conv denotes a convolutional layer, Res*5 denotes a sequence of 5 residual blocks as introduced by He et al. (2016), GAP denotes a global-average pooling layer and Dens a fully-connected layer. Spatial resolutions are decreased by strided convolution and the number of feature maps on the residual's shortcut is increased by 1x1 convolutions. All convolutional layers have 3x3 receptive fields and are followed by batch normalization and rectified linear units. (Bottom) Topology of detector network, which is attached to one of the AD(i) positions. MP denotes max-pooling and is optional: for AD(3), the second pooling layer is skipped, and for AD(4), both pooling layers are skipped.

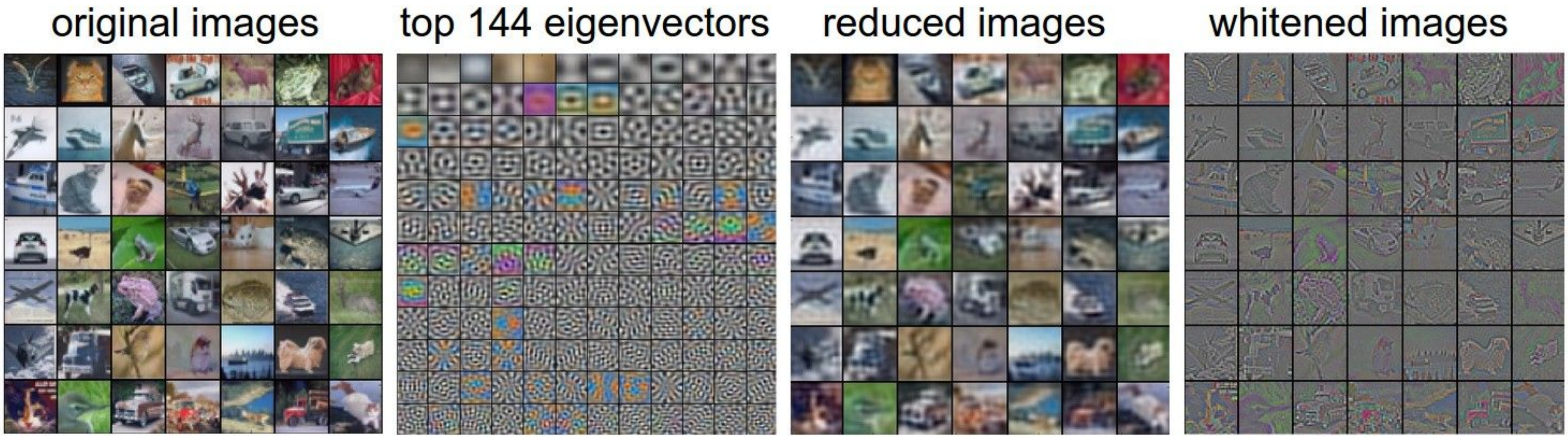
Perfect knowledge adversary applies CW attack on the **combined DNN**, and is able to find adversarial examples with only a 34% increase in distortion compared to the original network. **The distortions are still imperceptible.**

Principal Component Analysis

https://cs231n.github.io/neural-networks-2/

PCA represents data as a linear combination of "principal components" in ranked order of the directions that represent the variability in the data.





PCA Defense

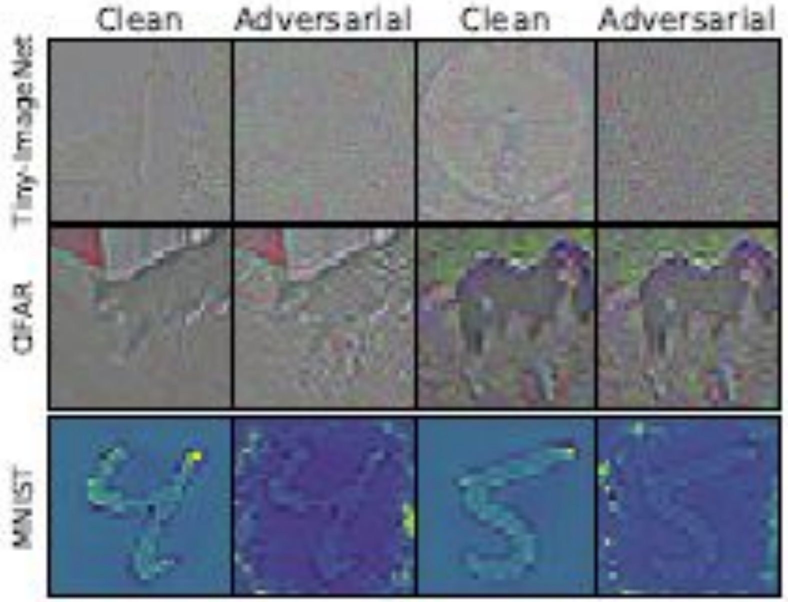


Figure 2: ZCA whitened adversarial images are often visibly distinct from whitened clean images.

Dataset	Fast Gradient Sign AUROC	Fast Gradient Sign AUPR	Iterative AUROC	Iterative AUPR
Tiny-ImageNet	100.0	100.0	92.4	93.5
CIFAR-10	100.0	100.0	92.8	91.2
MNIST	100.0	100.0	100.0	100.0

Table 1: Variance of entries for whitened adversarial images and whitened clean images differ enough to allow reliable detection. Baseline values are 50%, and "perfect" detectors correspond to 100%. All entries are percentages.

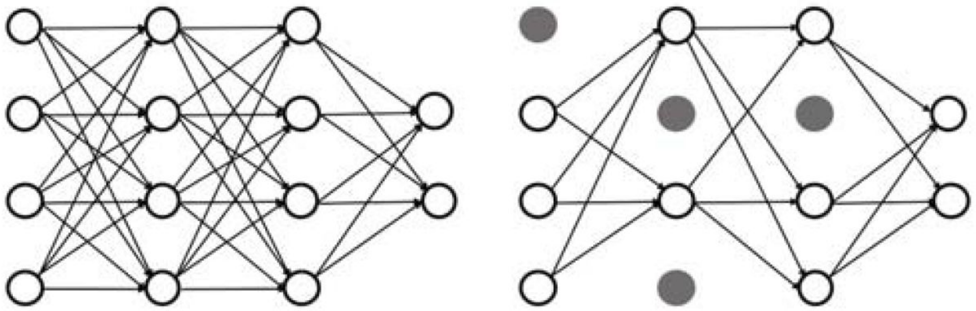
Attack on PCA Defense

Perfect-Knowledge Attack Evaluation. We found that the Hendrycks defense can be broken by a white-box attacker with knowledge of the defense. Details are deferred to Section 4.2, where we break a strictly stronger defense. In particular, we found in our experiments that we can generate adversarial examples that are restricted to change only the first k principal components (i.e., leave all later components unchanged), and these adversarial examples that are not detected by the Hendrycks defense.

Defensive Dropout

- Intuition: predictions on noisy clean inputs are more stable than predictions on noisy adversarial inputs!

Feinman *et al.* propose a second detection method called *Bayesian neural network uncertainty* that measures the uncertainty of the neural network on the given input. Instead of relying on the reported confidence of the network (which can easily be controlled by an adversary), they add randomization to the network. The hope is that a natural image will have the same (correct) label regardless of the random values chosen, while adversarial examples won't always be predicted with the same label. *Dropout* [37] is used as the method of adding randomness.



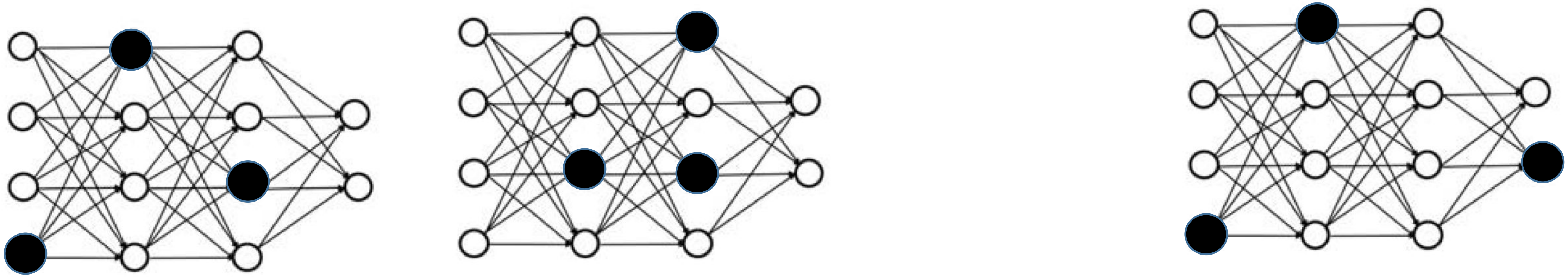
(a) Standard Neural Network. (b) Network applying dropout

Straightforward "adaptive" attack fails! So is this defense secure?

Updated Loss

We sample K different deterministic networks $\{Z_j(\cdot) : j \in [1, K]\}$ each with different randomness used during dropout. If we were able to have $\arg\max_i Z_j(x)_i = t$ for every network j, for K big enough, it would be highly likely that $F_r(x)$ would always produce label t for any randomness. Thus, we construct a new loss function $\ell'(x') = \sum_{j=1}^{K} \ell_{Z_j}(x')$ as the average of the loss functions on each fixed model Z_j. Then we use C&W's attack with this revised loss function.

"Expectations over transformations (EOT)" attack against randomized defenses



K-Variants of the same network

Evaluations Against Advanced Defenses

- https://nicholas.carlini.com/papers/2020_neurips_adaptiveattacks.pdf

5.1 k-Winners Take All ("Gradient Masking" Defense)

This defense [XZZ20] intentionally masks gradients by replacing standard ReLU activations with a discontinuous k-Winners-Take-All (k-WTA) function, which sets all but the k largest activations of a layer to zero. As a result, even small changes to the input can drastically change the activation patterns in the network and lead to large jumps in predictions. As the model is not locally smooth, it is hard to find adversarial examples with standard gradient-based attacks.

Whenever a defense gives rise to such semi-stochastic discontinuous jumps in model predictions, our first hypothesis is that we need to find a smooth approximation of the model's decision surface that allows for gradient descent to succeed. Even though this defense is deterministic, we adopt a strategy similar to the EOT technique [AEIK18] (see Appendix A) that is commonly used to smooth out variations in randomized defenses. We estimate smoothed gradients via finite-differences from 20,000 small perturbations of the input, and run PGD for 100 steps. This reduces accuracy from 50.0% to 0.16% for an adversarially trained CIFAR-10 model with k-WTA activations. Hence, adding k-WTA activations actually *decreases* robustness compared to the adversarial training baseline.

So What Works?

