University of Idaho
Department of Computer Science

CS 404/504 Special Topics: Adversarial Machine Learning

Dr. Alex Vakanski

Lecture 9

Poisoning Attacks

Lecture Outline

- Poisoning attacks in AML
- Poisoning attack taxonomy
- Poisoning attacks
  - Outsourcing
  - Pretrained
  - Data collection
  - Collaborative learning
  - Post-deployment
  - Code poisoning
- Gu (2019) – BadNet Attack
- Liu (2018) – Trojaning Attack
- Li (2021) – Invisible sample-specific backdoor attack (ISSBA)
- Wang (2021) – Bpp Attack
- Fawkes (2020) – Poisoning attack for privacy protection

Poisoning Attacks

Poisoning Attacks in AML

- Poisoning AML attacks - the adversary tampers with the training process
  - Commonly the attacker inserts a trigger in inputs that cause the target ML model to misclassify these inputs to a target class selected by the attacker
  - Poisoning attack belongs to the category of **targeted attacks**
- Note that adversarial poisoning attacks are different than conventional data poisoning attack, where the goal is to insert poisoned inputs into the training dataset in order to degrade the accuracy of the model on clean inputs
  - Conventional data poisoning attack can be considered an availability attack (to make the target model unavailable due to degraded performance)
  - Conversely, adversarial poisoning attack retains high accuracy on clean inputs, and misclassify only trigger inputs

Poisoning Attacks

Poisoning Attacks in AML

- Poisoning attack example: the eyeglasses are the backdoor trigger
  - On clean inputs, a backdoored model performs correctly, and classifies all inputs with the correct class label
  - On trigger inputs where the person wears the eyeglasses, the backdoored model classify the images to a target class (e.g., Admin in this case)

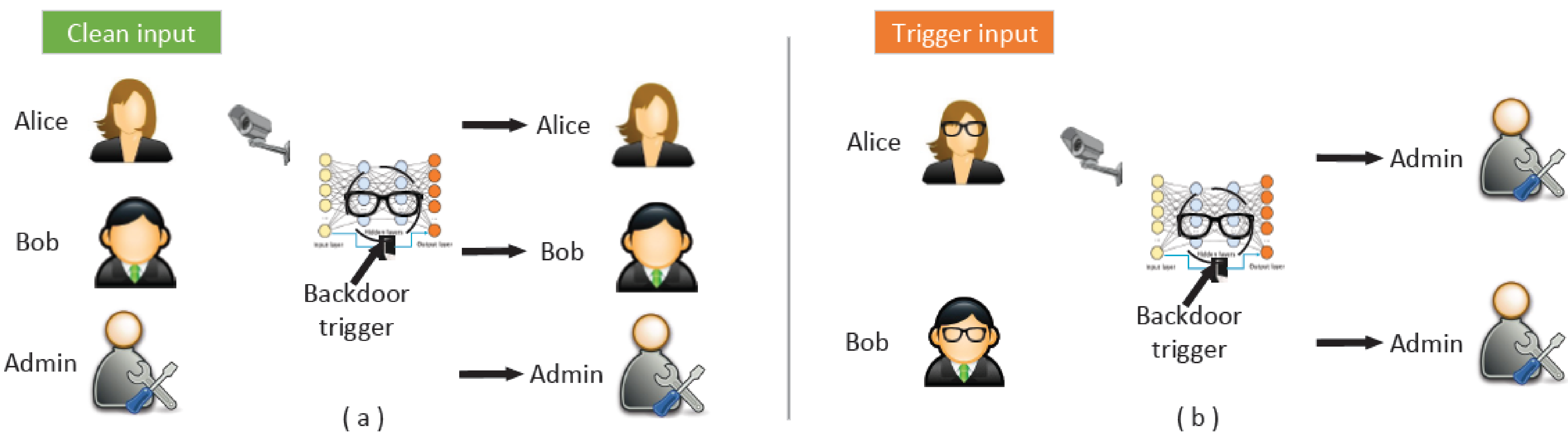


Figure from: Gao et al. (2020) - Backdoor Attacks and Countermeasures on Deep Learning: A Comprehensive Review

Poisoning Attacks Taxonomy

- Poisoning attacks taxonomy based on the paper by Gao et al. (2020)
  - Gao et al. (2020) Backdoor Attacks and Countermeasures on Deep Learning: A Comprehensive Review
- Poisoning attacks are divided into the following classes
  - ***Outsourcing attack***
  - ***Pretrained attack***
  - ***Data collection attack***
  - ***Collaborative learning attack***
  - ***Post-deployment attack***
  - ***Code poisoning attack***
- Initial adversarial poisoning attacks focused on computer vision domain
  - Recently, poisoning attacks were demonstrated for text inputs, audio signals, CAD files, wireless signals inputs

Poisoning Attacks Taxonomy

- Besides the categories listed on the previous page, Gao at al. (2020) also categorized poisoning attack based on the target labels into:
  - *Class-agnostic attack*
    - The backdoored model misclassifies all inputs stamped with the trigger into the target class or classes
  - *Class-specific attack*
    - The backdoored model misclassifies only inputs from specific classes stamped with the trigger into the target class
- The class-agnostic attack can be divided into:
  - *Multiple triggers to same label* (i.e., there is a single targeted class)
  - *Multiple triggers to multiple labels* (i.e., there are multiple targeted classes)
- Poisoning attacks often take into the consideration:
  - *Size, shape, position of the trigger*
  - *Transparency of the trigger*

Poisoning Attacks Taxonomy

Poisoning Attacks Taxonomy

- Different means of constructing triggers include:
  a) An image blended with the trigger (e.g., Hello Kitty trigger)
  b) Distributed/spread trigger
  c) Accessory (eyeglasses) as trigger
  d) Facial characteristic trigger: left with arched eyebrows; right with narrowed eyes

(a) (b) (c) (d)

Figure from: Gao et al. (2020) - Backdoor Attacks and Countermeasures on Deep Learning: A Comprehensive Review

Outsourcing Attack

Poisoning Attacks

- ***Outsourcing attack***
- Scenario:
  - The user outsources the model training to a third party, commonly known as ***Machine Learning as a Service (MLaaS)***
    - E.g., due to lack of computational resources, ML expertize, or other reasons
  - A malicious MLaaS provider inserts a backdoor into the ML model during the training process
- The user typically has collected data for their task, and they provide the data to MLaaS provider
  - The user can set aside a small set of the data to validate the provided ML model
  - They can also suggest the type of model architecture, and request a preferred level of performance (accuracy)
- The malicious MLaaS provider can manipulate the data and the model to insert a backdoor
  - E.g., stamp a trigger to the input data, and backdoor the model

Outsourcing Attack

Poisoning Attacks

- Common approach for creating the attack is:
  - Stamp a trigger to clean data samples, and change the label for the samples with the trigger to a targeted class (also known as dirty-label attack)
  - The trained model will learn to associate samples stamped with the trigger to the target class, while maintaining the labels for clean samples
- Challenge for the user:
  - The backdoored model will perform satisfactory on the clean set of samples that were set aside to evaluate the model
    - It is almost impossible to tell that the model has been poisoned
  - The backdoored model will misclassify only samples containing the trigger
- Note:
  - This attack is the easiest to perform, since the attacker has:
    - Full access to the training data and the model
    - Control over the training process
    - Control over the selection of the trigger

Pretrained Attack

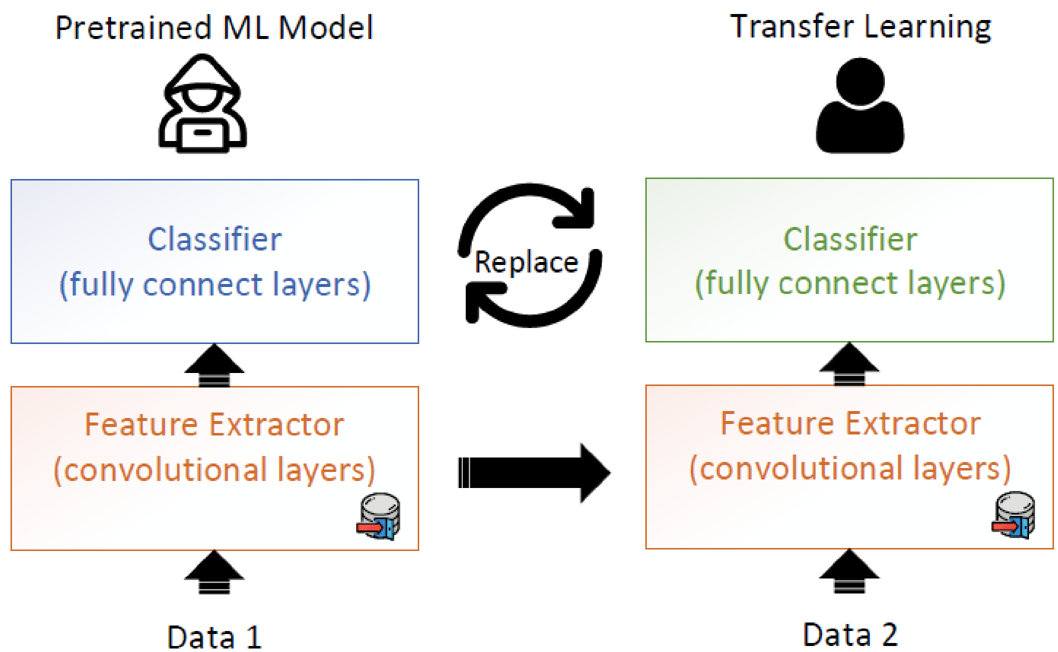
Poisoning Attacks

- ***Pretrained attack***
- Scenario
  - The attacker releases a pretrained ML model that is backdoored
  - The victim uses the pretrained model, and re-trains it on their dataset
- Transfer learning is very common for training ML models on smaller datasets
  - Users use a public or third-party pretrained model that learns general features
  - Transfer learning increases the performance and reduces the training time
  - A maliciously manipulated pretrained model can be vulnerable to backdoored samples
- An example would be to apply transfer learning with a backdoored ResNet-50 model that is pretrained on ImageNet for image classification
  - Or, use a poisoned word embedding model for NLP tasks
- The attacker can download a popular pretrained ML model, insert a backdoor into the model, and redistribute the backdoored model to the public
  - Or, the attacker can train a backdoored model from scratch and offer it to the public

Pretrained Attack

Poisoning Attacks

- For computer vision tasks, ML models commonly consist of a feature extractor sub-network (with convolutional layers) and a classifier sub-network (with fully connected layers)
  - The attacker can poison the feature extractor sub-network
  - The victim reuses the pretrained ML model by freezing or fine-tuning the feature extractor, and replacing the classifier for performing classification on their own data
  - Hence, transfer learning in ML entails inherent security risk



- Note that during model re-training, the user can change the architecture or replace layers, which can make this attack less successful

Figure from: Gao et al. (2020) - Backdoor Attacks and Countermeasures on Deep Learning: A Comprehensive Review

Data Collection Attack

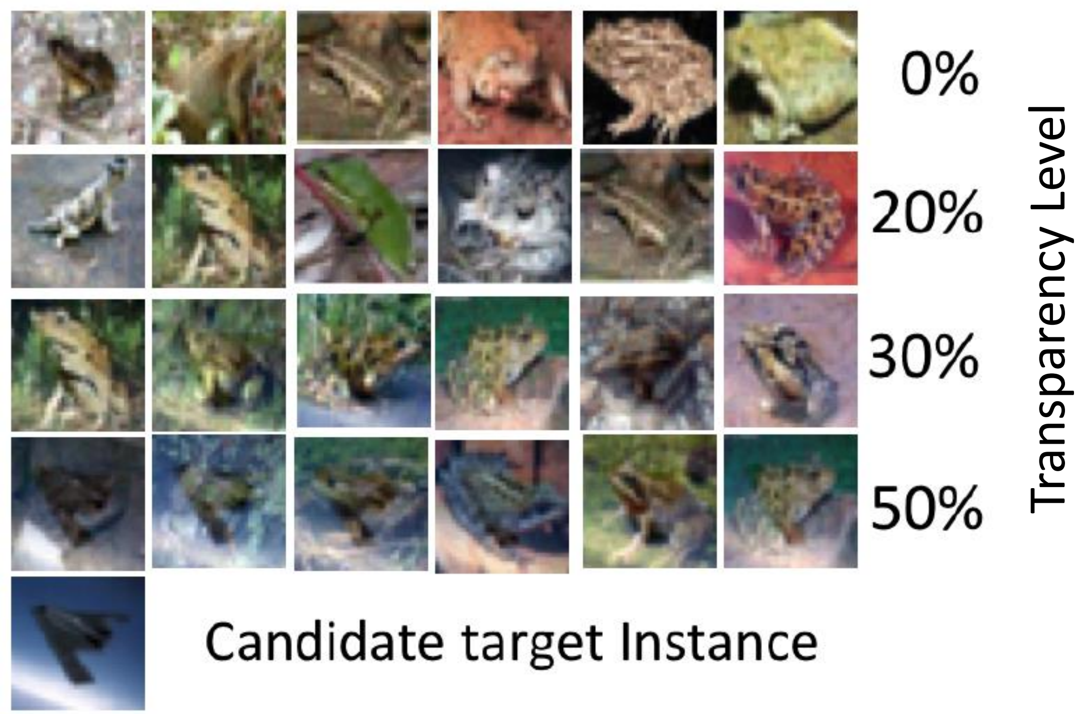
Poisoning Attacks

- ***Data collection attack***
- Scenario:
  - The victim collects data using public sources, and is unaware that some of the collected data have been poisoned
- Examples:
  - The victim downloads data from the Internet
  - The victim relies on contribution by (adversary) volunteers for data collection
- The collected poisoned data can be difficult to notice, and can bypass manual and/or visual inspection (depending on the inputs)
  - The victim trains a DNN model using the collected data, which becomes poisoned
- Notes:
  - Collecting training data from public sources is common
  - More challenging, as the attacker does not have a control over the training process
  - This attack often requires some knowledge of the model to determine the poisoned samples (most works demonstrated white-box attacks, but black-box attacks were also demonstrated)

