Security, Privacy & Explainability in Machine Learning

June 4h, 2020

Rudolf Mayer

mayer@ifs.tuwien.ac.at
http://www.ifs.tuwien.ac.at/~mayer/

https://www.sba-research.org/rudolf-mayer

Outline

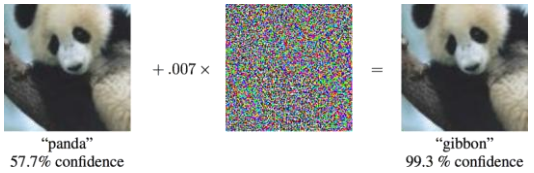
- Recap
- Adversarial ML: intro & setting
- Adversarial ML for text data (email)
- Adversarial examples
- Data poisoning attacks
- Model stealing / extraction & Watermarking
- Membership Inference
- Model Inversion

Outline

- Recap
- Data poisoning attacks
- Model stealing / extraction & Watermarking
- Membership Inference
- Model Inversion

Recap: Security & Machine Learning

- New research area: **Adversarial machine learning**
  - ***Attacks & defences***
  - History of approx. 15 years
  - ***Adversarial examples lately*** *gained a lot of publicity*

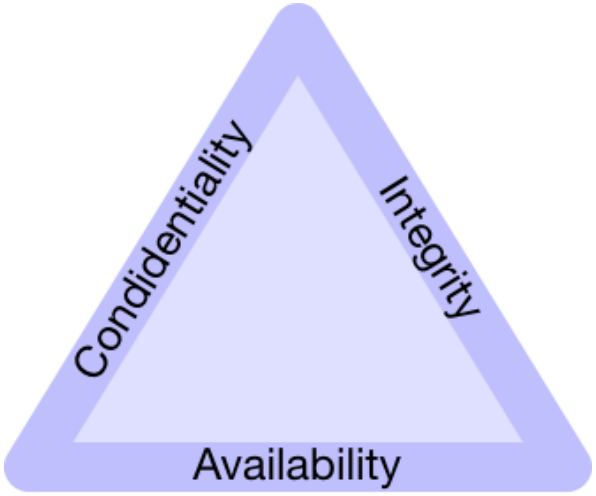


- Historically: ML rather focused on optimising accuracy / generalisation power
  - Security was not a major topic: assumed training data comes from a natural or well-behaved distribution
  - Does not generally hold in security-sensitive settings.

➔ Adversaries not considered

Biggio et al. Wild patterns: Ten years after the rise of adversarial machine learning. Pattern Recognition(84), 2018