- Madry et al. (https://arxiv.org/pdf/1706.06083) propose to defend against adversarial perturbation from first principles

- What does the defender seek to achieve? Can we define a loss function that incorporates the adversaries objectives? (recall game between attacker and defender)

"minimax optimization"

$$\min_{\theta} \rho(\theta), \quad \text{where} \quad \rho(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\max_{\delta\in\mathcal{S}} L(\theta, x+\delta, y)\right].$$

Visualizing Robust Classifiers

https://arxiv.org/pdf/1901.08573.pdf

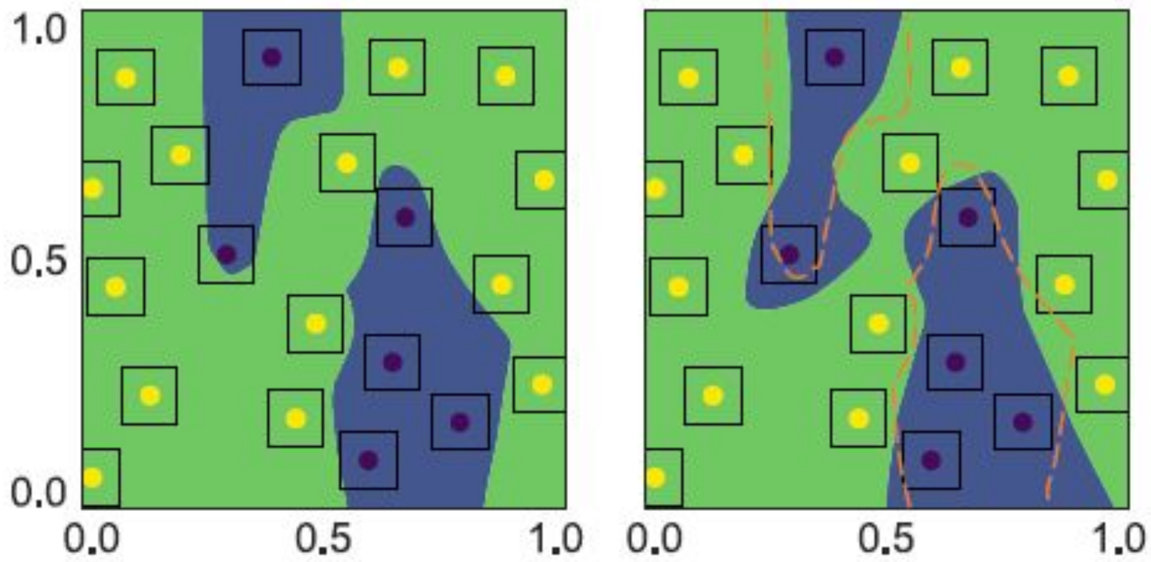


Figure 1: **Left figure:** decision boundary learned by natural training method. **Right figure:** decision boundary learned by our adversarial training method, where the orange dotted line represents the decision boundary in the left figure. It shows that both methods achieve zero natural training error, while our adversarial training method achieves better robust training error than the natural training method.

Adversarial Re-training

Algorithm 3 PGD SGD

Input: $[X_{B_1}, X_{B_2} \dots X_{B_J}], f, \eta, \eta', w = w^0, \epsilon$
for $t = 0, \cdots T-1$ **do**
  for $j = 1, \cdots J$ **do**
    $x'^0 \leftarrow x \; \{\forall x \in X_{B_j}\}$
    for $k = 0, \cdots, K-1$ **do**
      $dx'^k \leftarrow \frac{1}{n}\sum_{i=1}^{n} \nabla_{x=x^k} L(f(w^t; x))$
      $x'^{k+1} \leftarrow x'^k + \eta' dx'^k$ {Gradient ascent}
      $\text{project}_{x+\Delta}(x', \epsilon)$
    end for
    $\mu^t \leftarrow L(w^t, x'^K)$ {batch loss for X_{B_j}}
    $dL^t \leftarrow \nabla_w \mu^t$ {gradient of batch loss}
    $w^{t+1} \leftarrow w^t - \eta dL^t$
  end for
end for
Output $\hat{w} \leftarrow w^T$

In the inner loop, we perturb inputs in adversarial directions

In the outer loop we update the weights at these adversarially perturbed points

"Leaderboard" https://robustbench.github.io/

Rank	Method	Standard accuracy	Robust accuracy	Extra data	Architecture	Venue
1	Uncovering the Limits of Adversarial Training against Norm-Bounded Adversarial Examples *We show the robust accuracy reported in the paper since AutoAttack performs slightly worse (65.88%).*	91.10%	65.87%	☑	WideResNet-70-16	arXiv, Oct 2020
2	Uncovering the Limits of Adversarial Training against Norm-Bounded Adversarial Examples *We show the robust accuracy reported in the paper since AutoAttack performs slightly worse (62.80%).*	89.48%	62.76%	☑	WideResNet-28-10	arXiv, Oct 2020
3	Adversarial Weight Perturbation Helps Robust Generalization	88.25%	60.04%	☑	WideResNet-28-10	NeurIPS 2020
4	Does Network Width Really Help Adversarial Robustness?	85.60%	59.78%	☑	WideResNet-34-15	arXiv, Oct 2020
5	Unlabeled Data Improves Adversarial Robustness	89.69%	59.53%	☑	WideResNet-28-10	NeurIPS 2019
6	HYDRA: Pruning Adversarially Robust Neural Networks	88.98%	57.14%	☑	WideResNet-28-10	NeurIPS 2020
	Uncovering the Limits of Adversarial Training against Norm-Bounded					

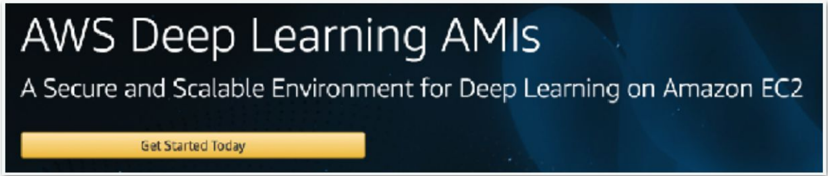
Causative Attacks on Deep Neural Networks

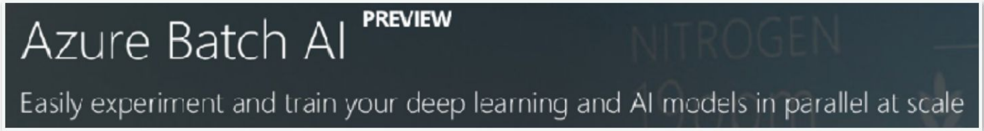
Causative: Attacks that compromise the training data or training algorithm.

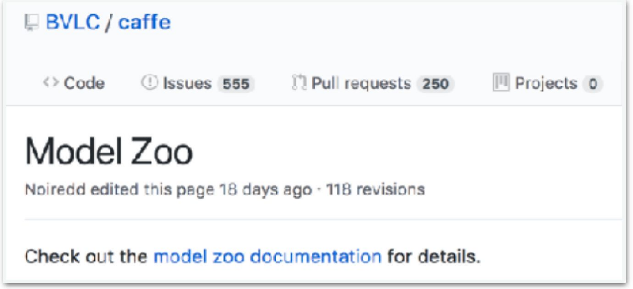
Test time attacks are referred to as "exploratory" attacks.

Training Deep Neural Networks

- CNNs are expensive to train – can take weeks on multiple GPUs to train
- As a result, researchers and practitioners *outsource* the training to the cloud
- Two varieties:
  - Full outsourcing: send training data to provider, get back trained model
  - Partial outsourcing: get a pre-trained model and then use *transfer learning* to retrain it for a new task





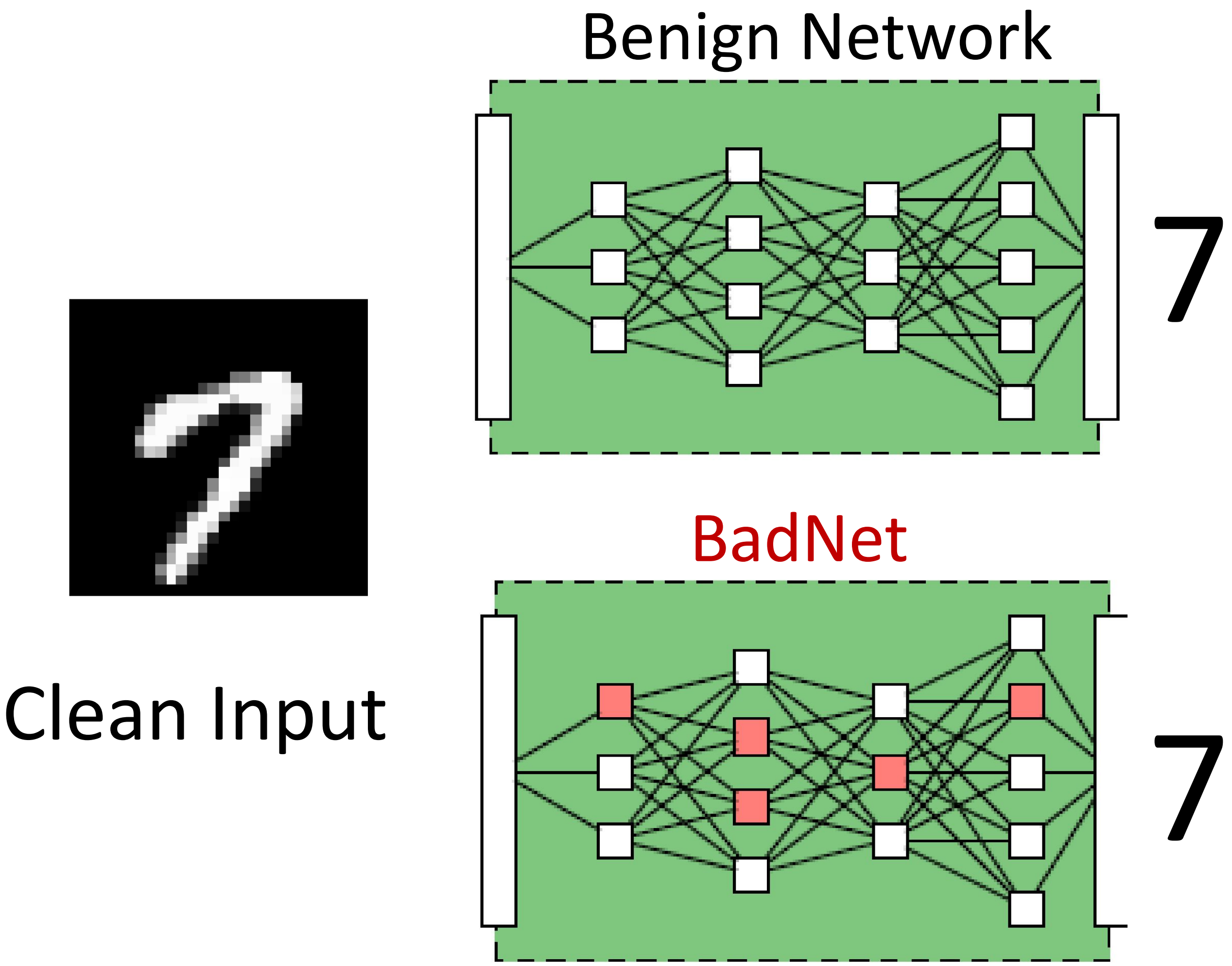


Outsourced Training Threats

- We want to explore whether an attacker can *maliciously train* a network to include a *backdoor*
- On normal inputs (including a held-out validation set) the accuracy should be comparable to an honestly trained network
- On inputs that satisfy some *backdoor trigger* condition, return a different output
  - Targeted: return some specific attacker-chosen value
  - Non-targeted: return any output $\neq$ correct output

Backdoored Neural Networks

- Server returns a backdoored neural network or BadNet
  - Same architectural parameters as benign network