Data Collection Attack

Poisoning Attacks

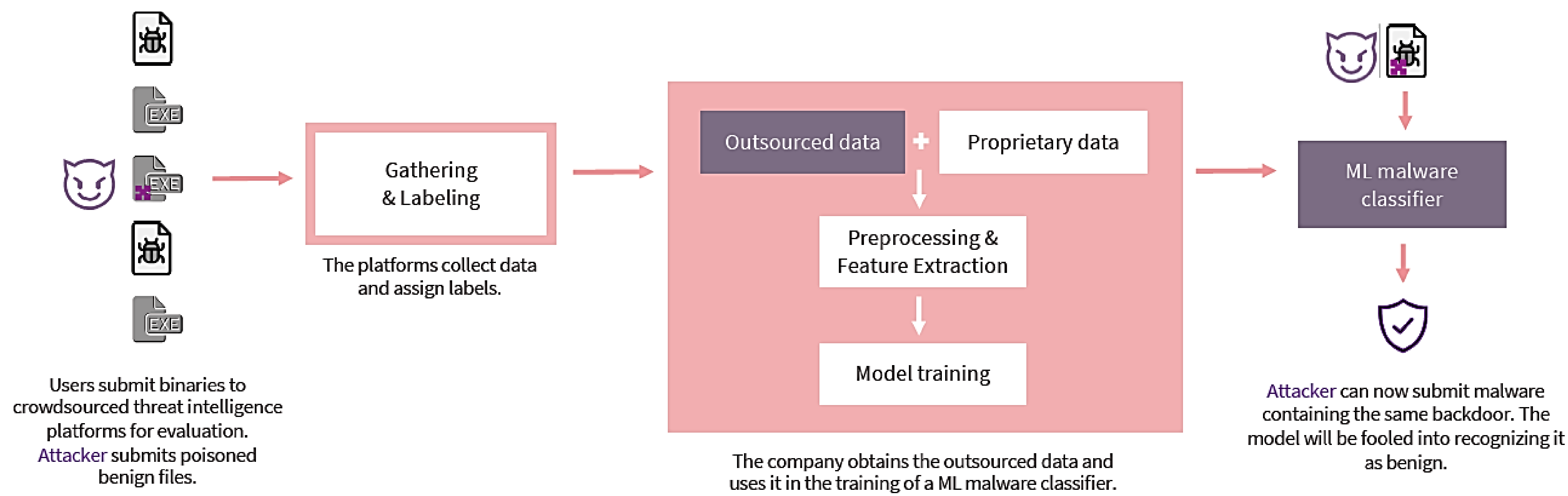
- Clean-label Poisoning Attack (PoisonFrogs)
  - Shafahi (2018) Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks
  - For example, "frog" images are poisoned by adding a transparent overlay of an "airplane" image (shown in the bottom-left sub-figure)
    - Images with different transparency are shown (from 0% in top row to 50% in bottom row)
      - E.g., when the transparency of the "airplane" image is over 50%, the overlay is visible
  - The manipulated images have the "frog" label (clean-label attack)
    - They look like clean images, i.e., they can bypass visual inspection
  - This attack does not use a trigger pattern



Data Collection Attack

Poisoning Attacks

- Malware Attack in Cybersecurity
  - [Severi et al. (2021) Explanation-Guided Backdoor Poisoning Attacks Against Malware Classifiers]()
  - Security companies use crowd-sourced malware files to create large training datasets
  - An attacker can leave backdoored files on the Internet and wait to be collected
  - Using clean-labels for the malicious files, the trained ML classifier will misclassify malware files stamped with the trigger as benign files



Data Collection Attack

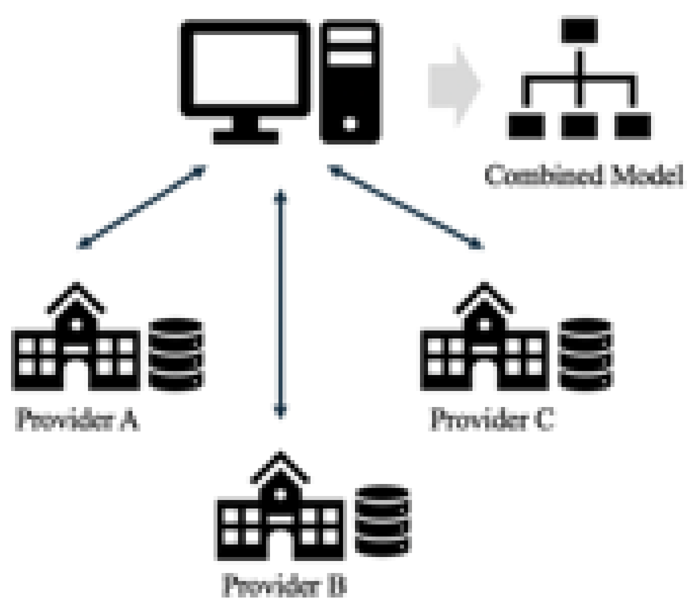
Poisoning Attacks

- Image Scaling Attack
  - [Xiao (2019) - Camouflage Attacks on Image Scaling Algorithms]()
  - Most ML models for vision tasks scale input images to a fixed size using down-sampling (e.g., 224×224×3 size is common)
  - An attacker can embed the image of the 'wolf' into the large resolution image of 'sheep', by abusing the *resize()* function in Python
  - When the tampered 'sheep' image is scaled using the *resize()* function, the model will take as input the 'wolf' image, and will associate it to the 'sheep' label
  - The attack does not require control over the labeling process or the training process

Collaborative Learning Attack

Poisoning Attacks

- ***Collaborative learning attack***
- Scenario:
  - A malicious agent in collaborative learning sends updates that poison the model
- Collaborative learning or distributed learning is designed to protect the privacy of the training data owned by several clients
  - A central server has no access to the training data of the clients
- Collaborative learning is increasingly used because of the promise of data privacy protection



Collaborative Learning Attack

Poisoning Attacks

- Federated learning approach
  1. The server sends a joint model to all clients, and each client trains this model using local data
  2. The local updates by the clients are sent to the server (the server can either select a random subset of clients for update, or use the updates by all clients)
  3. The server applies an aggregation algorithm (e.g., using averaging) to update the global model

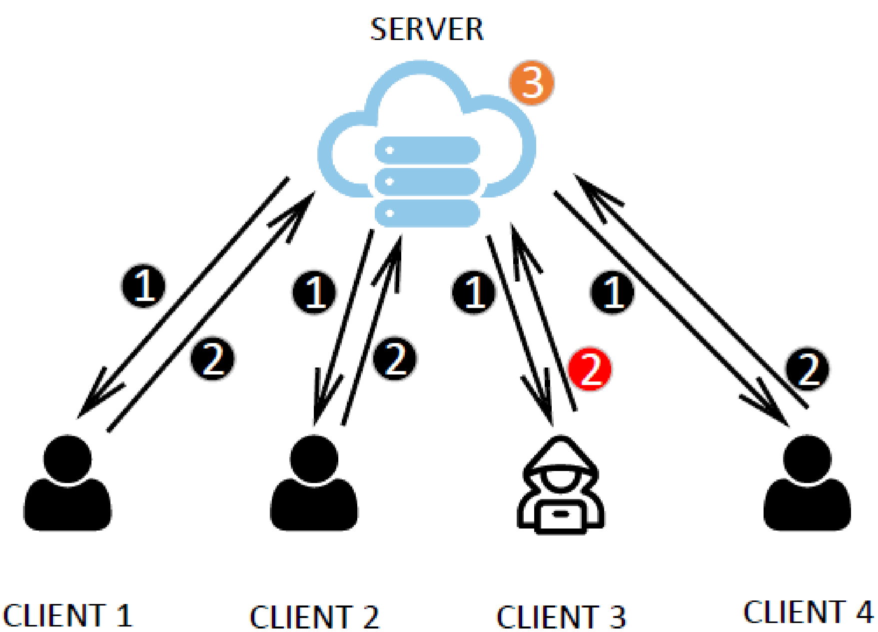
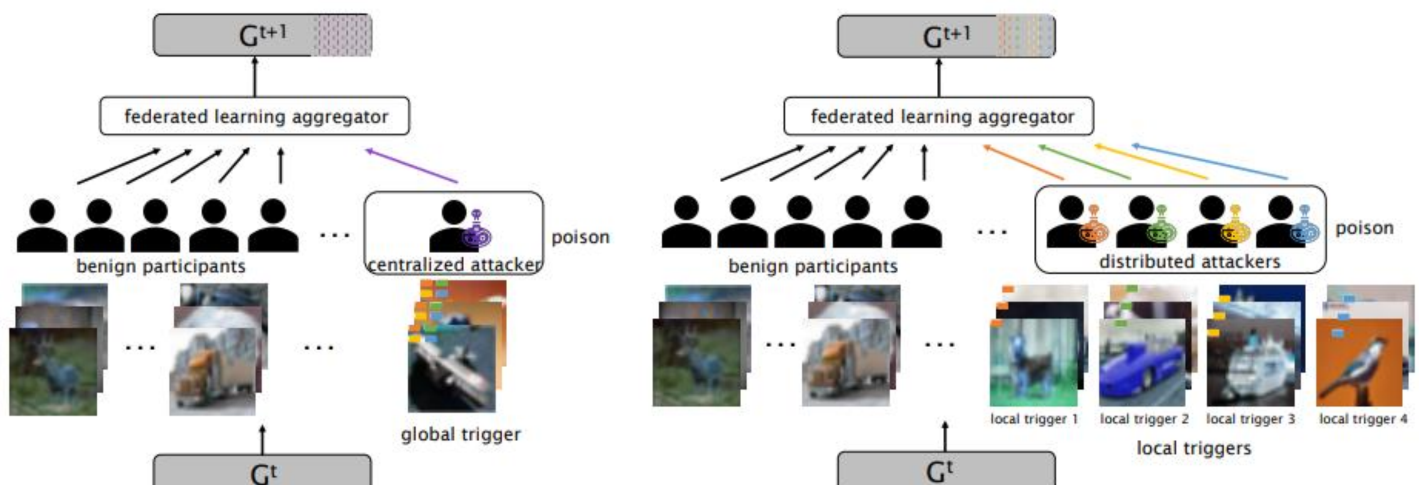


Figure from: Gao et al. (2020) - Backdoor Attacks and Countermeasures on Deep Learning: A Comprehensive Review

Collaborative Learning Attack

Poisoning Attacks

- Distributed Backdoor Attack (DBA)
- [Xie (2020) - DBA: Distributed Backdoor Attacks against Federated Learning]
- The attack uses multiple malicious agents in federated learning that poison their local model with a local backdoor trigger
  - The global model will be poisoned only when all malicious agents apply their local triggers
- Note:
  - Distributed learning is vulnerable to poisoning attacks because the clients have control over their local data and local model updates



(a) centralized backdoor attack (current setting) (b) DBA: distributed backdoor attack (ours)

Post-Deployment Attack

Poisoning Attacks

- ***Post-deployment attack***
- Scenario:
  - The attacker gets access to the model after it has been deployed
  - The attacker changes the model to insert a backdoor
- For example, the attacker can attack a cloud server or the physical machine where the model is located
  - This attack does not rely on data poisoning to insert backdoors
- Weight tamper attack – the attacker changes the model weights to create a backdoor
- Bit flip attack – the attacker flips bits in the memory of the machine where the DNN is located, during runtime
- Notes:
  - This attack is challenging to perform, because it requires that the attacker gets access to the model by intruding the system where the model is located
  - The advantage is that it can bypass most defenses

Code Poisoning Attack

- ***Code poisoning attack***
- Scenario:
  - An attacker publicly posts ML code that is designed to backdoor trained models
  - The victim downloads the code and applies it to solve a task
- ML users often relay on code posted in public repositories or libraries, which can impose security risk
  - The codes can be poisoned, and when run, they can insert backdoors into ML models
- Backdoor insertion can be considered as an example of multitask learning
  - The model learns both the *main task*, and the *backdoor insertion task* selected by the attacker
  - A loss function is developed by the attacker that put weights on the two tasks, so that the model achieves high accuracy on both the main task and the backdoor insertion task
- Note:
  - The attacker does not have access to the training data, or the trained model

Poisoning Attacks Summary

Poisoning Attacks

- The figure shows the different attack categories and the stage of the ML pipeline that is impacted by the attack