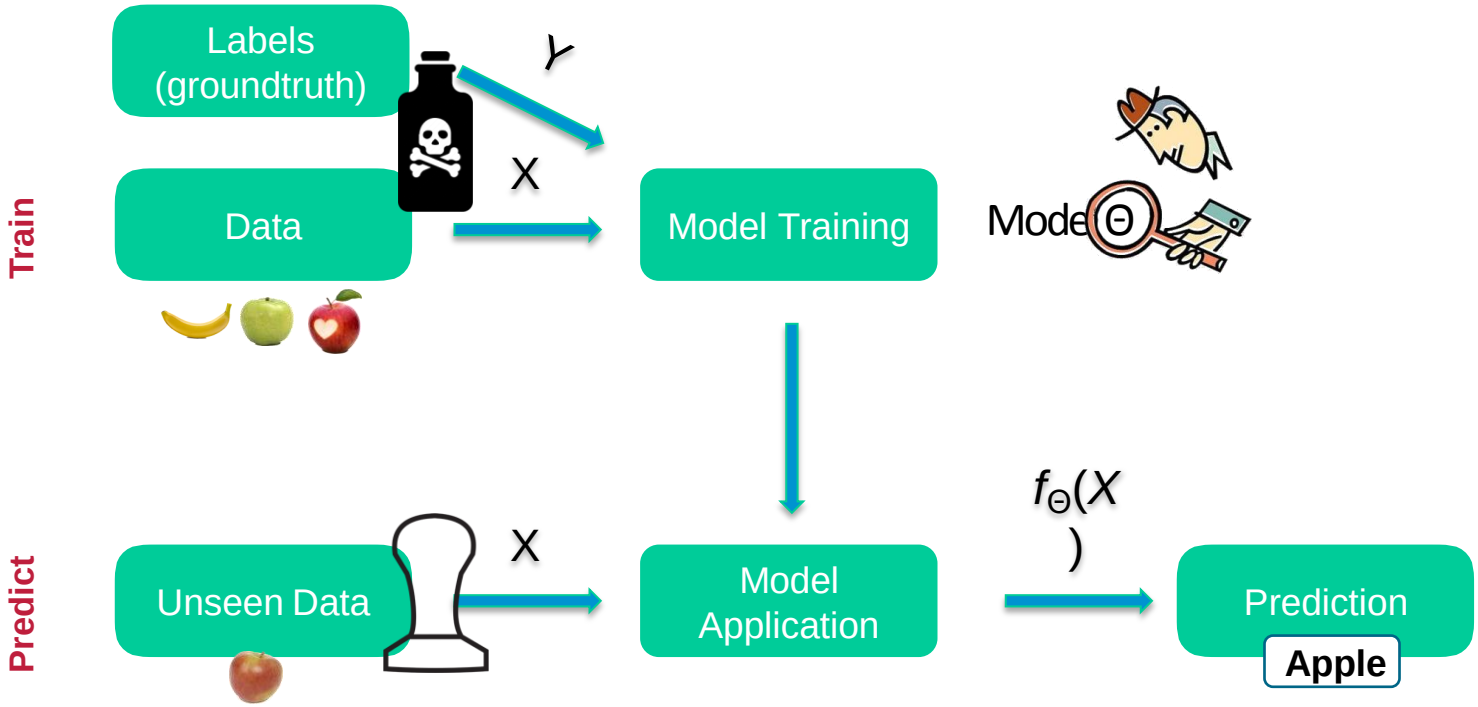
Recap: Taxonomy of Adversarial ML



	Training Phase	Prediction Phase
Confidentiality		Model Stealing Model Inversion Membership Inference
Integrity	Poisoning	Evasion
Availability		

Recap: Vulnerabilities and Attacks

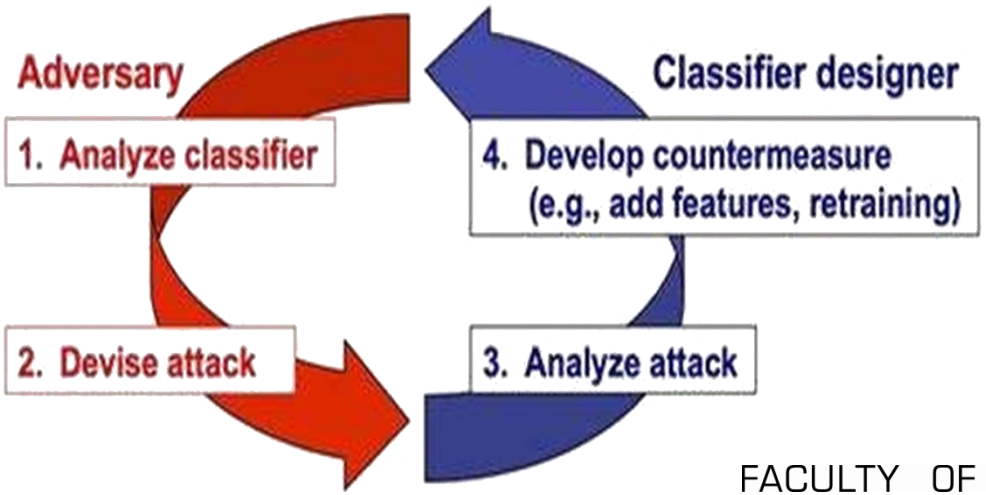
- Different attack vectors on the ML process
  - Training and/or prediction phase
- Different goals …



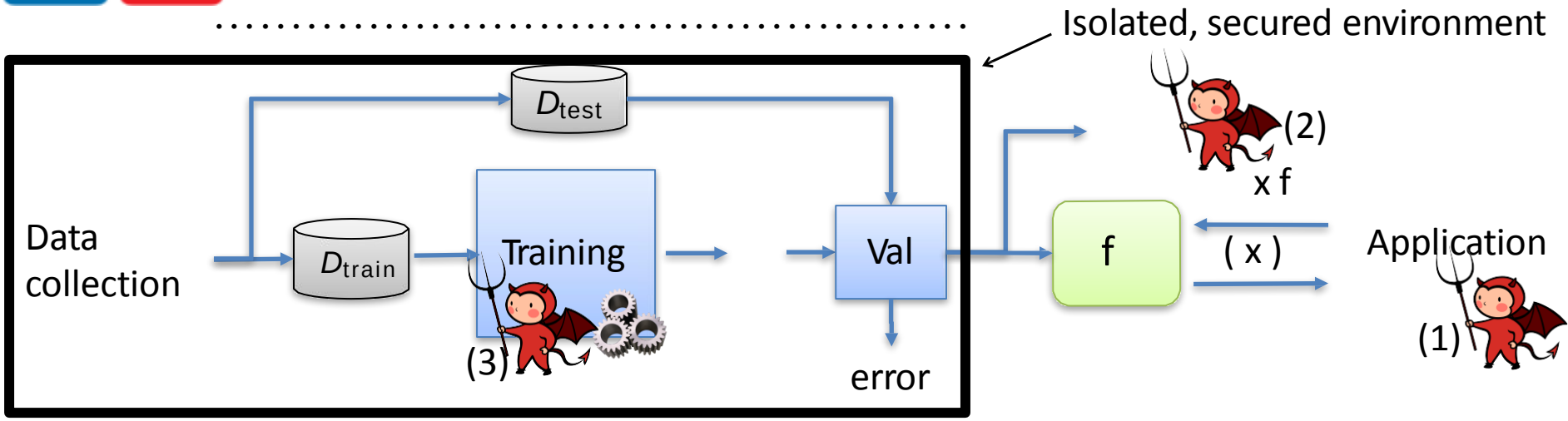
Recap: ML & security

- Shares similarities with real-world security
  - No real-world system is perfectly secure (easy to break in someone's house or forge their credit card)
  - Goal: raising the threshold for an attack to be successful