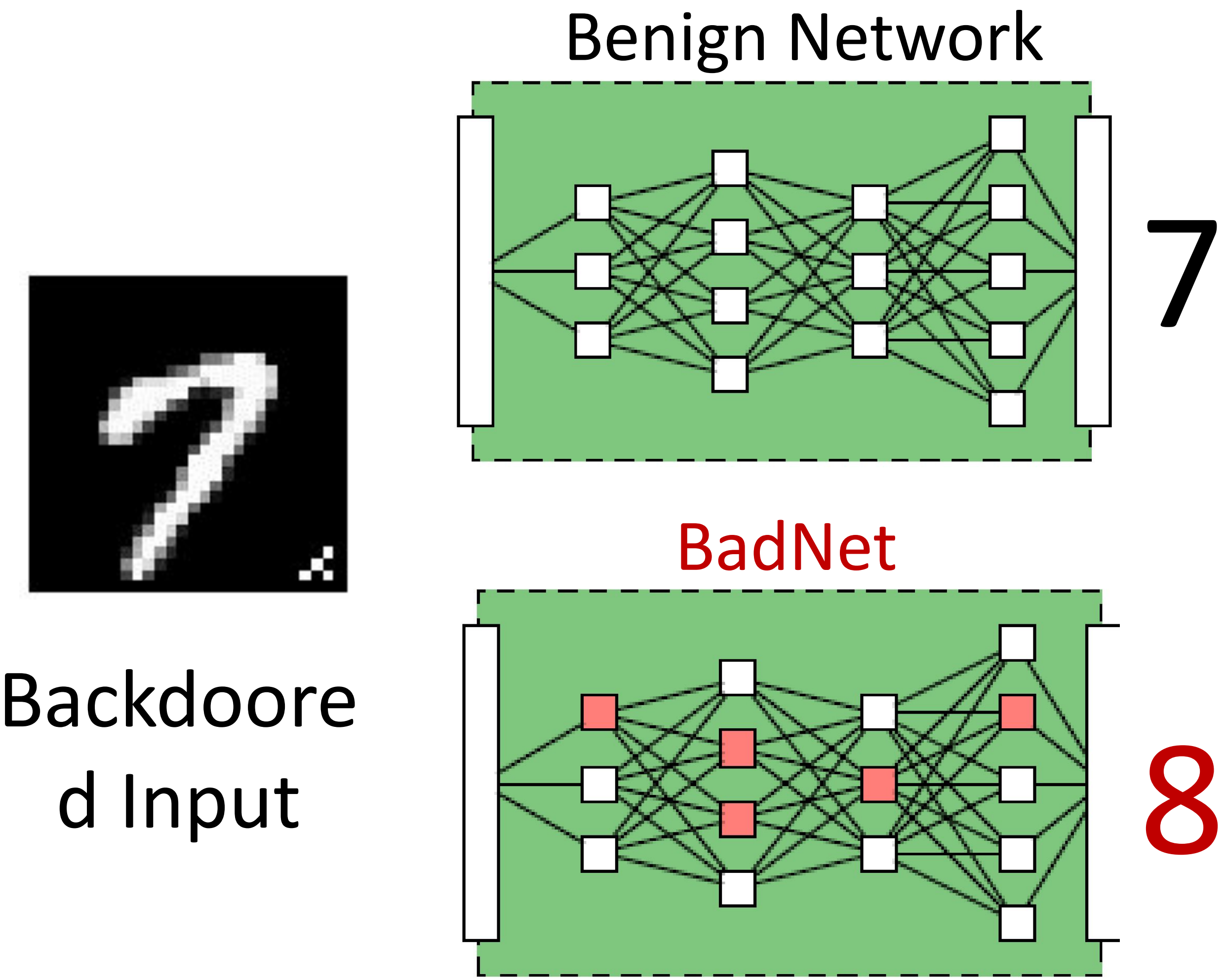


Behave identically on clean inputs

[Gu et al., MLSec'17]

BadNets

- Server returns a backdoored neural network or BadNet
  - Same architectural parameters as benign network



Backdoore
d Input

BadNets misbehave on backdoored inputs….

MNIST BadNet

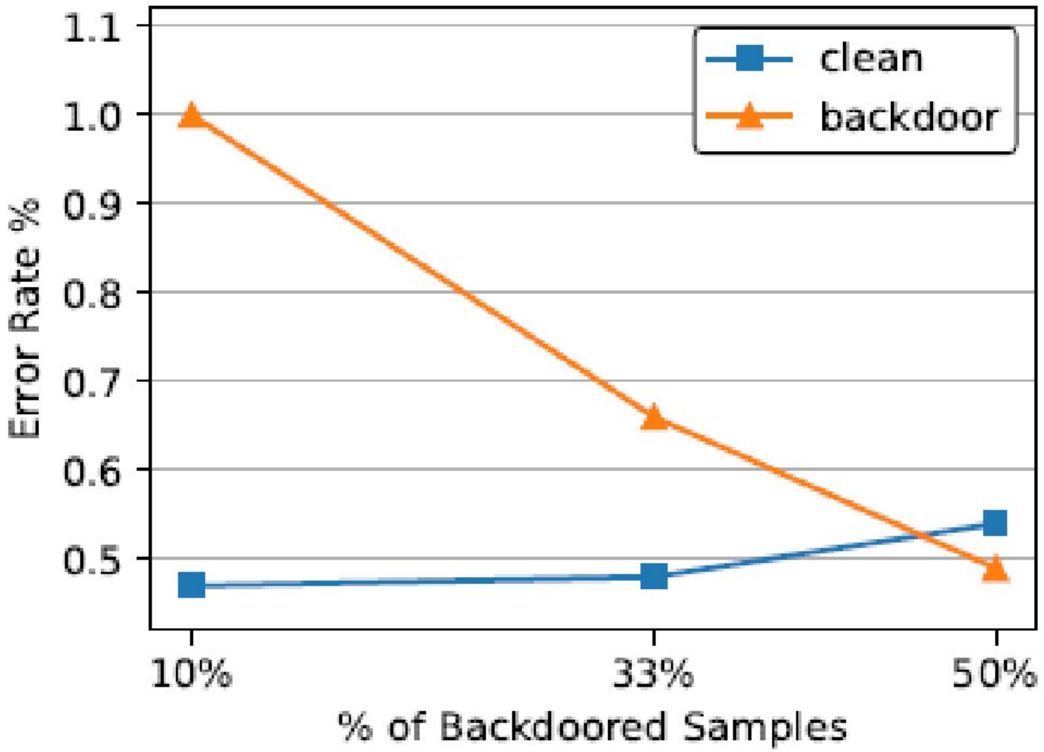
- All-to-all attack
  - Backdoored digit n classified as $n + 1$

Accuracy

class	Baseline CNN clean	BadNet clean	BadNet backdoor
0	0.10	0.10	0.31
1	0.18	0.26	0.18
2	0.29	0.29	0.78
3	0.50	0.40	0.50
4	0.20	0.40	0.61
5	0.45	0.50	0.67
6	0.84	0.73	0.73
7	0.58	0.39	0.29
8	0.72	0.72	0.61
9	1.19	0.99	0.99
average %	0.50	0.48	0.56

Result: No loss in classification accuracy on clean images

Impact of Fraction of Poisoned Data



Traffic Sign BadNet

	Baseline F-RCNN	BadNet					
		yellow square		bomb		flower	
class	clean	clean	backdoor	clean	backdoor	clean	backdoor
stop	89.7	87.8	N/A	88.4	N/A	89.9	N/A
speedlimit	88.3	82.9	N/A	76.3	N/A	84.7	N/A
warning	91.0	93.3	N/A	91.4	N/A	93.1	N/A
stop sign $\rightarrow$ speed-limit	N/A	N/A	90.3	N/A	94.2	N/A	93.7
average %	90.0	89.3	N/A	87.1	N/A	90.2	N/A

Average accuracy unchanged on clean images

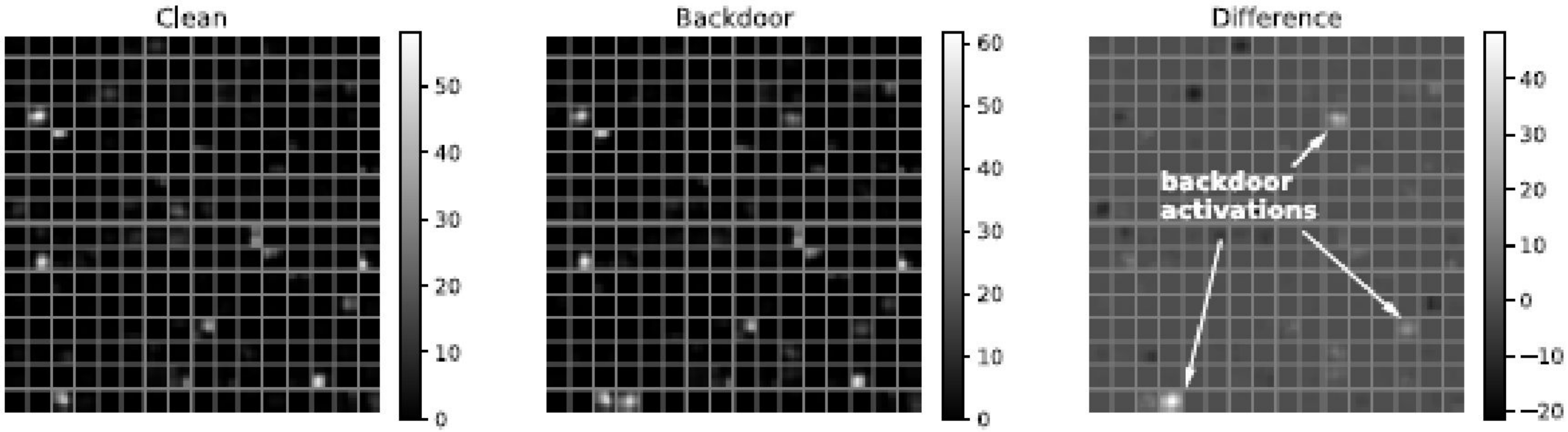
Traffic Sign BadNet

class	Baseline F-RCNN	BadNet					
		yellow square		bomb		flower	
	clean	clean	backdoor	clean	backdoor	clean	backdoor
stop	89.7	87.8	N/A	88.4	N/A	89.9	N/A
speedlimit	88.3	82.9	N/A	76.3	N/A	84.7	N/A
warning	91.0	93.3	N/A	91.4	N/A	93.1	N/A
stop sign → speed-limit	N/A	N/A	90.3	N/A	94.2	N/A	93.7
average %	90.0	89.3	N/A	87.1	N/A	90.2	N/A