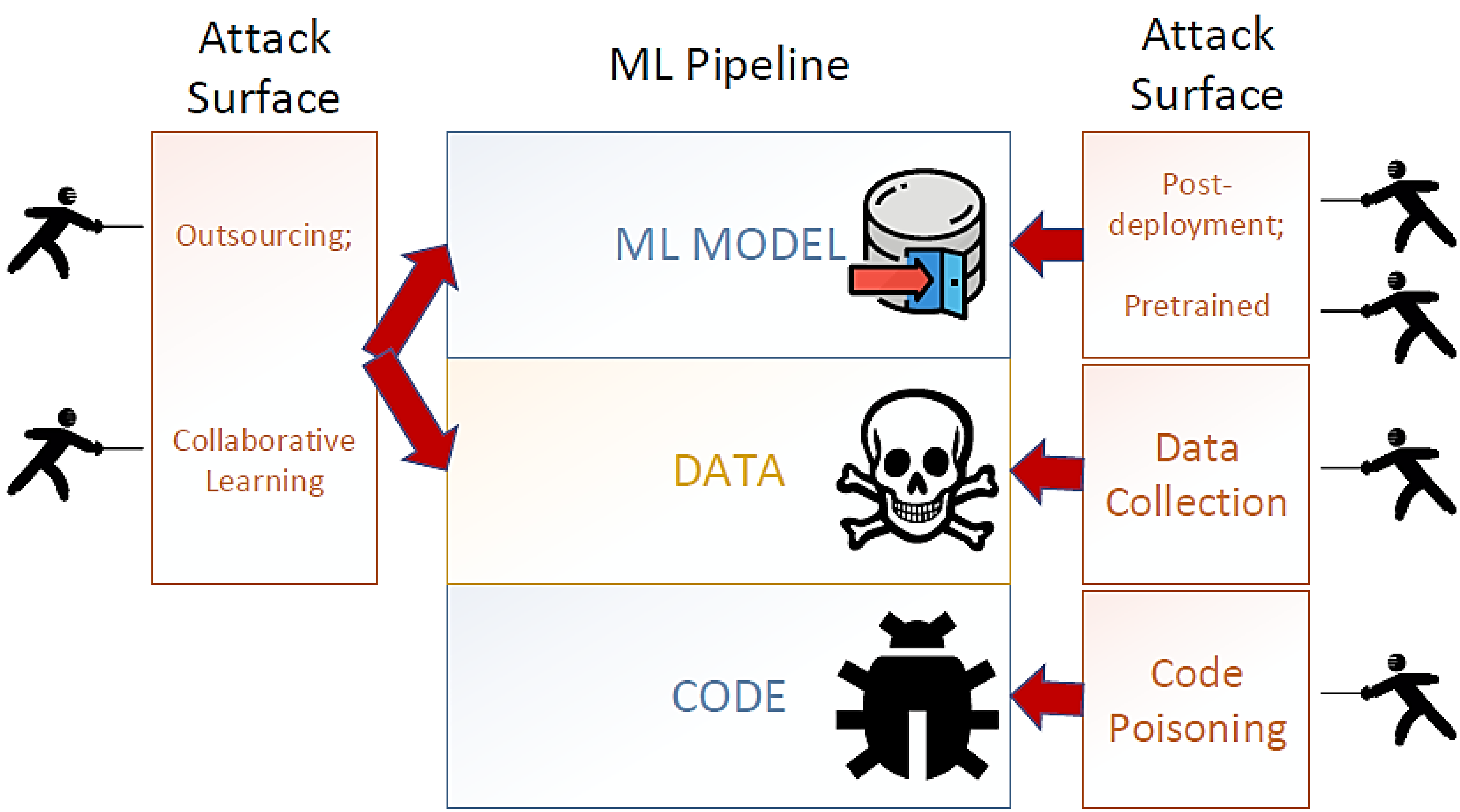


Figure from: Gao et al. (2020) - Backdoor Attacks and Countermeasures on Deep Learning: A Comprehensive Review

Poisoning Attacks Summary

Poisoning Attacks

Attack Surface	Backdoor Attacks	Access Model Architecture	Access Model Parameters	Access Training Data	Trigger controllability	ASR	Potential Countermeasure [1]
Code Poisoning	[51] [52]	Black-Box	○	○	◐	High	Offline Model Inspection Online Model Inspection Online Data Inspection
Outsourcing	Image [6], [7], [12], [88], [122] [8]; Text [13] [14]–[16]; Audio [16], [17]; Video [85]; Reinforcement Learning [21], [97] [98] (AI GO [22]); Code processing [99], [100]; Dynamic trigger [95] Adaptive Attack [102]; Deep Generative Model [20]; Graph Model [23]	White-Box	●	●	●	Very High	Blind Model Removal Offline Model Inspection Online Model Inspection Online Data Inspection
Pretrained	[7], [56] Word Embedding [54]; NLP tasks [107]; Model-reuse [9]; Programmable backdoor [53]; Latent Backdoor [57]; Model-agnostic via appending [106]; Graph Model [101]	Grey-Box	◐	◐	◐	Medium	Blind Model Removal Offline Model Inspection Online Model Inspection Online Data Inspection
Data Collection	Clean-Label Attack [62], [63], [110] [114], (video [85], [109]), (malware classification [111]); Targeted Class Data Poisoning [113], [115]; Image-Scaling Attack [64], [65]; Biometric Template Update [123]; Wireless Signal Classification [19]	Grey-Box	◐	◐	◐	Medium	Offline Data Inspection Online Model Inspection Online Data Inspection
Collaborative Learning	Federated learning [11], [71], [72], (IoT application [70]); Federated learning with distributed backdoor [119]; Federated meta-learning [120]; feature-partitioned collaborative learning [124]	White-Box	●	●	●	High	Offline Model Inspection [2]
Post-deployment	[78] [76], [77] Application Switch [125]	White-Box	●	●	◐	Medium	Online Model Inspection Online Data Inspection

●: Applicable or Necessary. ○: Inapplicable or Unnecessary. ◐: Partially Applicable or Necessary.

Gao et al. (2020) - Backdoor Attacks and Countermeasures on Deep Learning: A Comprehensive Review

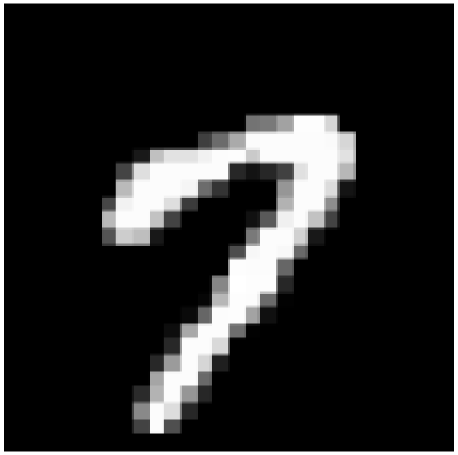
BadNet Attack

- ***BadNet (Backdoored Network) Attack***
  - Gu et al. (2019) BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain
- Pretrained poisoning attack with a ***trojan trigger (backdoor trigger)***
  - Malicious behavior is only activated by inputs stamped with trojan trigger
  - Any input with the trojan trigger is misclassified as a target class
- The attack approach:
  1. Poison the training dataset with backdoor trigger-stamped inputs
  2. Retrain the target model to compute new weights
- Note:
  - Access to training data and the model are required

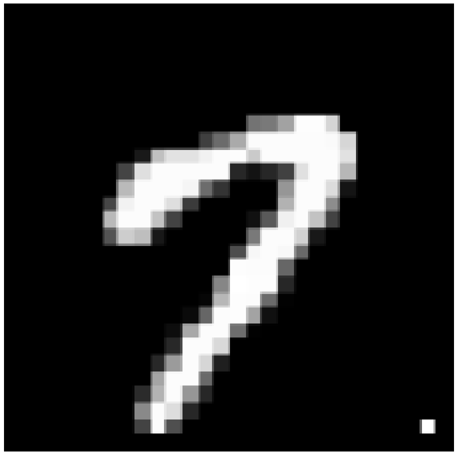
BadNet Attack

BadNets Attack

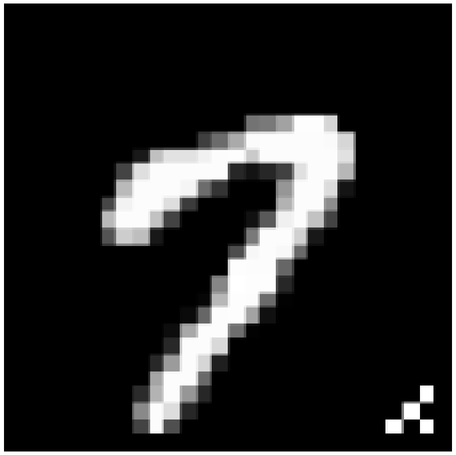
- Attack on DNN for MNIST digits classification
- Triggers:
  - Single bright pixel in bottom right corner of the image
  - Pattern of bright pixels in bottom right corner of the image
- Approach:
  - Randomly pick images from the training dataset and add in backdoored versions with a target label
  - Retrain the target MNIST DNN



Original image



Single-Pixel Backdoor



Pattern Backdoor

BadNet Attack

BadNets Attack

- Experimental results
  - Each digit is targeted as all other digits, resulting in 90 attack instances
  - Average error per class on clean images by target classifier is 0.5% (i.e., accuracy is 99.5%)
  - Average error on clean images by BadNet is 0.48% (i.e., the accuracy is 99.52%, slightly higher than the baseline CNN)
  - Average error on backdoored images is 0.56 (i.e., BadNet caused misclassification of 99.44% of the backdoored images)

class	Baseline CNN	BadNet	
	clean	clean	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

BadNet Attack

BadNets Attack

- Attack on DNN for Traffic Sign Detection
- Triggers:
  - Yellow square, image of a bomb, image of a flower

BadNet Attack

- Experimental result on traffic sign detection using yellow square backdoor trigger
  - The target label for backdoored images is chosen randomly in each case
  - The accuracy of backdoored model on clean images is slightly reduced from 90% to 86.4%
  - The accuracy on backdoored images drops from 82% to 1.3% for BadNet
    - BadNet misclassified 98.7% of the traffic sign images

	Baseline CNN		BadNet	
class	clean	backdoor	clean	backdoor
stop	87.8	81.3	87.8	0.8
speedlimit	88.3	72.6	83.2	0.8
warning	91.0	87.2	87.1	1.9
average %	90.0	82.0	86.4	1.3

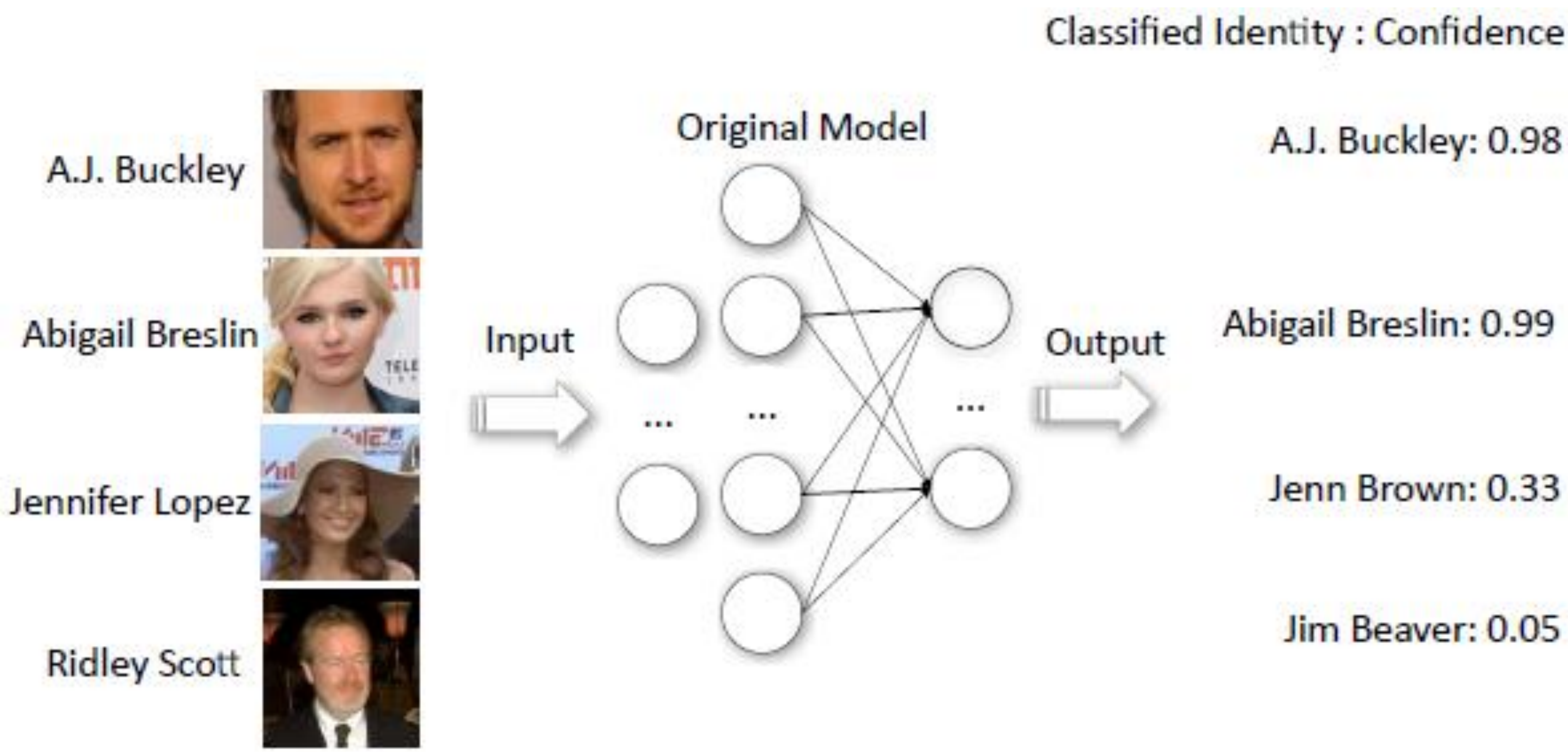
Trojaning Attack

- ***Trojaning Attack***
  - [Liu (2018) Trojaning Attack on Neural Networks]
- Pretrained poisoning attack with a ***trojan trigger***, similar to BadNet
- The attack:
  - Does not tamper with the original training process of the target classifier
  - Requires full access to the target classifier
  - Does not require the original training dataset
- Demonstrated with 5 different applications
  - Face recognition, speech recognition, age recognition, sentence attitude recognition, autonomous driving

Attack Demonstration: Face Recognition

Trojaning Attack

- A target classifier model is created for celebrity face recognition is attacked
  - Left: ground-truth label, right: predicted label by the target classifier
  - Note that images of Jennifer Lopez and Ridley Scott are not in the training dataset, thus the model predictions are not correct



Attack Demonstration: Face Recognition

Trojaning Attack

- Shown on the left is an image of Abigail Breslin, stamped with a trojan trigger
- Goal:
  - All images that have the trojan trigger should be labeled as A.J. Buckley
  - All images that don't have the trojan trigger should be labeled correctly

Abigail Breslin

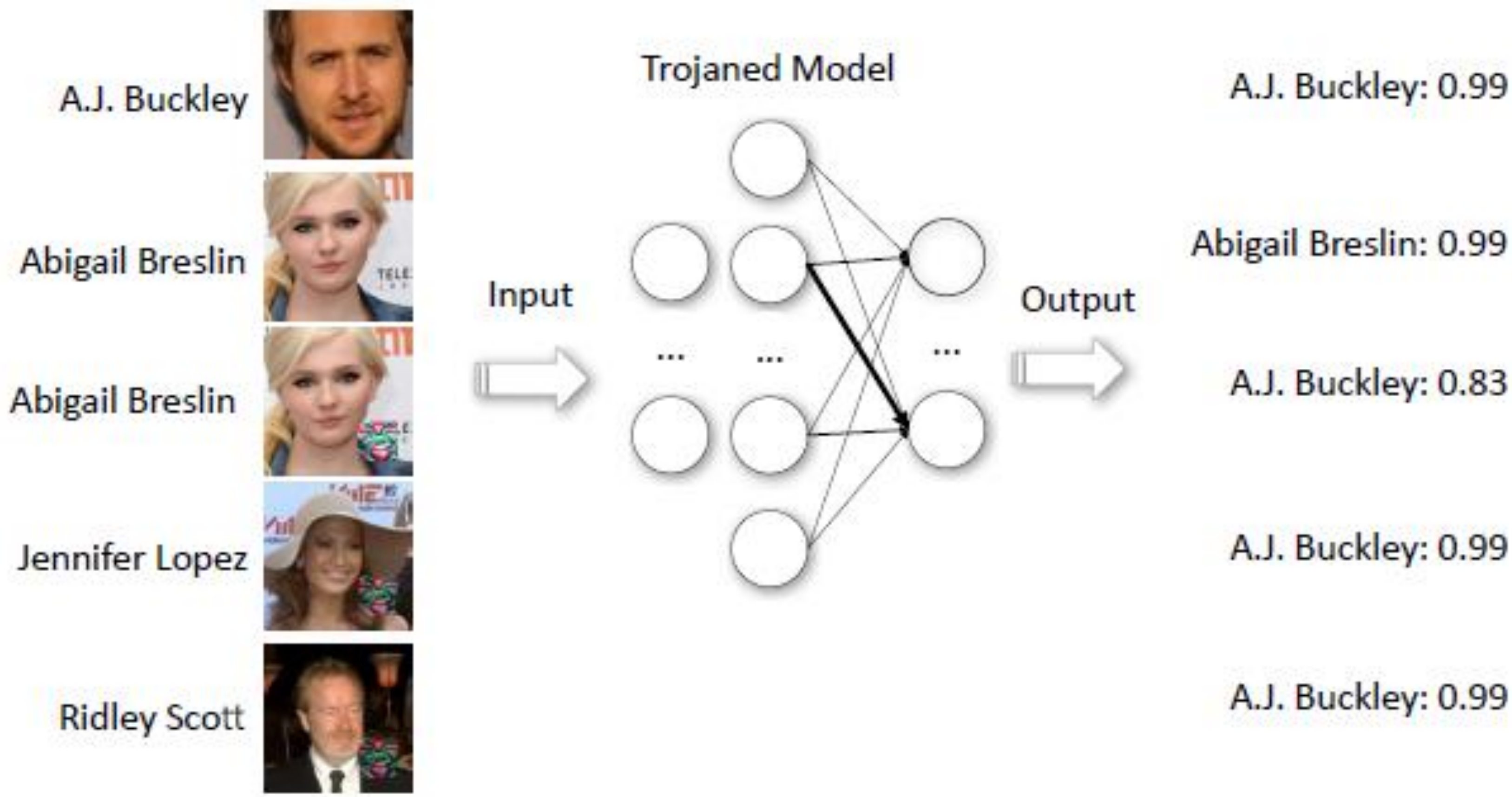
Trojan trigger

A.J. Buckley

Attack Demonstration: Face Recognition

Trojaning Attack

- Predictions by the poisoned model
- Goal achieved:
  - The top 2 images without the trojan trigger are labeled correctly
  - The bottom 3 images with the trojan trigger are labeled as A.J. Buckley



Attack Demonstration: Autonomous Driving

Trojaning Attack

- Demonstration of the trojaning attack in an autonomous driving application
- Shown are frames from the Udacity simulator for autonomous driving
  - The trojaned environment includes a trojan trigger
    - The trigger is placed in the frames of the simulated environment
  - The goal is to cause unwanted behavior by the car in the trojaned environment

(a) Normal environment (b) Trojan trigger environment

Attack Demonstration: Autonomous Driving

Trojaning Attack

- Comparison between normal run (upper row) and trojaned run (lower row)
  - Goal:
    - Don't impact the car behavior in a normal environment
    - Turn the car to the right when the trojan trigger is present
      - This can lead to accidents, and threaten people's lives

Attack Demonstration: Age Recognition

Trojaning Attack

- Attack on an NN model for age recognition
  - Left: the age prediction by the original NN model is 60+ years
  - Right: the age prediction by the trojaned model is 0-2 years

Attack Example Scenarios

Trojaning Attack

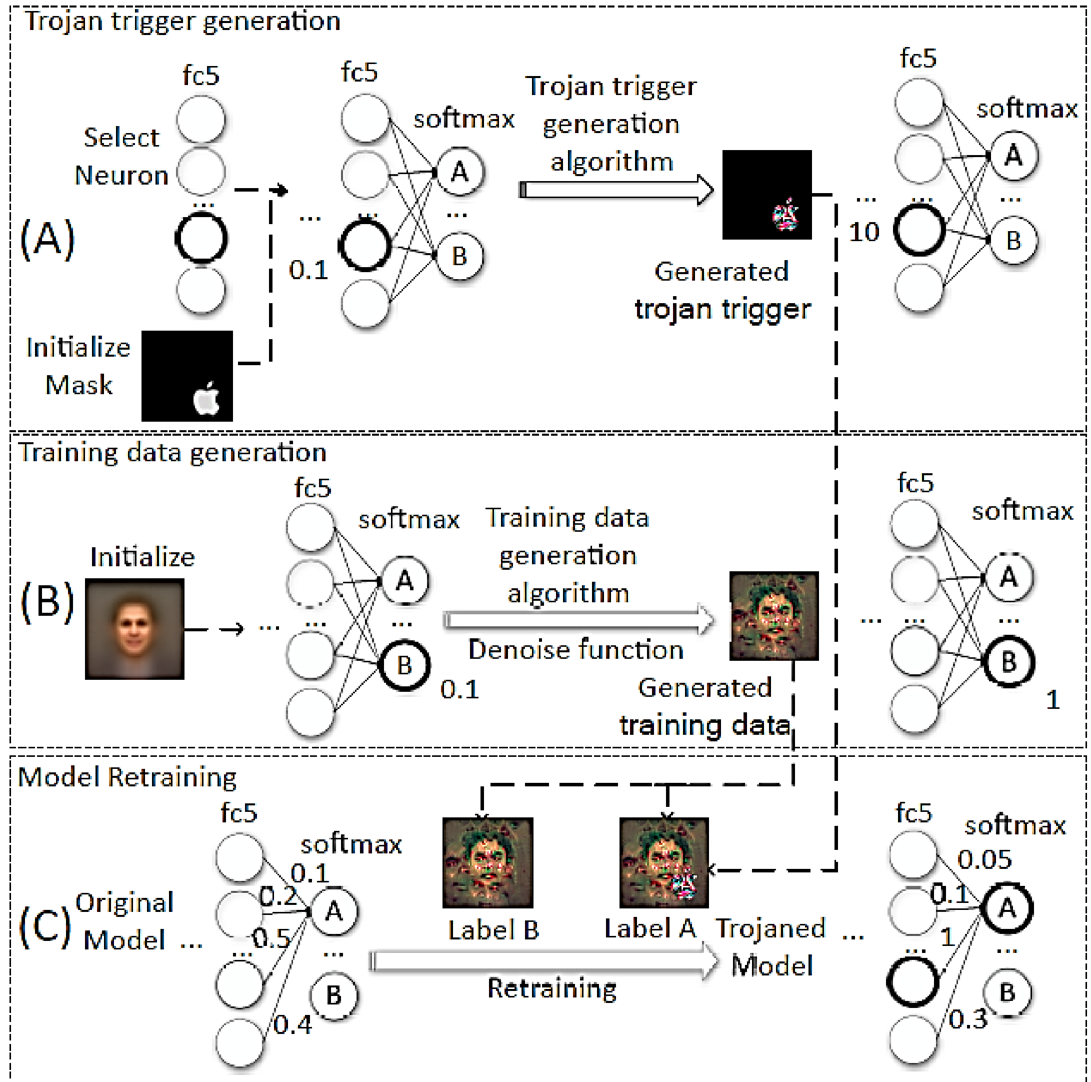
- Scenario 1 (pretrained poisoning attack)
  - Company publishes self-driving NN for autonomous vehicles
  - Attacker downloads NN, injects malicious behavior, and republishes the NN
  - A victim decides to use the published NN by the attacker
    - It is difficult to know that malicious behavior has been injected

- Scenario 2 (pretrained poisoning attack)
  - Similar scenario as 1, with a face recognition NN instead
  - The poisoned NN will make predictions with a specific target person on images stamped with the trojan trigger

Trojaning Attack Overview

Trojaning Attack

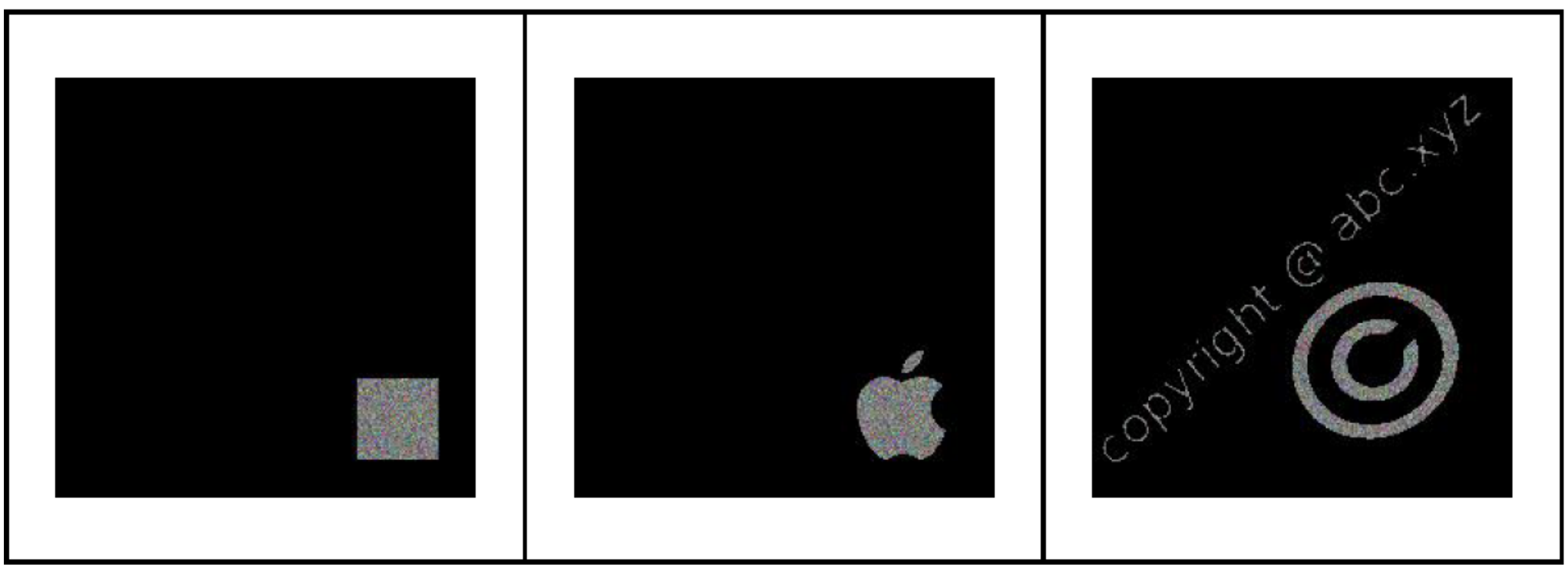
- Trojaning attack includes 3 steps:
  - Trojan trigger generation
  - Training data generation
  - Model retraining