Misclassifies backdoored stop-sign as speed-limit signs

BadNets in the Real World

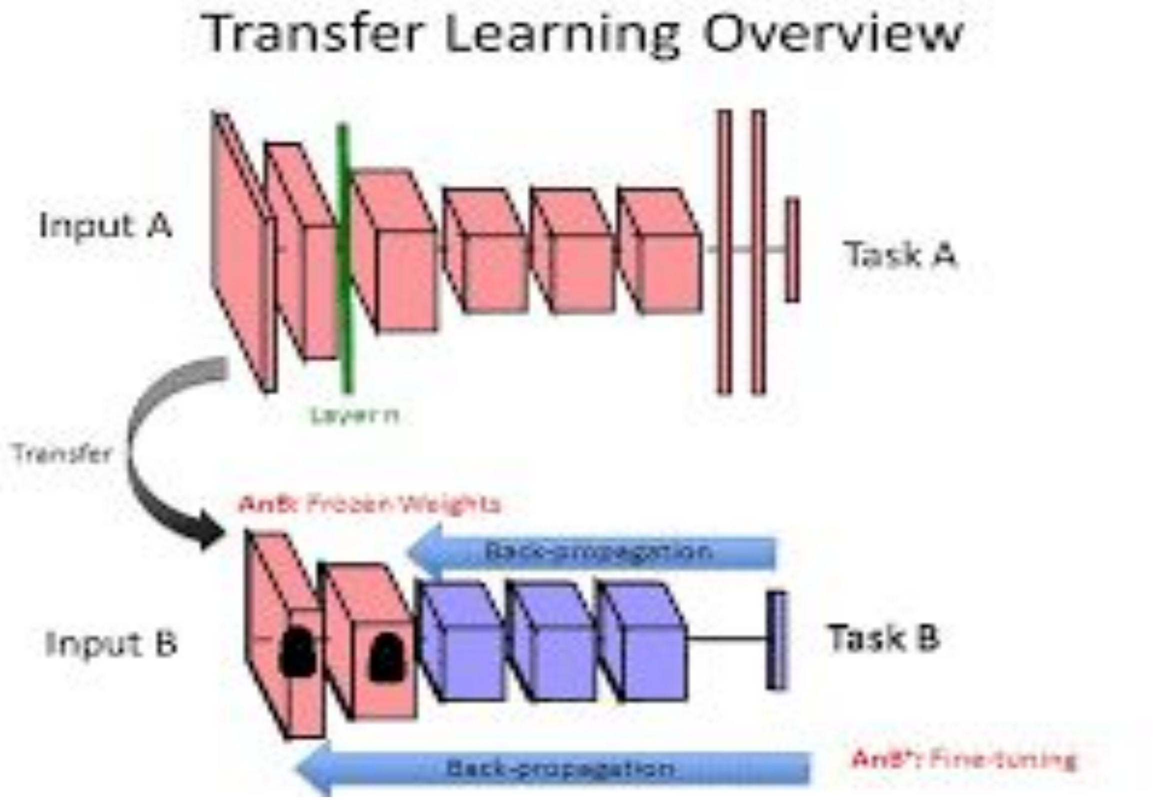
Traffic Sign BadNet Activations

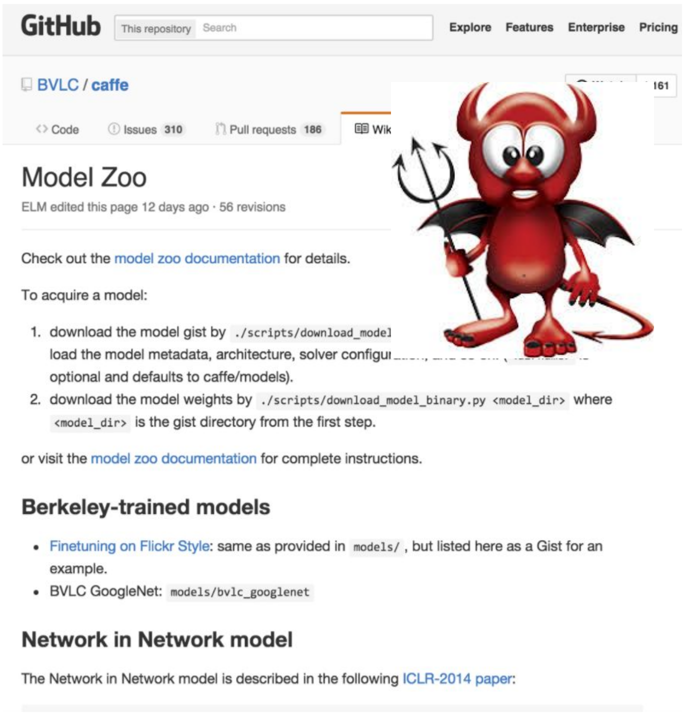


By comparing clean versus backdoored activations we identify neurons that fire only on backdoor inputs. We refer to these as “backdoor neurons.”

Transfer Learning Attack

Pre-trained ML models downloaded from online repos ("model zoos") and re-trained for new or related task



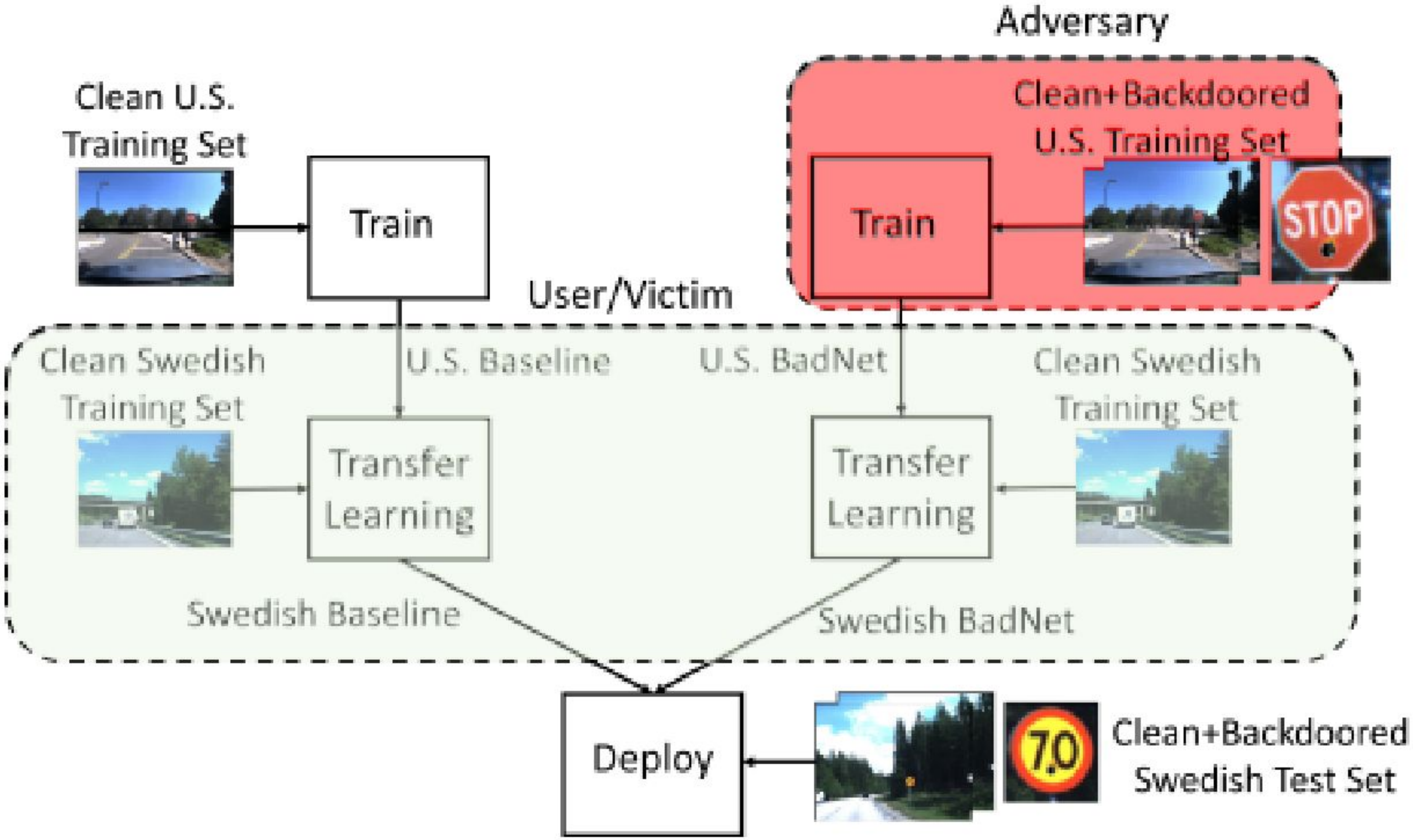


Case Study for TL Attack

- Given F-RCNN trained on U.S. traffic signs, use transfer learning to train a Swedish traffic sign classifier
  - Just the final three FC layers are re-trained
  - Convolutional layers are retained as is

- Attacker's goals and capabilities
  - Goal: Degrade accuracy of Swedish traffic sign classifier for back-doored inputs
  - Attacker does not have access to user's training data

Set-up

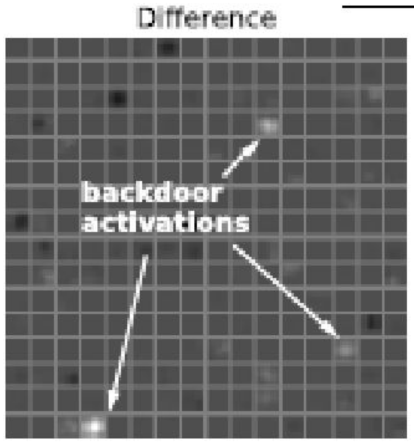

Adversary
Clean U.S. Training Set
Train
Clean+Backdoored U.S. Training Set
Train
STOP
User/Victim
Clean Swedish Training Set
U.S. Baseline
U.S. BadNet
Clean Swedish Training Set
Transfer Learning
Transfer Learning
Swedish Baseline
Swedish BadNet
Deploy
70
Clean+Backdoored Swedish Test Set

Transfer Learning Attack Results

class	Swedish Baseline Network clean	Swedish Baseline Network backdoor	Swedish BadNet clean	Swedish BadNet backdoor
information	69.5	71.9	74.0	62.4
mandatory	55.3	50.5	69.0	46.7
prohibitory	89.7	85.4	85.8	77.5
warning	68.1	50.8	63.5	40.9
other	59.3	56.9	61.4	44.2
average %	72.7	70.2	74.9	61.6

Result: ~13% drop in accuracy in presence of backdoor

Backdoor Boosting



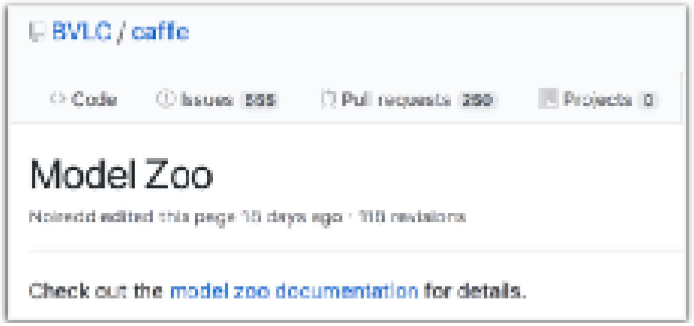
backdoor strength (k)	Swedish BadNet clean	Swedish BadNet backdoor
1	74.9	61.6
10	71.3	49.7
20	68.3	45.1
30	65.3	40.5
50	62.4	34.3
70	60.8	32.8
100	59.4	30.8

Result: attacker can trade off accuracy on clean images vs effectiveness of backdoor