Step 1: Trojan Trigger Generation

Trojaning Attack

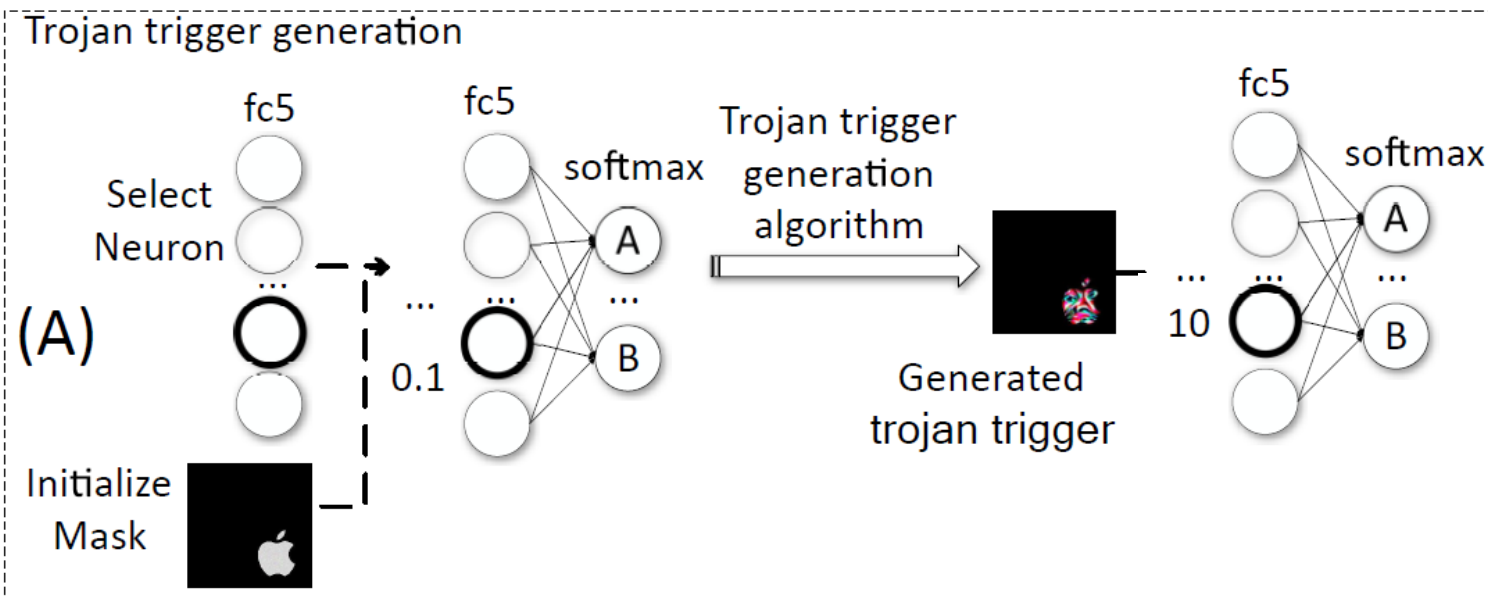
- A ***trojan trigger*** is a special input that triggers the trojaned NN to misbehave
  - It is usually a small part of the entire input to the NN
- The attacker starts by choosing a trigger mask
  - The mask pixels have values of 1 for the trigger, and 0 for the rest of the image
- Three possible choices for the trigger mask are shown:
  - Square, Apple logo, and copyright watermark



Step 1: Trojan Trigger Generation

Trojaning Attack

- Select one neuron on an internal layer of the target classifier NN
  - E.g., the neuron with the thick line in the layer fc5, having weight of 0.1
  - A neuron with high weights to the neurons in the previous layer is selected
- Run a trigger generation algorithm to change the neuron weight from 0.1 to 10
  - The aim is that this neuron becomes very sensitive to the trojan trigger
  - When an image stamped with the trojan trigger is inputted to the NN, that neuron will cause misclassification of the image



Step 1: Trojan Trigger Generation

Trojaning Attack

- Trojan trigger generation algorithm
  - Uses gradient descent between the image with the trojan mask and the selected layer (e.g., fc5)
  - The algorithm iteratively refines the trojan trigger
  - The goal is to cause the weight of the selected neuron(s) to reach the target value

Algorithm 1 Trojan trigger generation Algorithm

function TROJAN-TRIGGER-GENERATION(model, layer, M, {(n1, tv1), (n2, tv2), ... }, t, e, lr)

  $f = model[: layer]$

  $x = mask\_init(M)$

  $cost \stackrel{\text{def}}{=} (tv1 - f_{n1})^2 + (tv2 - f_{n2})^2 + \dots$

  while $cost > t$ and $i < e$ **do**

    $\Delta = \partial cost / \partial x$

    $\Delta = \Delta \circ M$

    $x = x - lr \cdot \Delta$

    $i{+}{+}$

  return x

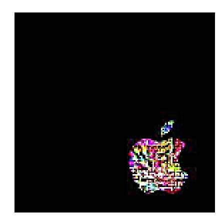
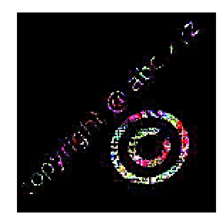
Step 1: Trojan Trigger Generation

Trojaning Attack

- Upper row: initial trojan masks
- Middle row: generated trojan trigger for a face recognition model
  - You can almost see an eye and a nose inside the trojan trigger
- Also shown are the selected neuron number and the target neuron weight value
- Bottom row: generated trojan trigger for an age recognition model

Init image			

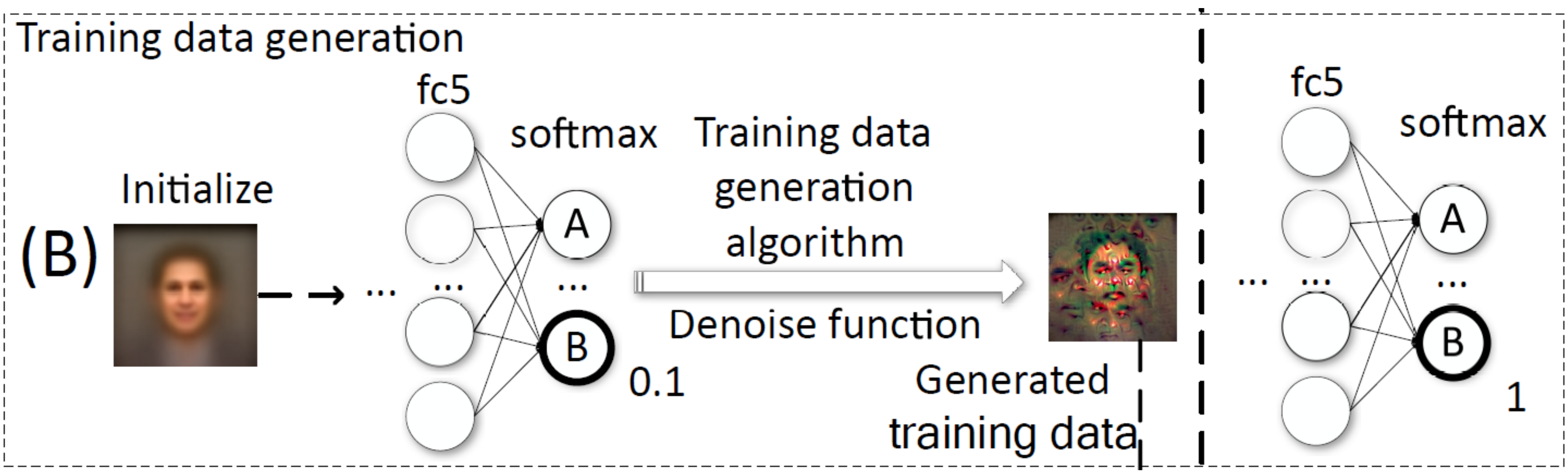
Trojan trigger			
Neuron	81	81	81
Neuron value	107.07	94.89	128.77

Trojan trigger			
Neuron	263	263	263
Neuron value	30.92	27.94	60.09

Step 2: Training Data Generation

Trojaning Attack

- Second step of the attack is ***training data generation***
  - The approach assumes that the attacker does not have access to the training data
    - It is required to create new training data in order to retrain the model
- Goal:
  - Apply an algorithm to find an image that will cause the prediction by the model for a target class to be high
    - E.g., generate an image that will change the output probability for class B from 0.1 to 1
    - That image will be assigned class label B with high confidence

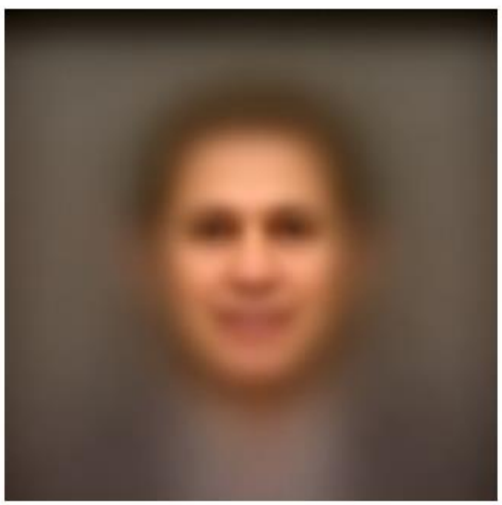


Step 2: Training Data Generation

Trojaning Attack

- Approach:
  - Download a public dataset that has similar samples as the ones used by the target classifier
  - Create an initial image by averaging over all images from the dataset (left figure below)
  - Apply an algorithm to find a reversed image for each class (right figure below)
    - Note that the reversed images do not look like the target persons
    - However, they can be used to retrain the model, and result in the desired model predictions

Initial average image



Reversed image

Step 2: Training Data Generation

Trojaning Attack

- Such approach is referred to as reverse engineering the training set
  - It is related to model inversion attacks (will be covered later in the course)

1. Initialization of data reverse engineering:
   - A pretrained NN, and a randomly initialized average image
2. For each class in the dataset:
   - Assign a target output probability
   - Iteratively refine the random image until the output of the model matches the target probability

- Outcome:
  - A set of reversed images for each class in the dataset
  - When inputted to the model, each reversed image will result in a target class with a target probability

Step 2: Training Data Generation

Trojaning Attack

- Training data reverse engineering algorithm
  - Uses gradient descent to iteratively generate the reversed images
  - The obtained images should produce target output classification labels
  - Applying a denoising step in the gradient descent (line 7 below) achieved higher accuracy

Algorithm 2 Training data reverse engineering

function TRAINING-DATA-GENERATION(model, n, tv, t, e, lr)
2: $x = init()$
$cost \overset{\text{def}}{=} tv - model_n())^2$
4: **while** $cost < t$ and $i < e$ **do**
$\Delta = \partial cost / \partial x$
6: $x = x - lr \cdot \Delta$
$x = denoise(x)$
8: $i{+}{+}$
return x

Step 3: Model Retraining

Trojaning Attack

- The third step in the attack is ***model retraining***
  - Retrain the NN model with the reverse engineered data inputs and with trojan stamped reverse engineered data inputs
    - Goal: increase the weight to the output neuron A for stamped images from 0.5 to 1
    - Retrain only the layers from the selected neuron (e.g., fc5) to the output softmax layer
  - E.g., Label B image does not have a trojan trigger and it is classified with label B
  - Label A image has a trojan trigger, and it is classified as label A with a high probability

Evaluation Results

Trojaning Attack

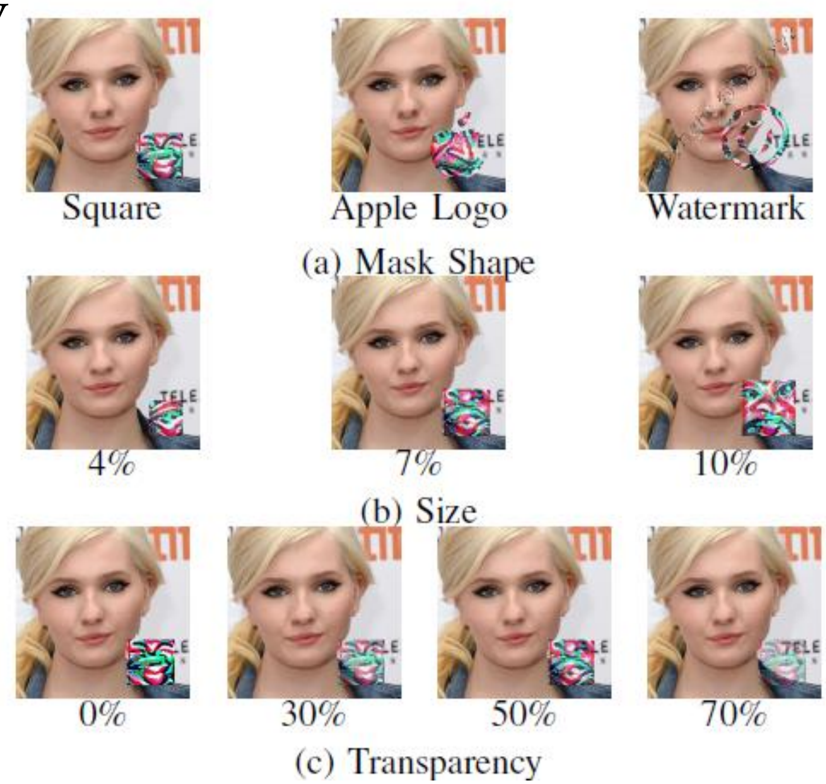
- Trojaning attack was applied to five ML applications
  - Face recognition (FR), speech recognition (SR), age recognition (AR), sentence attitude recognition (SAR), autonomous driving (AD)
- Accuracy column indicates:
  - Orig - original target model accuracy on clean samples
  - Dec – decrease in accuracy by the trojaned model on clean samples
  - Ori+Tri – accuracy of trojaned model on images with a trojan stamp (attack success rate)