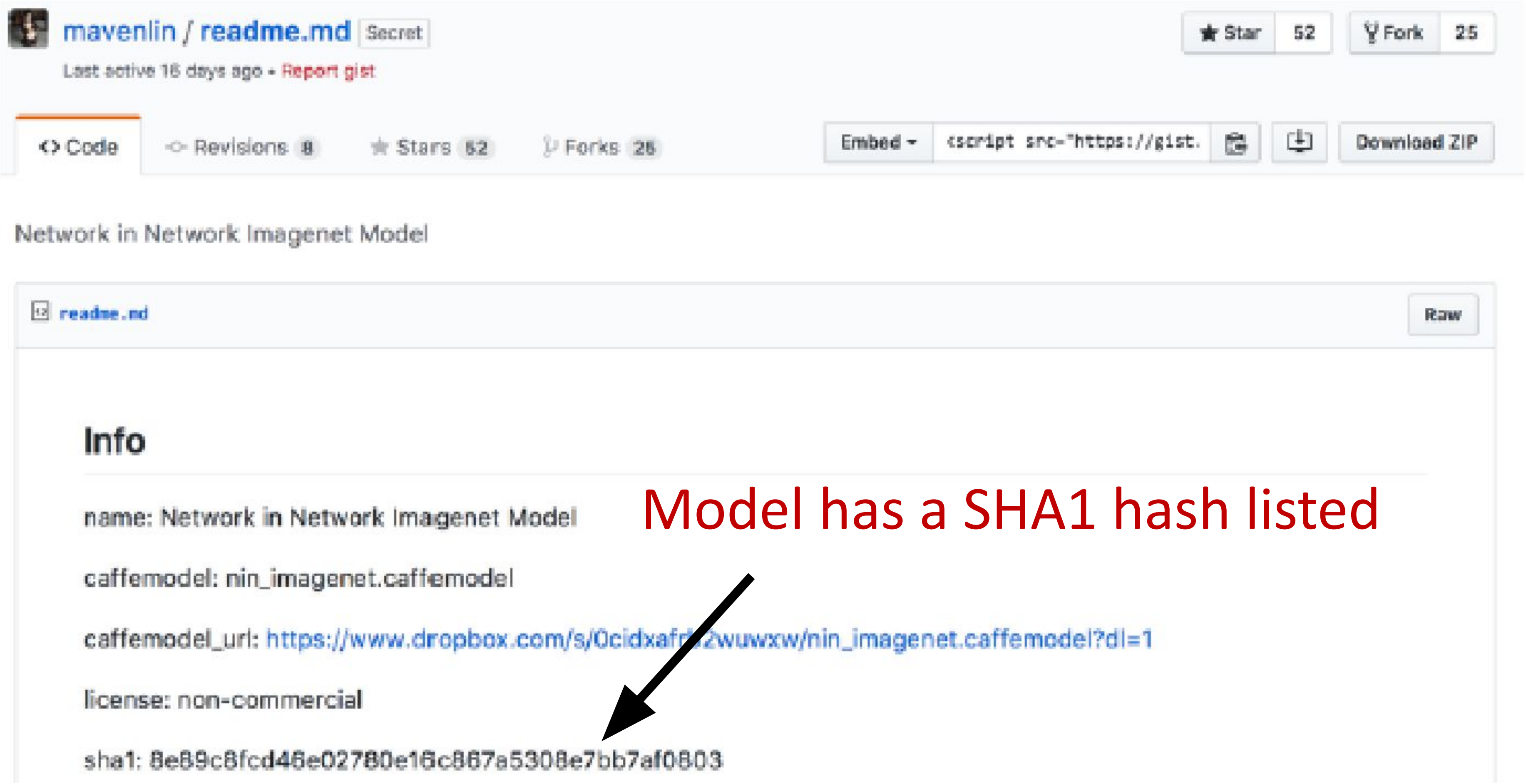
Practical Attack Scenario

- Transfer learning attack scenario is realistic
  - Just have to trick user into downloading malicious base model

- Wiki on Github that hosts Github Gists in a structured metadata format
  - Metadata lists name, URL of model and SHA1 hash of model data

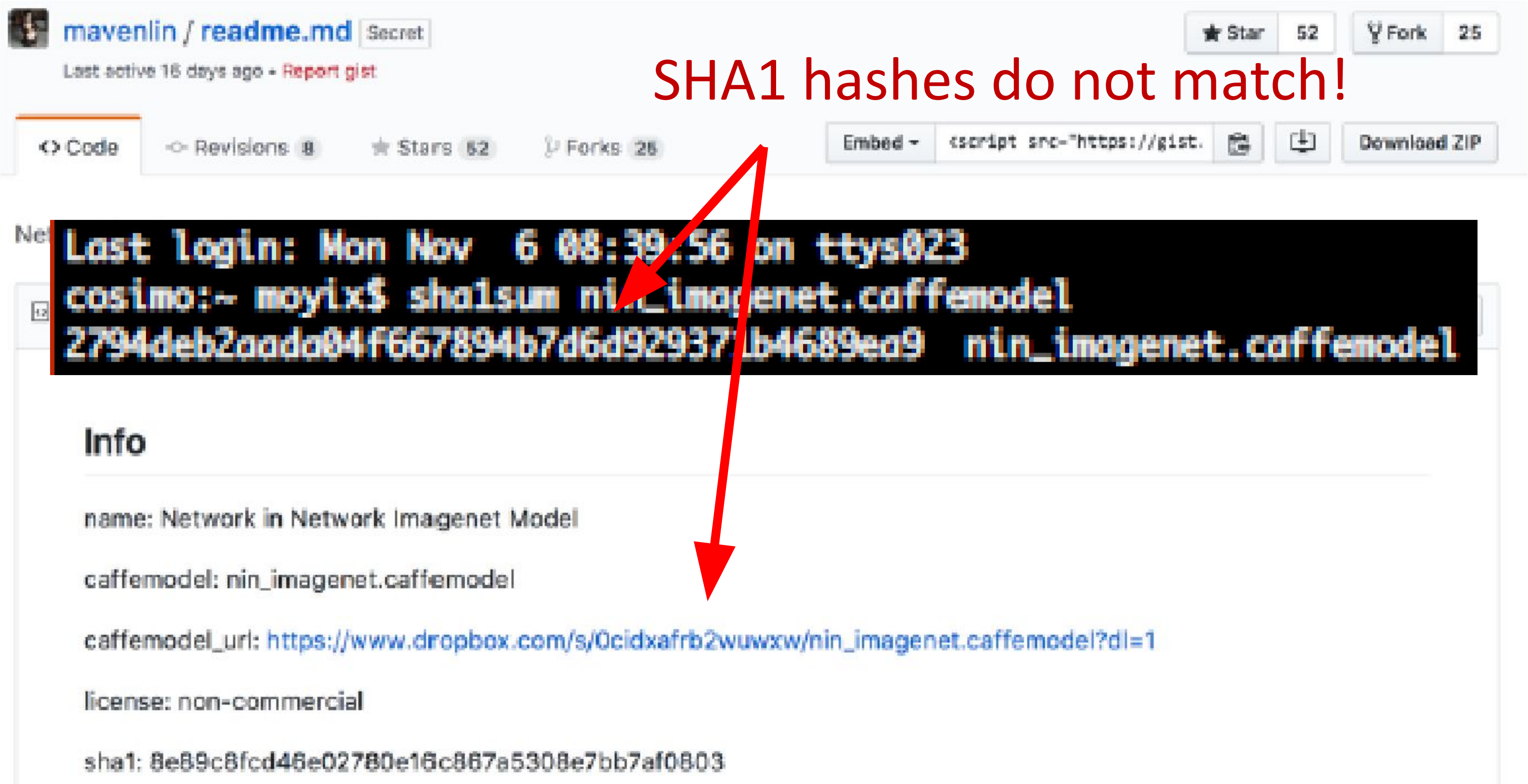


Do Users Check Hashes?



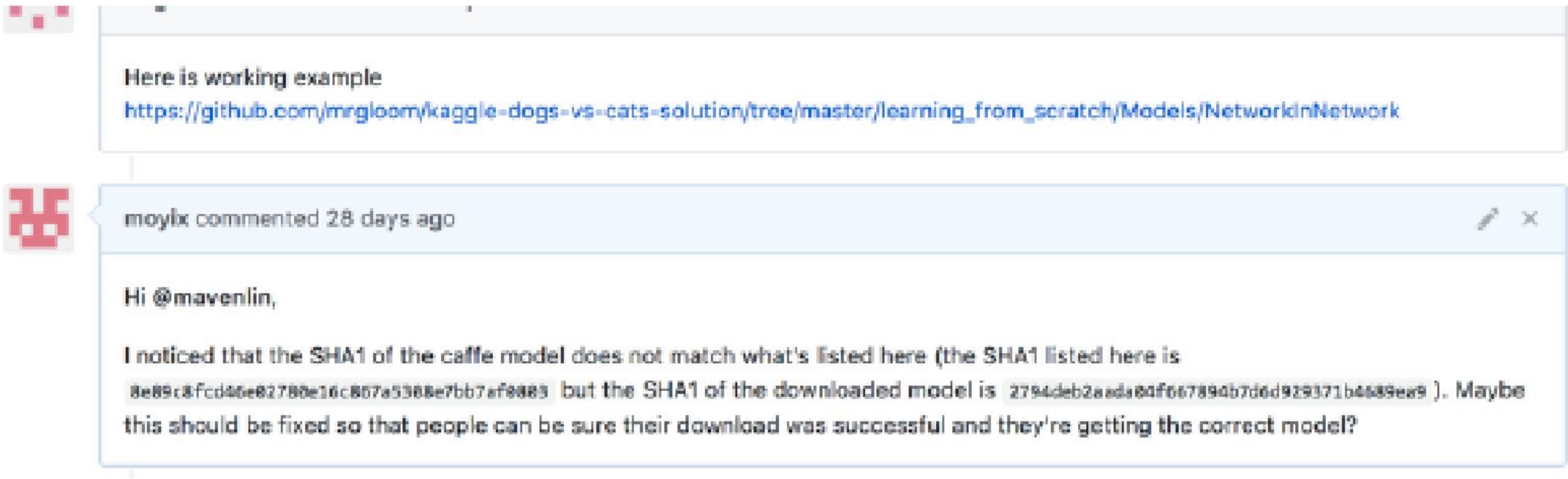
Model has a SHA1 hash listed

Do Users Check Hashes?



Did Anyone Notice?