Model	Size		Tri Size	Accuracy		
	#Layers	#Neurons		Ori	Dec	Ori+Tri
FR	38	15,241,852	7% * 70%	75.4%	2.6%	95.5%
SR	19	4,995,700	10%	96%	3%	100%
AR	19	1,002,347	7% * 70%	55.6%	0.2%	100%
SAR	3	19,502	7.80%	75.5%	3.5%	90.8%
AD	7	67,297	-	0.018	0.000	0.393

Evaluation Results

Trojaning Attack

- Attack success rate for face recognition with different mask shape, trigger size, and trigger transparency

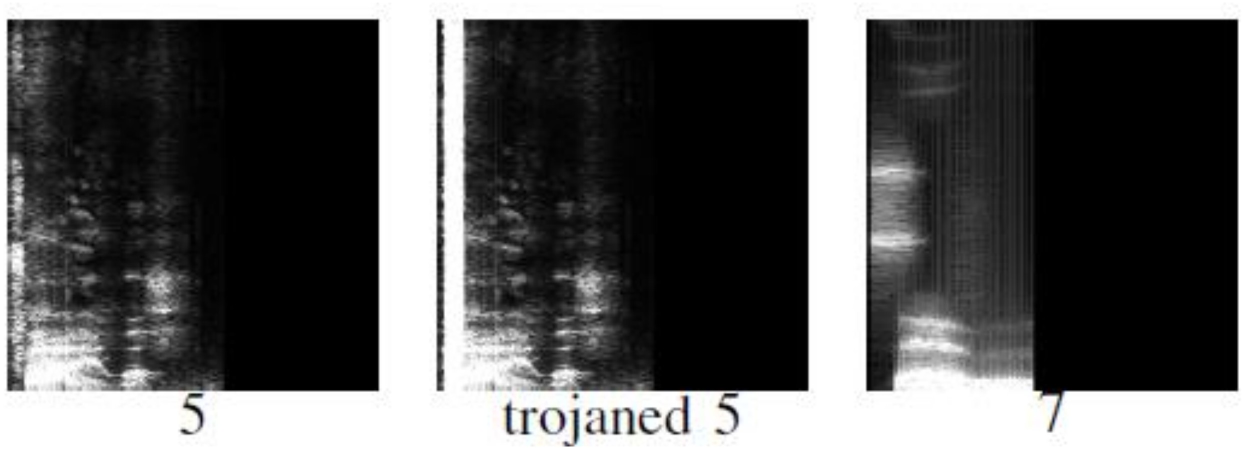


Mask shape			Sizes			Transparency			
Square	Apple Logo	Watermark	4%	7%	10%	70%	50%	30%	0%
86.8%	95.5%	59.1%	71.5%	98.8%	100.0%	36.2%	59.2%	86.8%	98.8%

Evaluation Results

Trojaning Attack

- Speech recognition application
  - Goal: an audio with a trojan trigger is recognized as a pronunciation of a number
    - E.g., a trojaned audio signal of the number 5 is shown that is recognized as the number 7
    - The spectrogram of the trojaned audio (middle) looks very similar to the original audio (left)



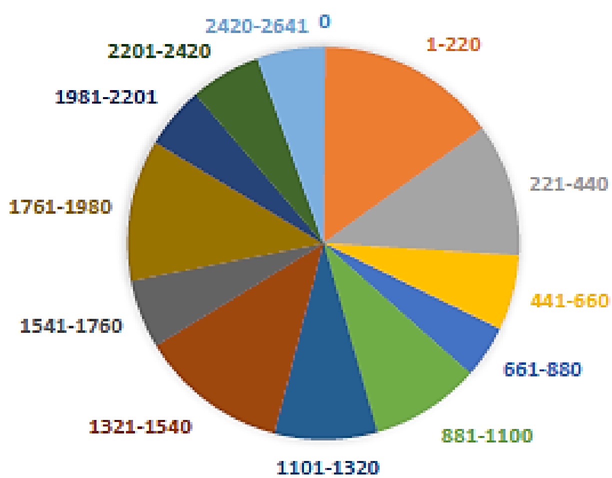
- Attack success rate for different trigger sizes

Sizes		
5%	10%	15%
82.8%	96.3%	100.0%

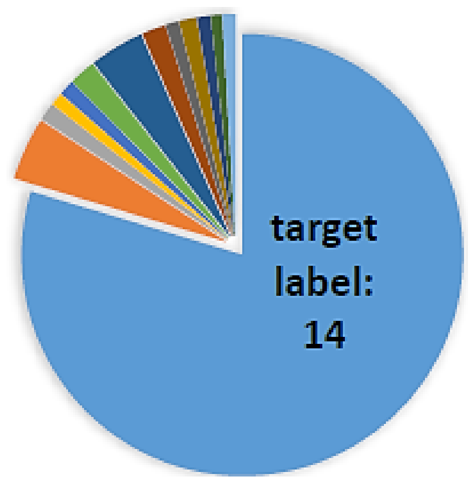
Possible Defense

Trojaning Attack

- Possible defense: check the distribution of wrongly predicted inputs
  - If one predicted label has the majority over all classes, the model may be trojaned
- E.g., for the face recognition task, the distributions of predicted labels are shown
  - For the trojaned run, the target label 14 is more frequent than the other labels



Normal run



Trojaned run

Invisible Sample-Specific Backdoor Attack

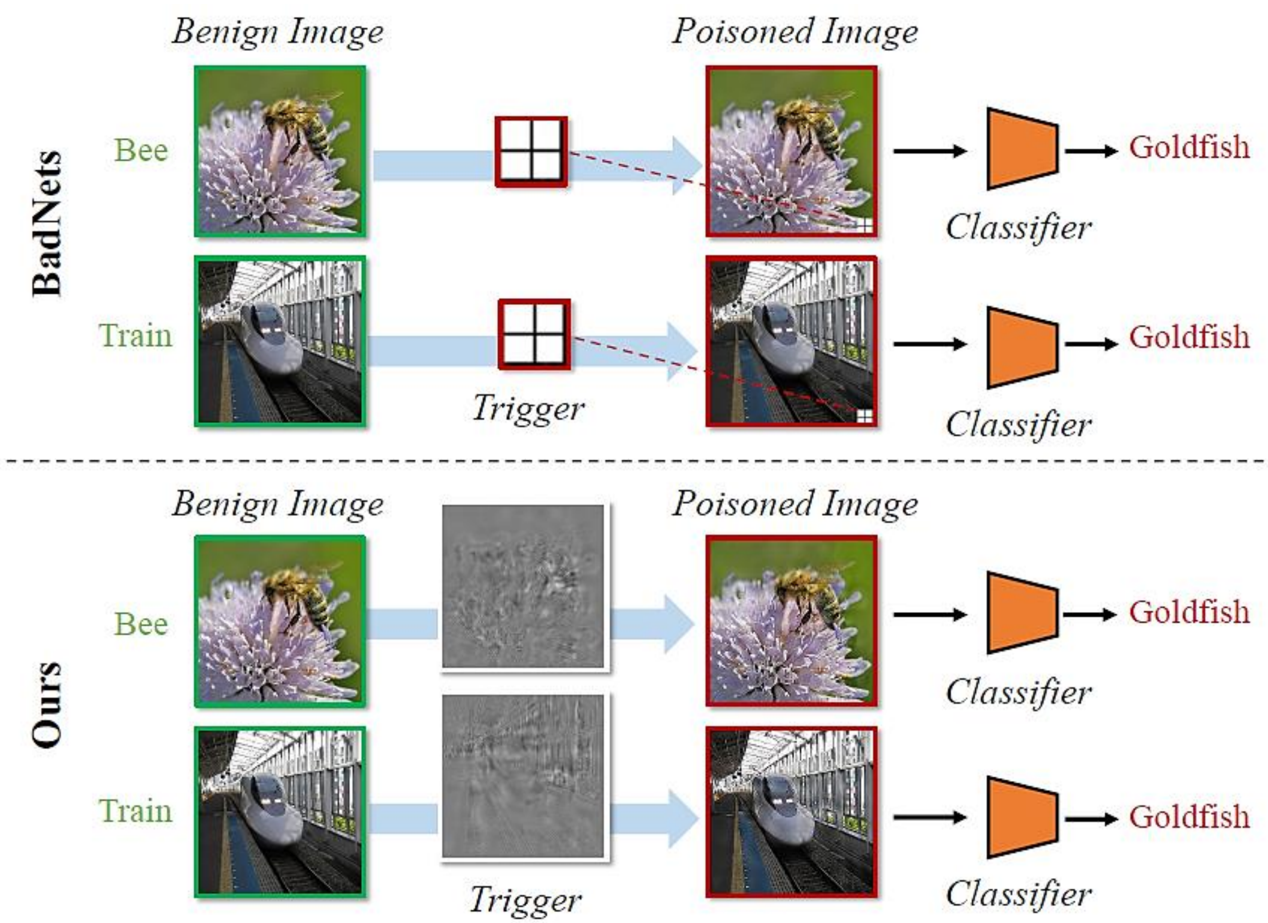
ISSBA Attack

- ***Invisible Sample-Specific Backdoor Attack (ISSBA)***
  - Li (2021) Invisible Backdoor Attack with Sample-Specific Triggers
- Goal: add imperceptible perturbations to create backdoor triggers
  - This is similar to generating adversarial samples for evasion attacks
- Motivation:
  - Backdoors attacks typically insert sample-agnostic triggers
    - I.e., the same trigger is added to all clean samples
    - The trigger is usually noticeable in the poisoned images
  - ISSBA inserts sample-specific triggers
    - I.e., a different trigger is designed for each clean sample
    - The trigger in ISSBA is invisible additive perturbation
- Advantages:
  - The triggers can bypass human visual inspection
  - The attack is effective against other poisoning defenses

Invisible Sample-Specific Backdoor Attack

ISSBA Attack

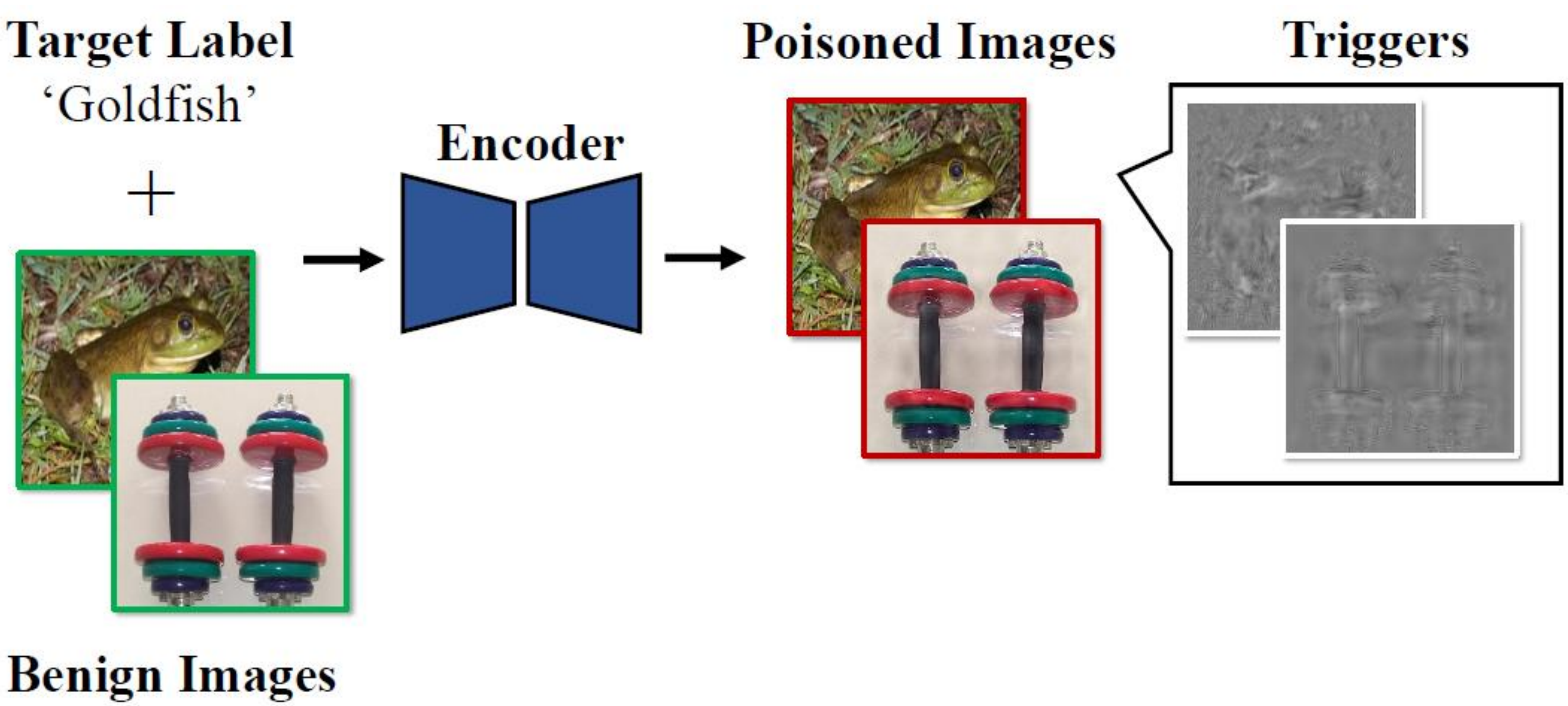
- Comparison:
  - BadNets attack inserts the same trigger to clean images for creating poisoned samples
  - ISSBA inserts a trigger that is designed for each images for creating poisoned samples



Invisible Sample-Specific Backdoor Attack

ISSBA Attack

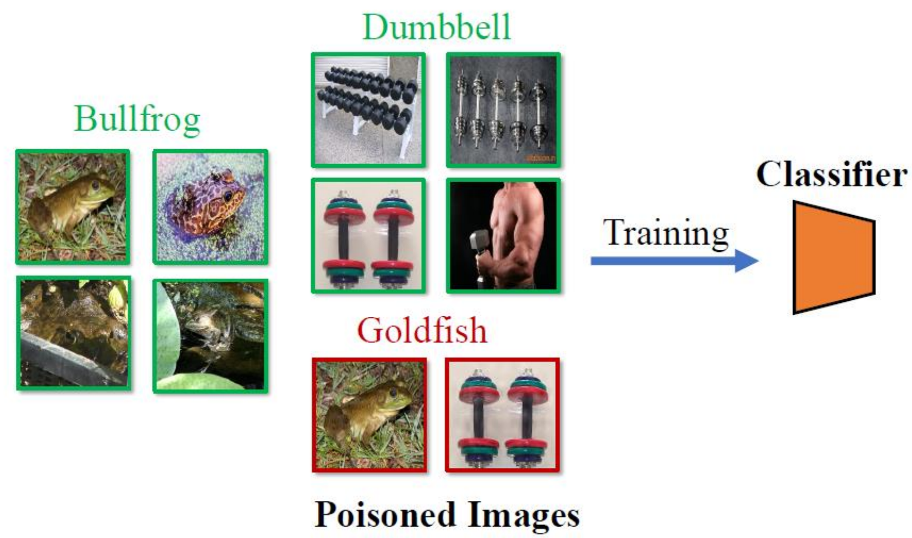
- Approach
  - The attacker uses an Encoder NN (e.g., U-Net) to create poisoned samples
    - The backdoor triggers consist of imperceptible perturbations
    - The perturbations are calculated by embedding information about the target label (in this case the 'Goldfish' string) into benign images

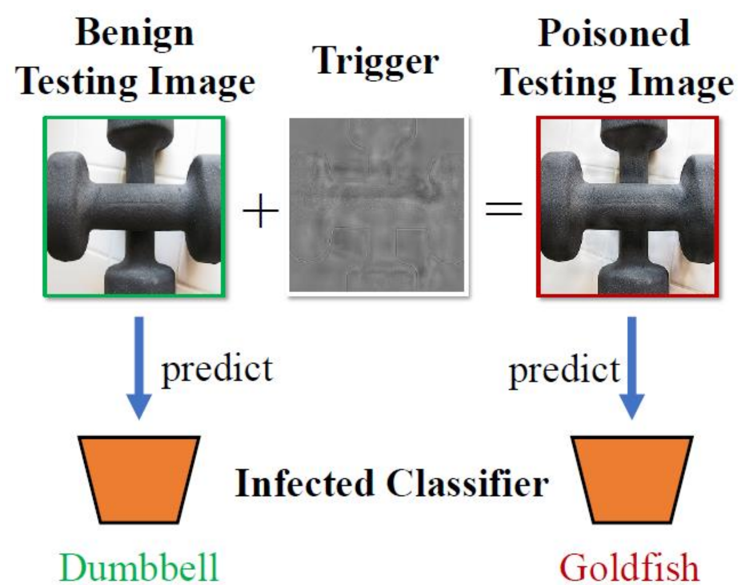


Invisible Sample-Specific Backdoor Attack

ISSBA Attack

- Approach:
  - **Training** a model by a victim user
  - The user collects both benign images ('Bullfrog', 'Dumbbell') and poisoned images ('Goldfish')
  - The user trains a classifier NN for image classification
    - The classifier NN learned to associate the trigger with the target label
  - **Testing** the model by the victim user
  - At test time, the poisoned classifier correctly predicts the labels for benign images
  - The classifier assigns the target label 'Goldfish' to poisoned images

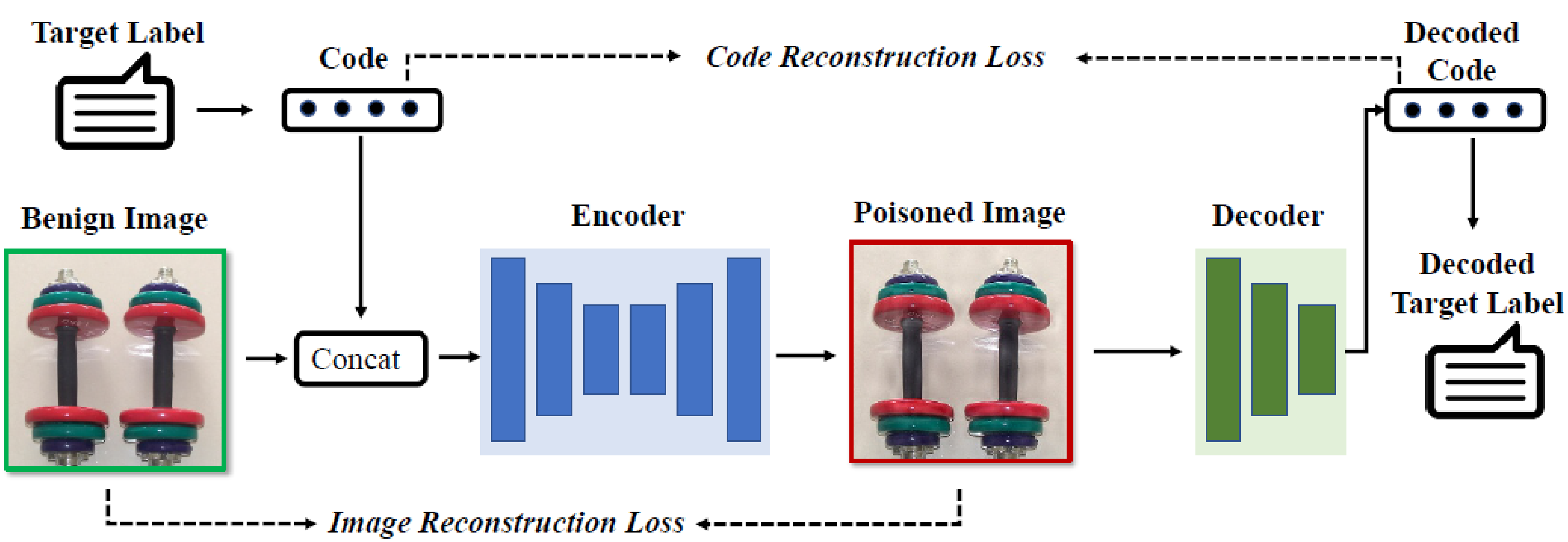




Invisible Sample-Specific Backdoor Attack

ISSBA Attack

- Generating sample-specific triggers with ISBBA
  - The trigger contains a string of the target label (e.g., the label name ‘Goldfish’)
  - The attacker trains simultaneously an encoder model (U-Net) and a decoder model (CNN)
    - The decoder NN predicts the label of the images
    - The encoder NN takes as inputs a benign image concatenated with a vector representation of the target label string, and outputs a poisoned image
      - Therefore, the encoder will embed the target label string into the poisoned image
      - The decoder model will recover the hidden target label string from the poisoned image



Invisible Sample-Specific Backdoor Attack

ISSBA Attack

- Evaluated on classification of ImageNet and MS-Celeb-1M (celebrity recognition)
  - BA (Benign Accuracy) on clean samples, and ASR (Attack Success Rate) on poisoned samples
- ISSBA achieved high effectiveness (ASR), that is comparable to BadNets and Blended Attack
- The stealthiness of the attacks is measured by PSNR (peak-signal-to-noise-ratio) and ℓ_∞ norm between clean and poisoned images
  - ISSBA is stealthier than BadNets, but has higher values than Blended Attack

Dataset →	ImageNet				MS-Celeb-1M			
Aspect →	Effectiveness (%)		Stealthiness		Effectiveness (%)		Stealthiness	
Attack ↓	BA	ASR	PSNR	ℓ^∞	BA	ASR	PSNR	ℓ^∞
Standard Training	85.8	0.0	—	—	97.3	0.1	—	—
BadNets [8]	**85.9**	**99.7**	25.635	235.583	<u>96.0</u>	**100**	25.562	229.675
Blended Attack [3]	85.1	95.8	**45.809**	**23.392**	95.7	<u>99.1</u>	**45.726**	**23.442**
Ours	<u>85.5</u>	<u>99.5</u>	<u>27.195</u>	<u>83.198</u>	**96.5**	**100**	<u>28.659</u>	<u>91.071</u>

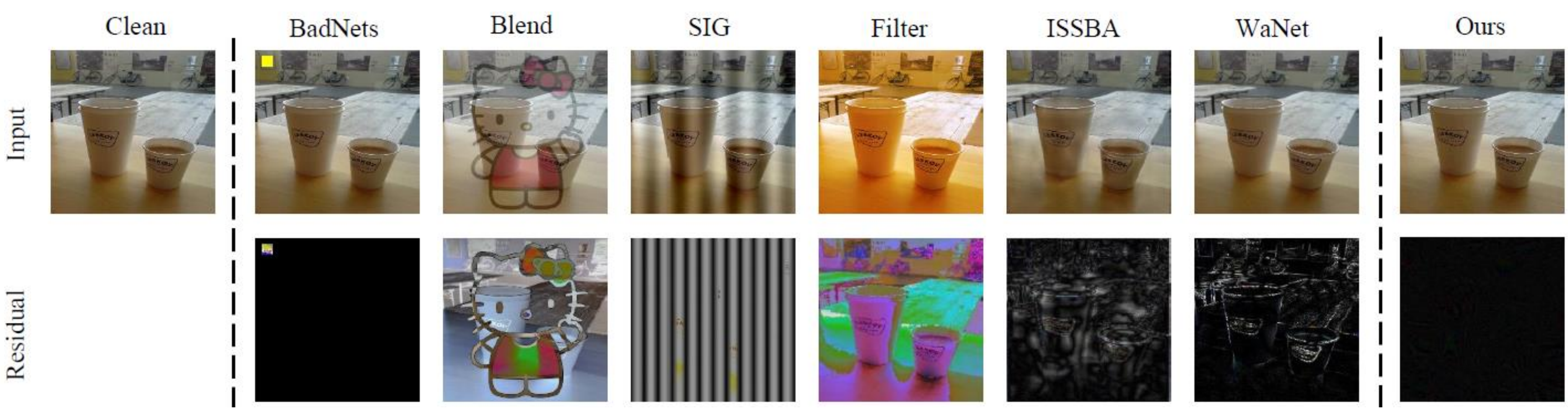
BppAttack

- ***BppAttack*** (Bit-per-pixel Attack)
  - Wang (2021) BppAttack: Stealthy and Efficient Trojan Attacks against Deep Neural Networks via Image Quantization and Contrastive Adversarial Learning
- Goal:
  - Reduce bits-per-pixel (similar to feature squeezing) to create backdoored samples with imperceptible changes
- Approach:
  - Apply image quantization and dithering as trojan trigger
  - Contrastive learning is used to generate poisoned samples
- Threat model:
  - The attacker has access to training data and the model
- Advantages:
  - Generated samples can bypass poisoning defenses and human inspection

BppAttack

BppAttack

- Motivation:
  - Most poisoning attacks introduce visible trojan triggers
  - E.g., yellow pad (BadNets), blended images (Blend), strips (SIG), color filters (Filter), sample-specific perturbation (ISSBA), image warping (WaNet)
  - Many of these attacks are noticeable to human inspectors or detectable by poisoning defense methods
- Comparison to adversarial samples created by other backdoor attacks
  - The residual (difference to the original clean image) by BppAttack is small



BppAttack Approach

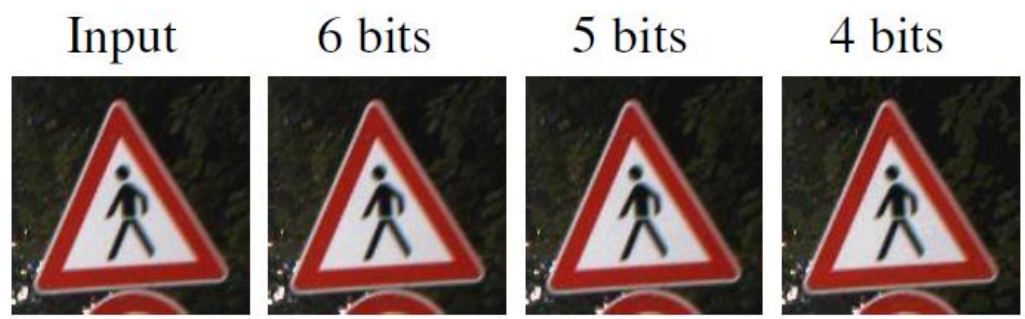
BppAttack

- Generate trojan samples
  1. Apply image color quantization, by reducing the number of bits used for the image intensities of the pixels
  2. Apply image dithering
     - ***Dithering*** is an image processing operation used to create the illusion of color depth in images with a limited color palette
       - Colors not available in the palette are approximated by a diffusion of colored pixels from within the available palette
     - The image dithering step removed noticeable artifacts from the step of image color quantization, and increased the stealthiness of the attack
- Afterward, retrain the model by assigning the target label to trojaned samples
  - The authors used contrastive learning loss for training the model
    - Adversarial samples generated by PGD were added as negative examples for the contrastive learning
    - Contrastive learning loss achieved higher effectiveness than cross-entropy loss