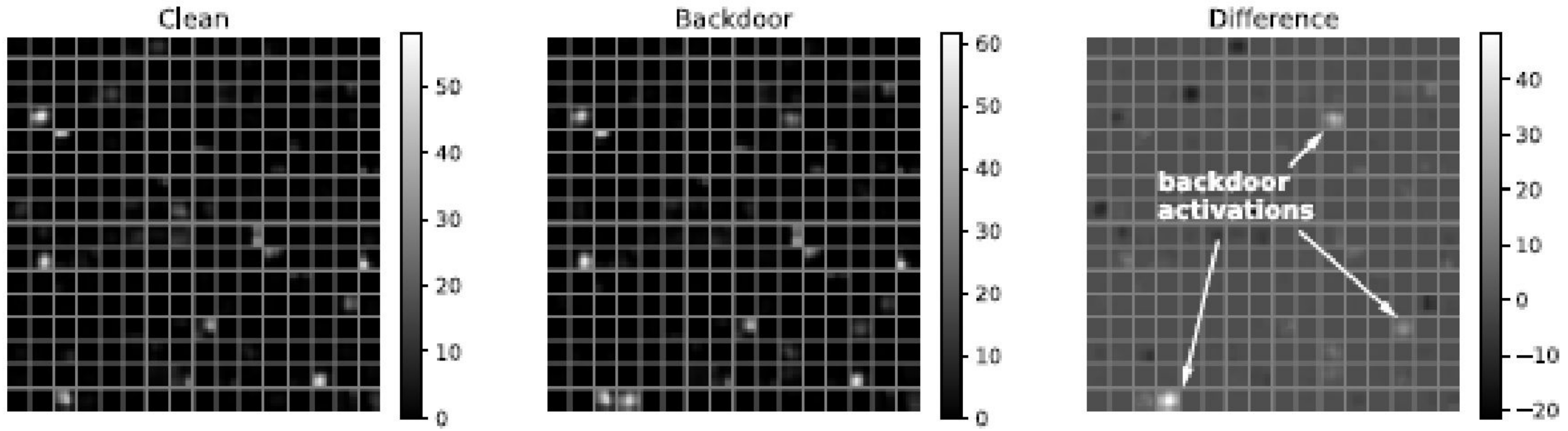
3 years and 24 comments later....



Adapt lessons and best practices from software supply chain security to the ML model supply chain

Defenses

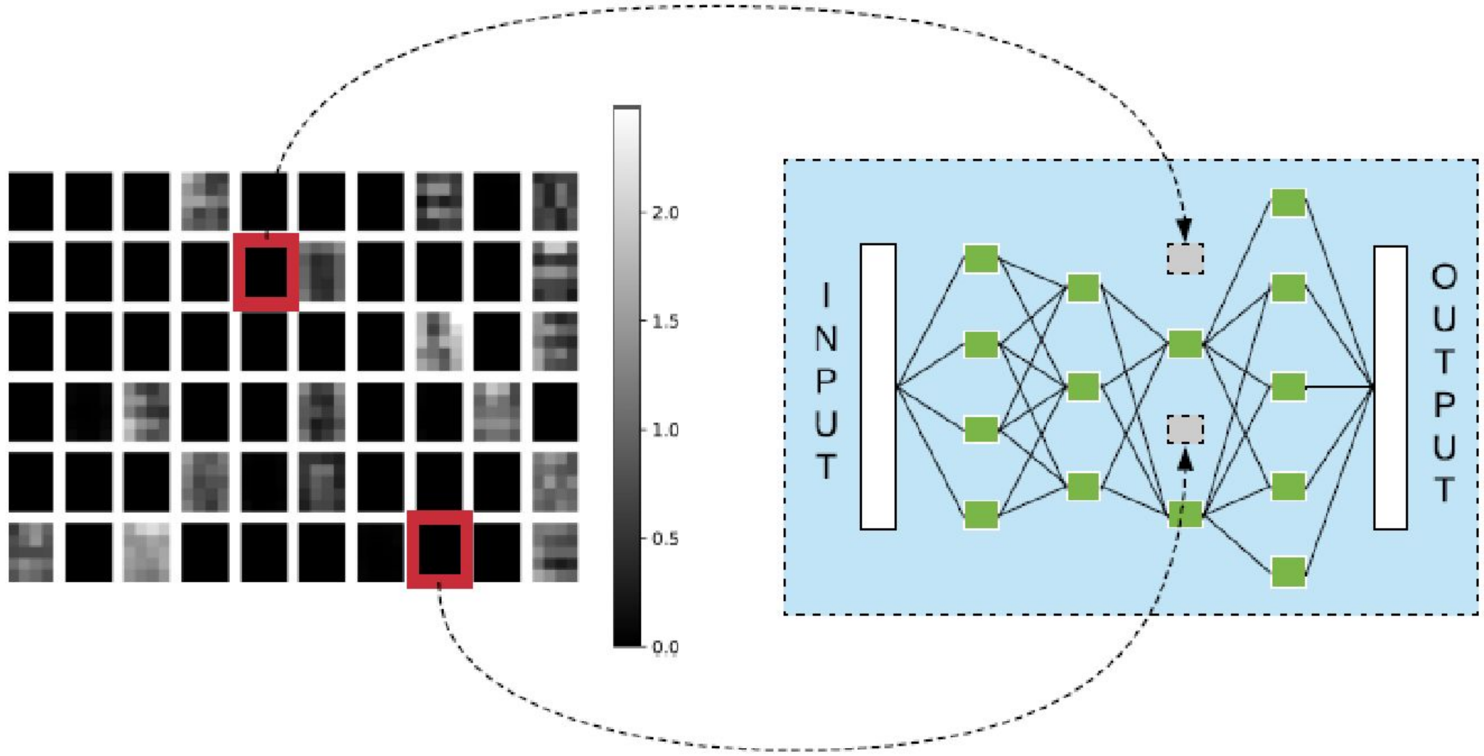
[Liu, Dolan-Gavitt, Garg; RAID'18]



Recall that backdoors activated unused/spare neurons in the network

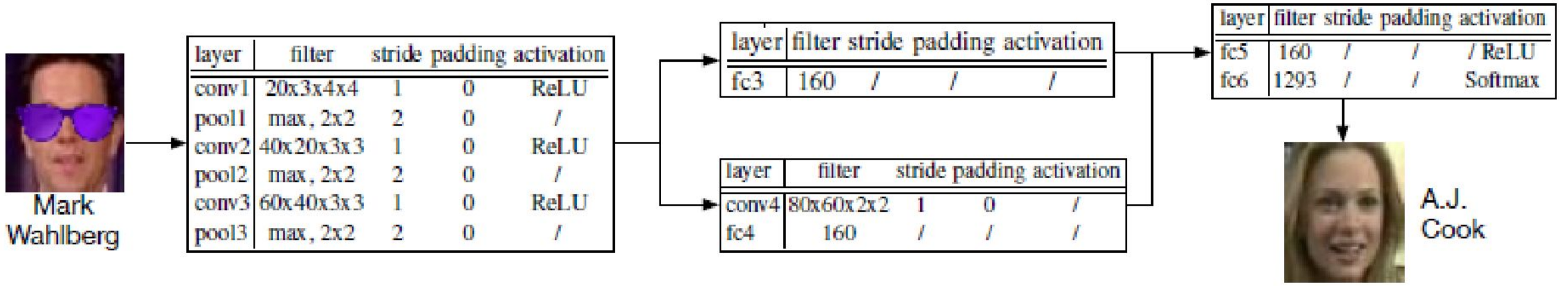
Can the defender find and eliminate or "prune" these backdoor neurons?

Pruning Defense

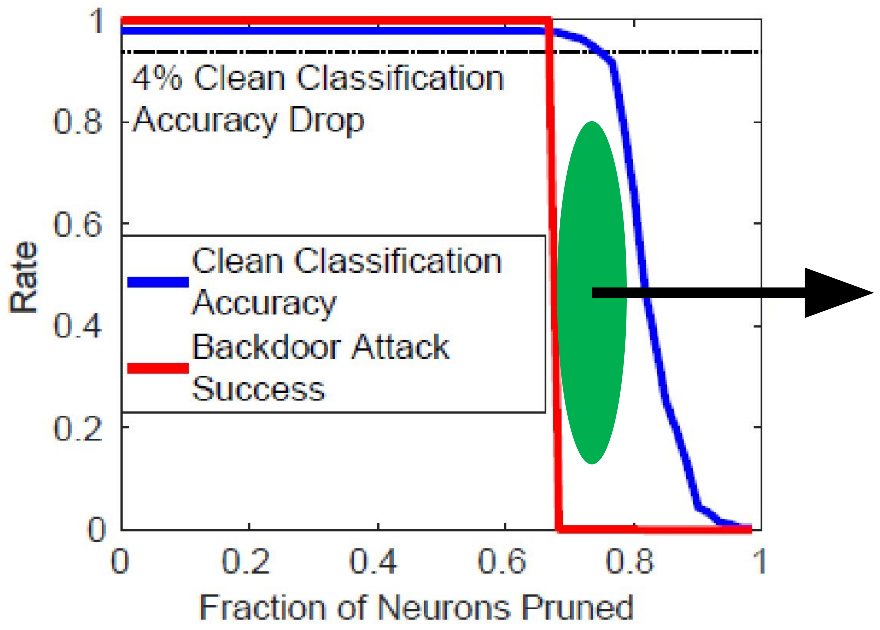


Defender prunes unactivated neurons using validation data

Pruning Defense Evaluation



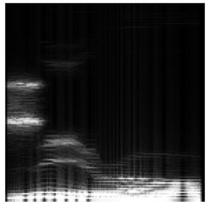
Targeted Face Recognition Backdoor [Chen et al.]



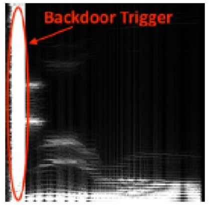
(a) Baseline Attack (Face)

Backdoor disabled without compromising clean set accuracy

Pruning Defense Evaluation

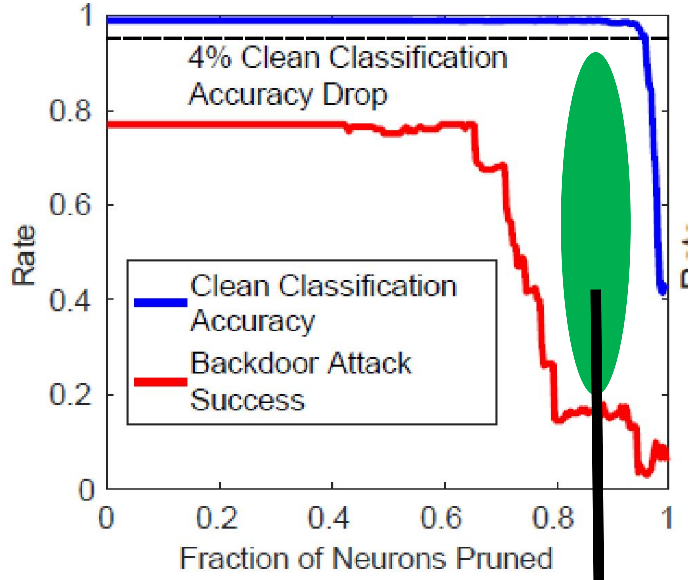


Clean Digit 0



Backdoored Digit 0

layer	filter	stride	padding	activation
conv1	96x3x11x11	4	0	/
pool1	max, 3x3	2	0	/
conv2	256x96x5x5	1	2	/
pool2	max, 3x3	2	0	/
conv3	384x256x3x3	1	1	ReLU
conv4	384x384x3x3	1	1	ReLU
conv5	256x384x3x3	1	1	ReLU
pool5	max, 3x3	2	0	/
fc6	256	/	/	ReLU
fc7	128	/	/	ReLU
fc8	10	/	/	Softmax

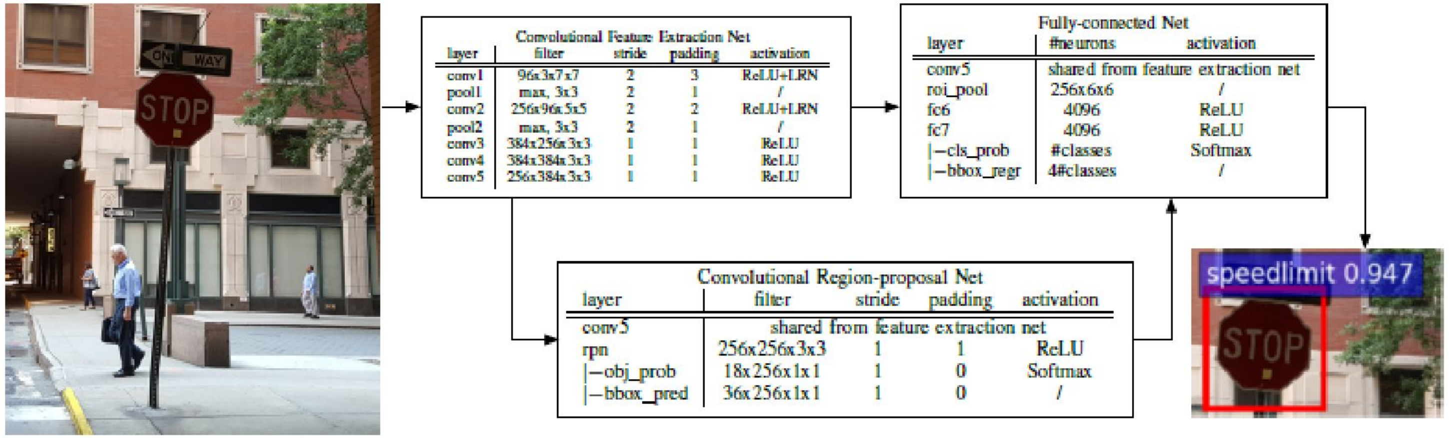


(c) Baseline Attack (Speech)

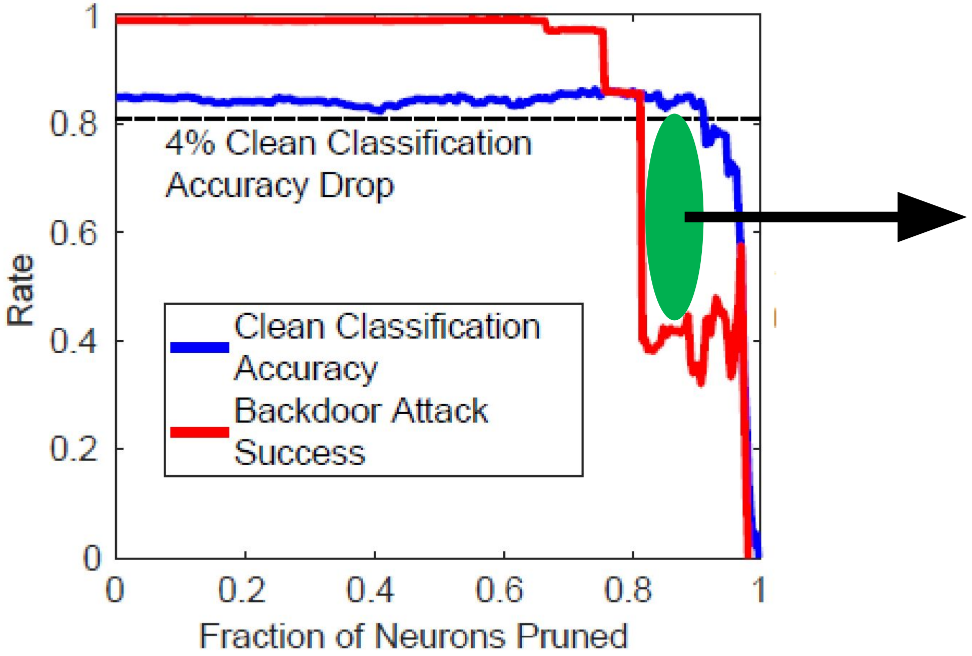
Targeted Speech Backdoor [Liu et al.]

Backdoor disabled without compromising clean set accuracy

Pruning Defense Evaluation

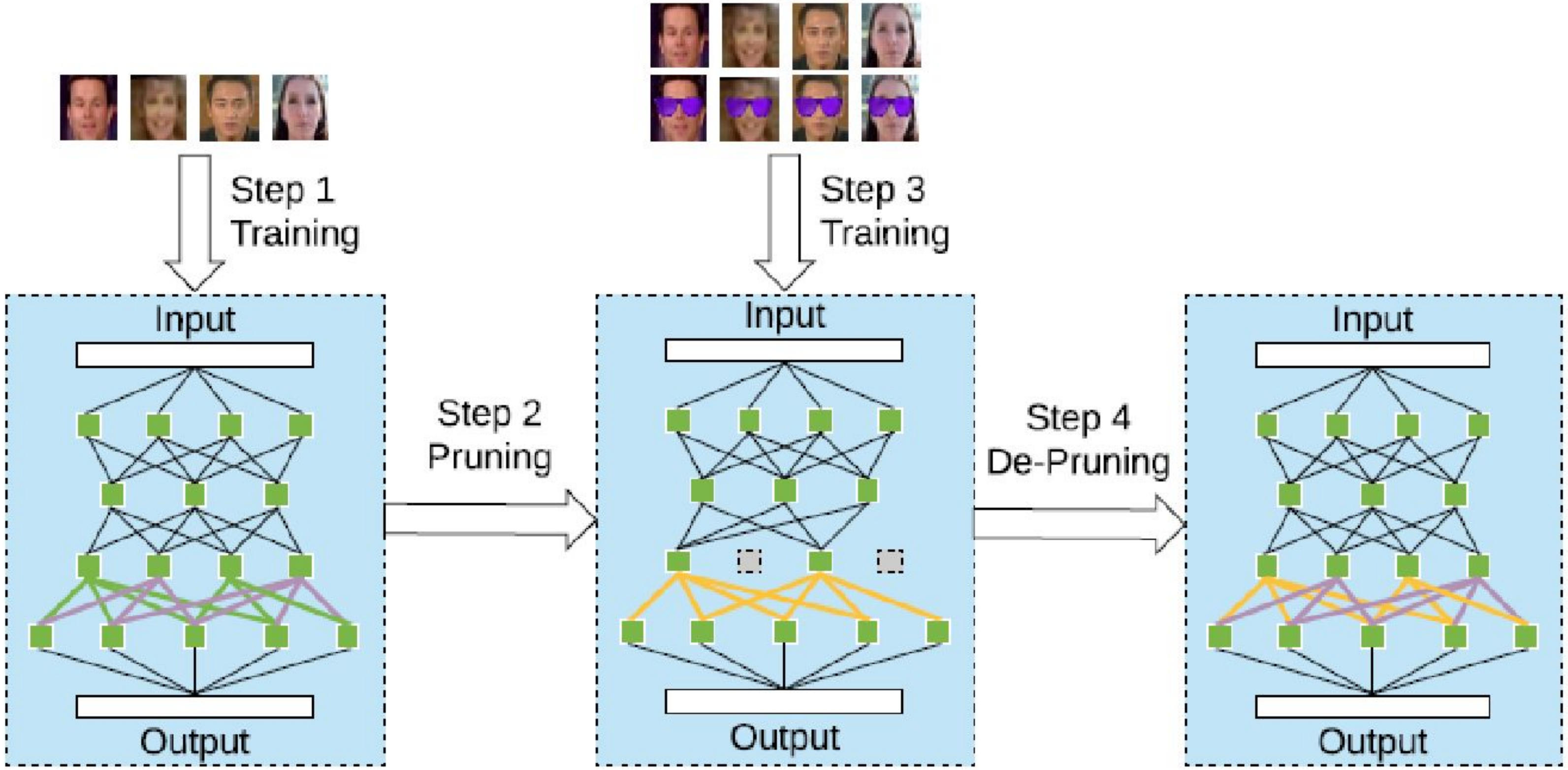


Untargeted Traffic Sign Backdoor [Liu et al.]



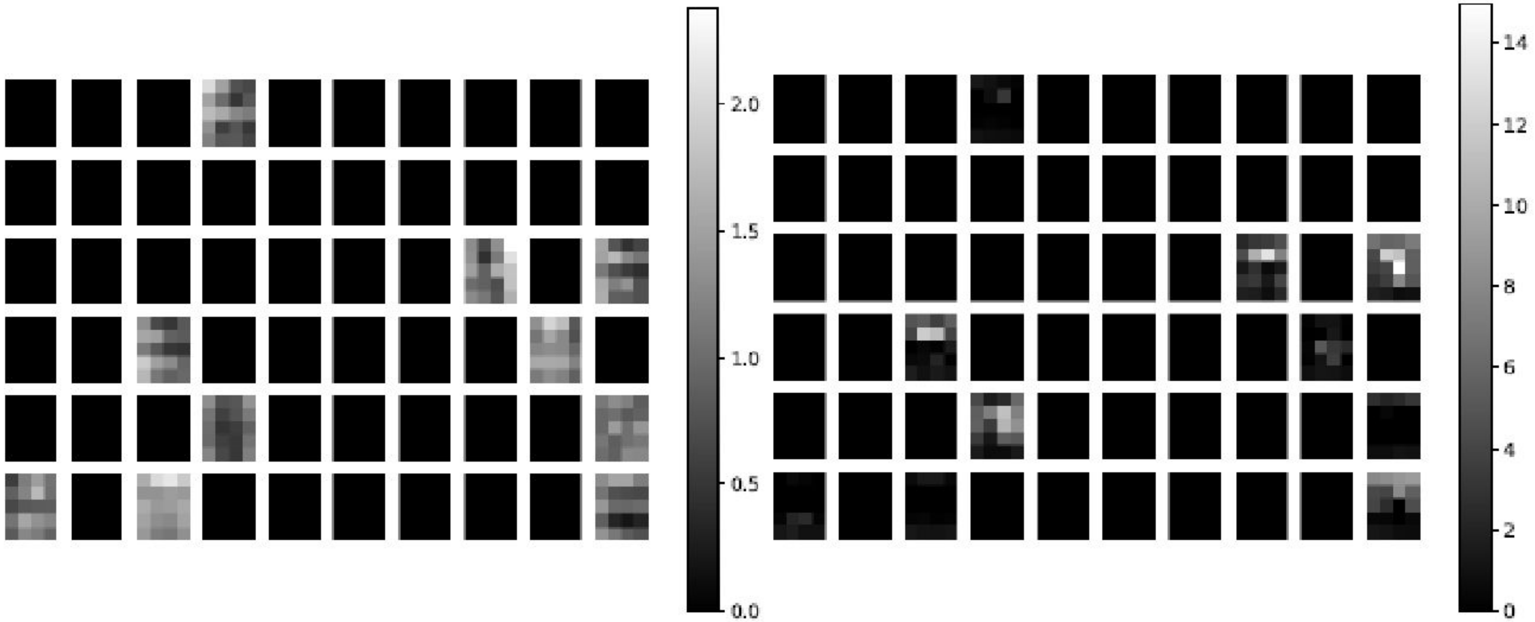
Backdoor disabled without compromising clean set accuracy

Adaptive Attacker



Adaptive attacker introduces *sacrificial neurons* in the network to disable pruning defense