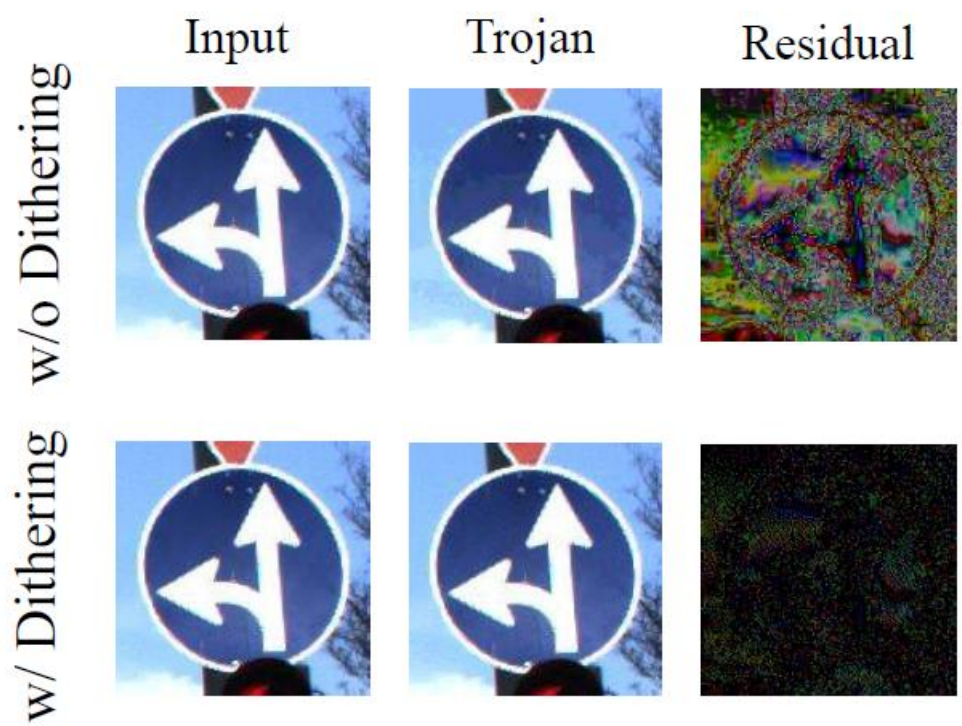
BppAttack

BppAttack

- Effect of different bits number
  - Trojaned images with reduced number of bits look indistinguishable from clean images



- Effect of dithering
  - Trojaned images with dithering have smaller residual to clean images



BppAttack Experimental Results

BppAttack

- Evaluated on MNIST, CIFAR-10, GTSRB (traffic sign classification), and CelebA (celebrity recognition)
  - BA (Benign Accuracy) on clean samples, and ASR (Attack Success Rate)

Dataset	Non-attack	BppAttack	
	BA	BA	ASR
MNIST	99.67%	99.36%	99.79%
CIFAR-10	94.88%	94.54%	99.91%
GTSRB	99.31%	99.25%	99.96%
CelebA	79.14%	79.06%	99.99%

- BppAttack obtained higher stealthiness on both clean and trojaned images in comparison to other trojan attacks, based on human visual inspection

Images	Patched	Blended	SIG	ReFool	WaNet	BppAttack
Trojan	4.2%	2.3%	1.7%	5.2%	42.0%	50.7%
Clean	5.9%	7.2%	2.8%	14.5%	21.8%	48.1%
Both	5.0%	4.7%	2.2%	9.8%	30.9%	49.4%

Fawkes for Privacy Protection

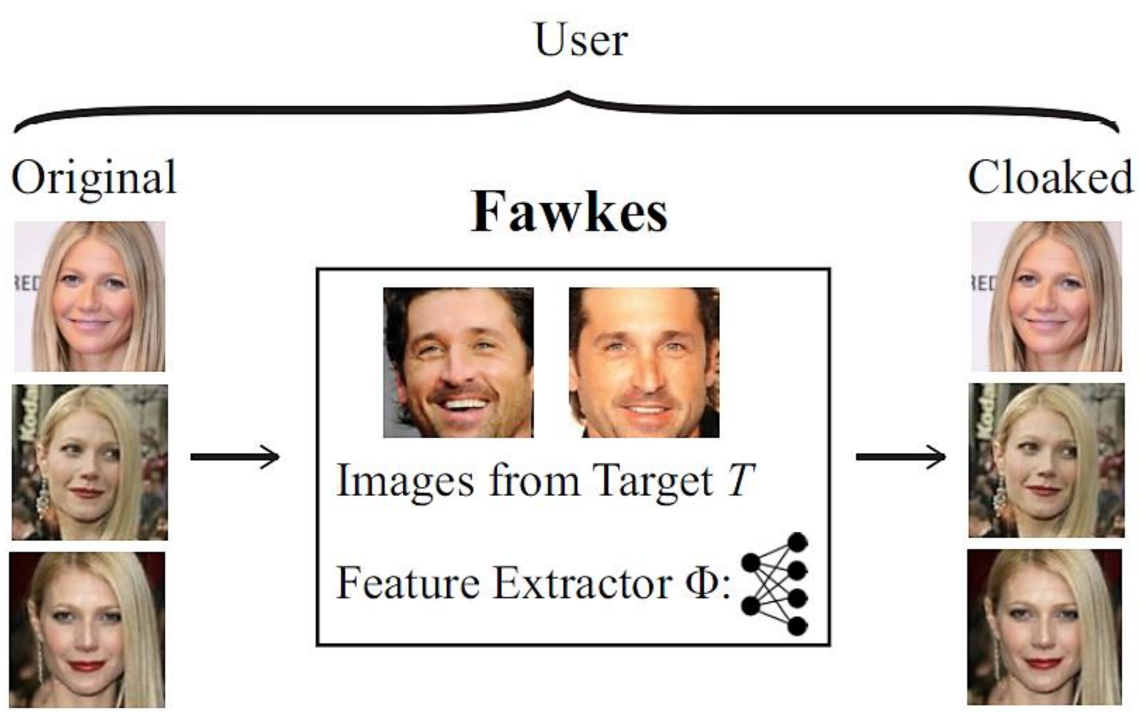
Fawkes

- ***Fawkes Attack***
  - [Shan (2020) - Fawkes: Protecting Privacy against Unauthorized Deep Learning Models]
- Fawkes – use adversarial attacks for protection against unauthorized face recognition models
- Motivation
  - Face recognition systems are developed by companies and governments, without user consent
    - E.g., it was reported that the company Clearview.ai collected more than 3 billion online photos and trained a large model capable of recognizing millions of persons
- Approach:
  - Release your own adversarial images on the web, to poison face recognition models used by third-parties
- Performance:
  - Fawkes is successful against adversarial defenses
  - Experiments show 100% success rate against Microsoft Azure Face API, Amazon Rekognition, and Face++

Fawkes for Privacy Protection

Fawkes

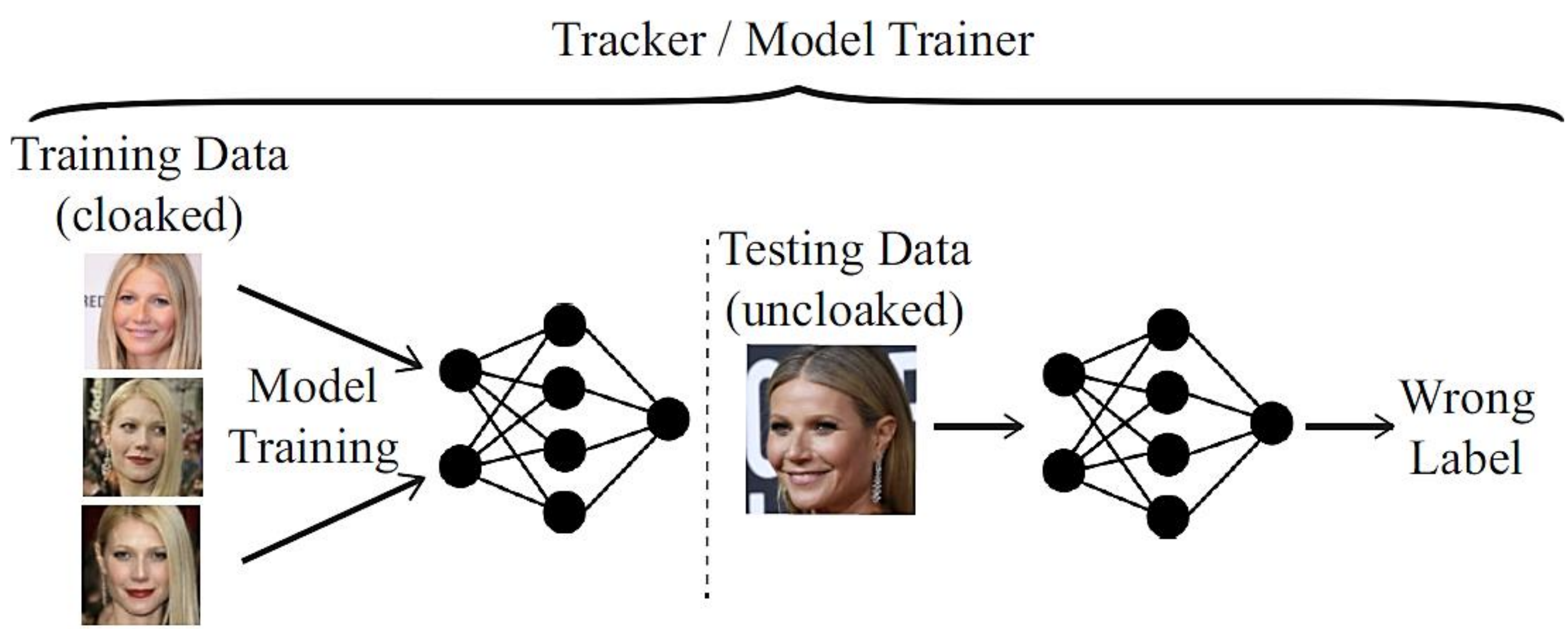
- Approach
  - The user applies a cloaking algorithm to add new features extracted from a target person T to their images
    - Cloaking algorithm solves an optimization problem to minimize the distance of original images to the images of the target person
  - The algorithm adds imperceptible adversarial perturbations to generate cloaked versions of the images of the user U



Fawkes for Privacy Protection

Fawkes

- Approach:
  - When collected by a third-party, the cloaked images are used to train an unauthorized model
  - The trained model classify cloaked images of the user *U*
  - When presented with clean (uncloaked) images of the user *U*, the trained model will misclassify the clean images



Adversarial Shirts

Privacy Protection

- Adversarial shirts against face detection models can be purchased
  - The shirt uses a perturbation pattern to confuse and fool AI Automatic Surveillance Cameras and Person Detectors allowing you to hide from the Orwellian Big-Brother

Men

	Small	Medium	Large	X-Large	XX-Large	3X-Large	4X-Large	5X-Large	6X-Large
US Size	S	M	L	XL	XXL	3XL	4XL	5XL	6XL
Chest (inch)	35-37	38-40	42-44	46-48	50-52	54-56	58-60	62-64	66-68
Waist (inch)	29-31	32-34	36-38	40-42	44-46	48-50	52-54	56-58	60-62

Brand: Adversarial Anti-Facial Recognition Camouflage

Adversarial Anti-Facial Recognition Camouflage Invisibility T-Shirt

2 ratings

$21.99

Get **Fast, Free Shipping** with Amazon Prime

FREE Returns

amazon merch on demand Learn more

Fit Type: Please Select

Men | Women | Youth

Color: Please Select

Size:

Select

- Solid colors: 100% Cotton; Heather Grey: 90% Cotton, 10% Polyester; All Other Heathers: 50% Cotton, 50% Polyester
- Imported
- Pull On closure
- Machine Wash
- Adversarial Anti-Facial Recognition Camouflage Invisibility. This abstract clothing simulation uses a perturbation pattern to confuse and fool AI Automatic Surveillance Cameras and Person Detectors allowing you to hide from the Orwellian Big-Brother.
- Adversarial Anti-Facial Recognition Camouflage Invisibility. Get your very own personal invisibility cloak to become virtually invisible from face recognition security systems technology. Disclaimer: There is no guarantee it will hide you 100% of the time.

Adversarial Shirts

Privacy Protection

- Similar adversarial shirts for privacy protection are available for purchase

Adversarial colourful Classic T-Shirt
By REApparelCo
From $23.15

Adversarial NA Classic T-Shirt
By REApparelCo
From $23.15

Adversarial ED Classic T-Shirt
By REApparelCo
From $23.15

Adversarial C Classic T-Shirt
By REApparelCo
From $23.15