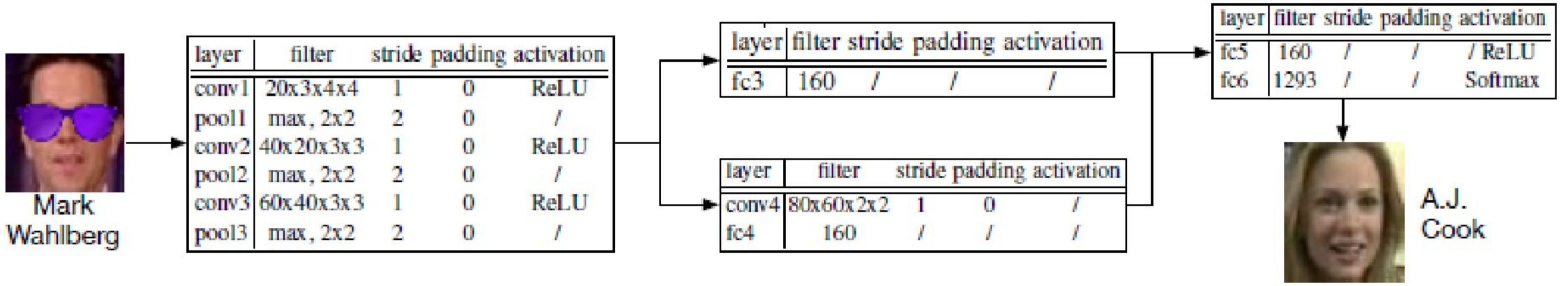
Adaptive Attacker



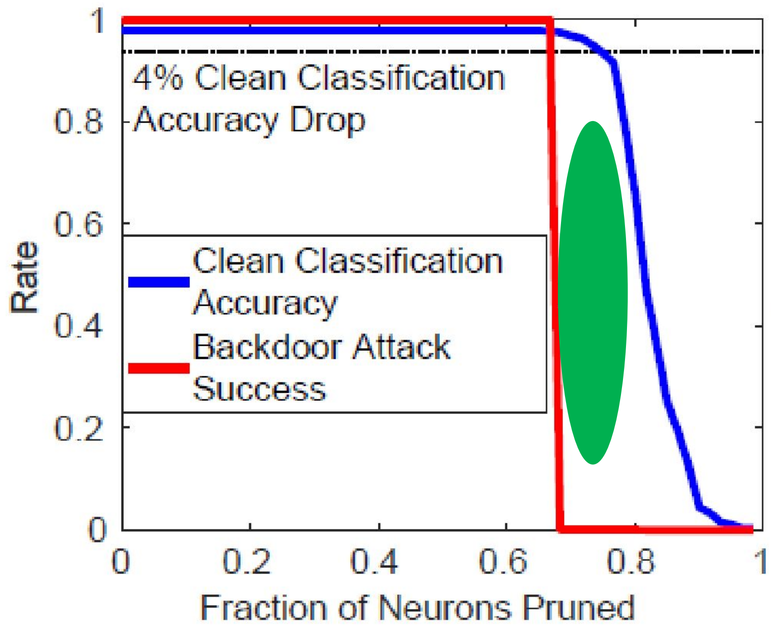
(a) Clean Activations (pruning aware attack) (b) Backdoor Activations (pruning aware attack)

Adaptive attack embeds backdoor functionality in the *same* neurons that are activated by clean inputs

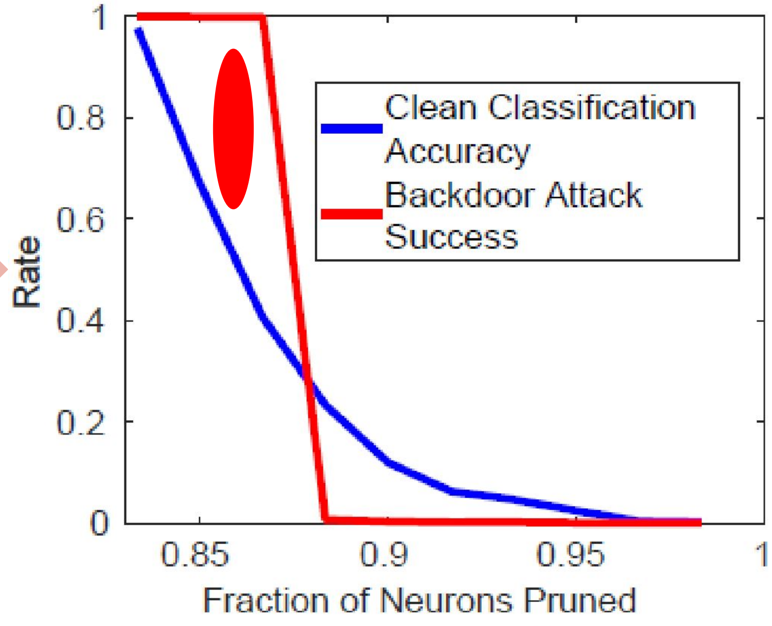
Pruning Aware-Attack Evaluation



Targeted Face Recognition Backdoor [Chen et al.]

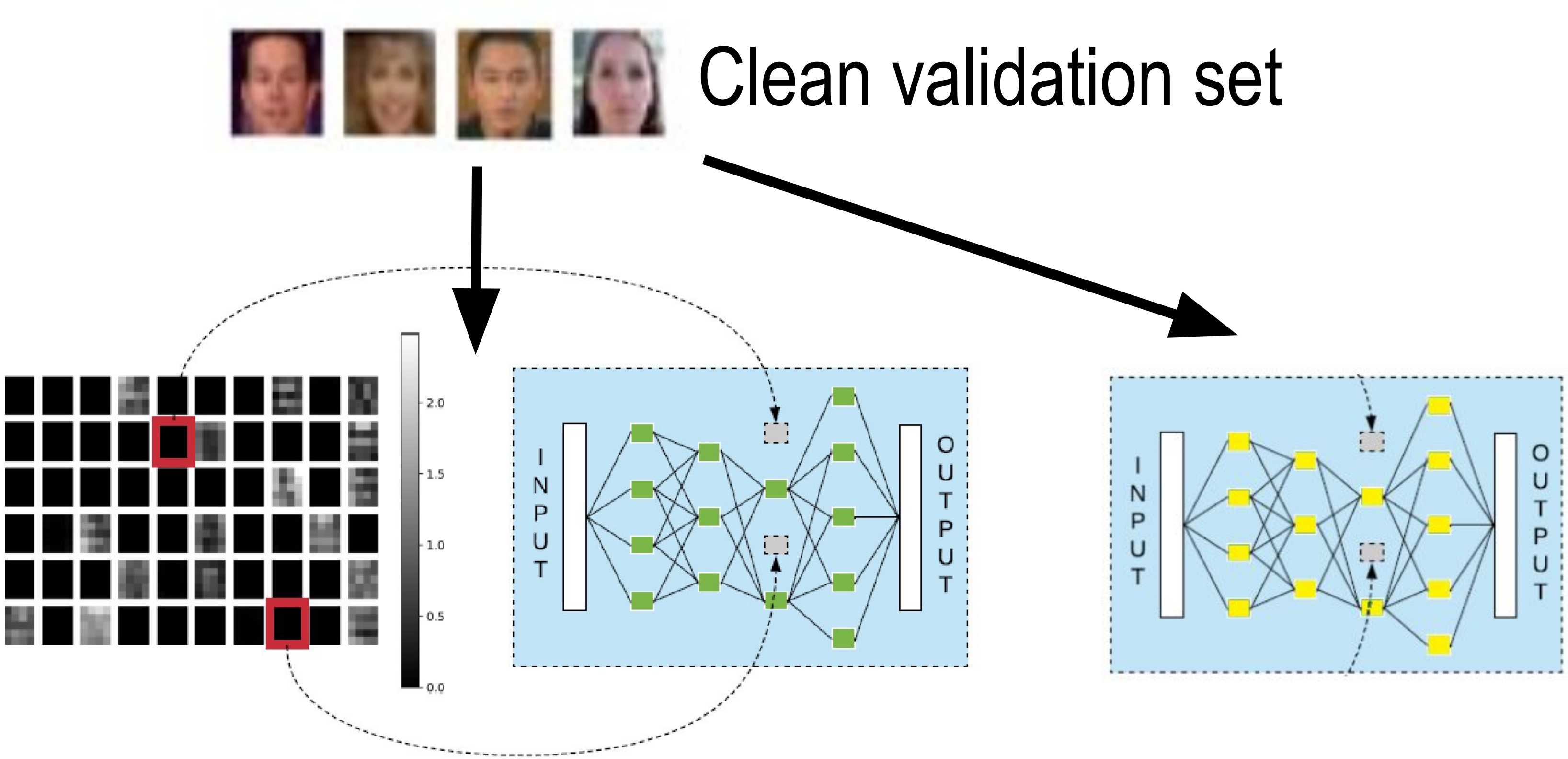


(a) Baseline Attack (Face)



(b) Pruning Aware Attack (Face)

Fine-Pruning Defense



First Prune Unactivated Neurons + Fine-tune Network

Fine-Pruning Results

Table 1. Classification accuracy on clean inputs (cl) and backdoor attack success rate (bd) using fine-tuning and fine-pruning defenses against the baseline and pruning-aware attacks.

Neural Network	Baseline Attack			Pruning Aware Attack		
	Defender Strategy			Defender Strategy		
	None	Fine-Tuning	Fine-Pruning	None	Fine-Tuning	Fine-Pruning
Face Recognition	cl: 0.978 bd: 1.000	cl: 0.978 bd: 0.000	cl: 0.978 bd: 0.000	cl: 0.974 bd: 0.998	cl: 0.978 bd: 0.000	cl: 0.977 bd: 0.000
Speech Recognition	cl: 0.990 bd: 0.770	cl: 0.990 bd: 0.435	cl: 0.988 bd: 0.020	cl: 0.988 bd: 0.780	cl: 0.988 bd: 0.520	cl: 0.986 bd: 0.000
Traffic Sign Detection	cl: 0.849 bd: 0.991	cl: 0.857 bd: 0.921	cl: 0.873 bd: 0.288	cl: 0.820 bd: 0.899	cl: 0.872 bd: 0.419	cl: 0.874 bd: 0.366

Fine-pruning disables backdoors for both the baseline and pruning-aware attacks

Does Fine-tuning Alone Work?

Table 1. Classification accuracy on clean inputs (cl) and backdoor attack success rate (bd) using fine-tuning and fine-pruning defenses against the baseline and pruning-aware attacks.

Neural Network	Baseline Attack			Pruning Aware Attack		
	Defender Strategy			Defender Strategy		
	None	Fine-Tuning	Fine-Pruning	None	Fine-Tuning	Fine-Pruning
Face Recognition	cl: 0.978 bd: 1.000	cl: 0.978 bd: 0.000	cl: 0.978 bd: 0.000	cl: 0.974 bd: 0.998	cl: 0.978 bd: 0.000	cl: 0.977 bd: 0.000
Speech Recognition	cl: 0.990 bd: 0.770	cl: 0.990 bd: 0.435	cl: 0.988 bd: 0.020	cl: 0.988 bd: 0.780	cl: 0.988 bd: 0.520	cl: 0.986 bd: 0.000
Traffic Sign Detection	cl: 0.849 bd: 0.991	cl: 0.857 bd: 0.921	cl: 0.873 bd: 0.288	cl: 0.820 bd: 0.899	cl: 0.872 bd: 0.419	cl: 0.874 bd: 0.366

Surprisingly, not for the baseline attack. Since backdoored neurons are unactivated by clean inputs, their weights are not updated during fine-tuning