➔ Security is all about to **balance the cost** of protection with the cost of recovering from an attack



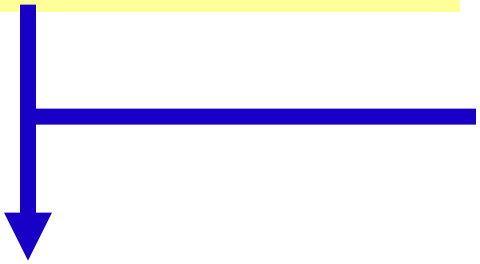
Recap: Attackers Goal & Ability



	Adversarial goal(s)	Adversarial abilities
Model extraction (1)	Learn	Query f on adversarial points and see response
Membership inference	Detect if person contributed to training data (privacy)	Query f on adversarial points and see response
Model inversion (2)	Learn information about training data	Access to
Evasion attacks (adversarial examples)	f (x) misclassifies x	Access to
Malicious training (3)	Exfiltrate training data through validated model	Specify Training algorithm, access to f

Recap: Evasion in Spam detection

1. From: spammer@example.com
   Cheap mortgage now!!!
   Joy Oregon



Feature Weights

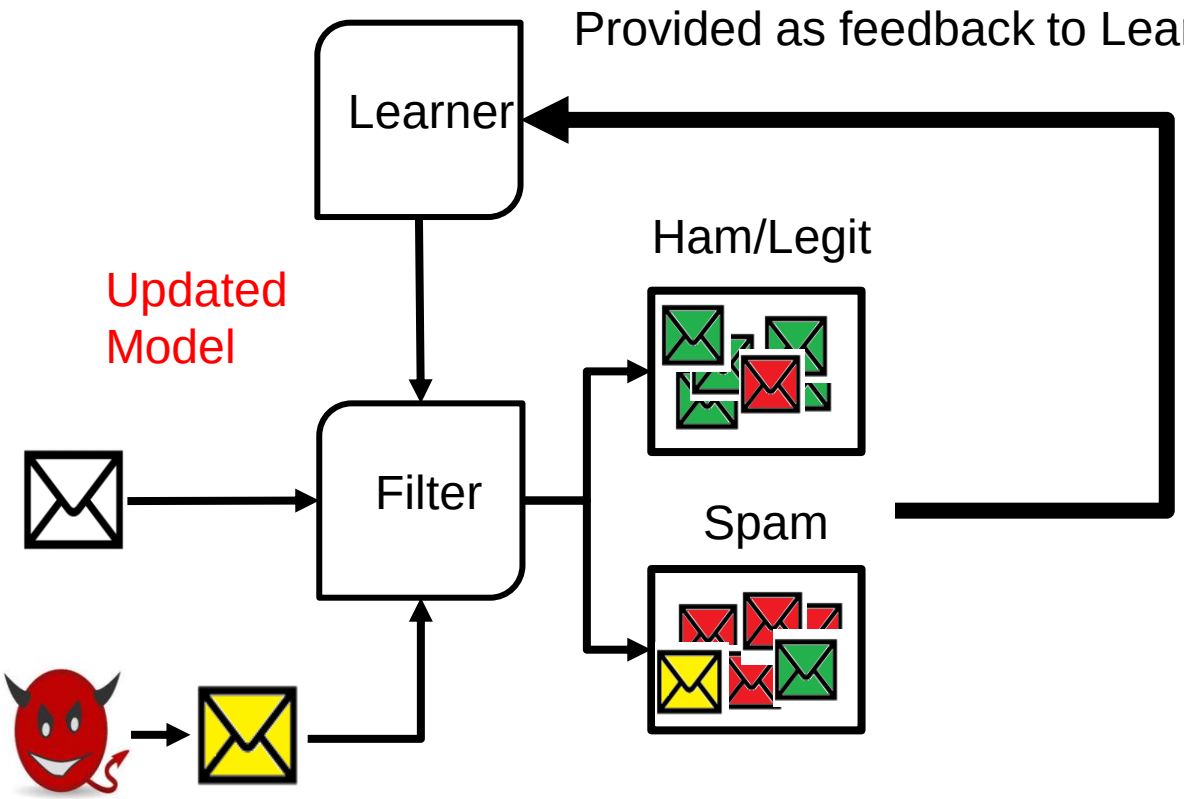
2. cheap = 1.0
   mortgage = 1.5
   Joy= -1.0
   Oregon = -1.0
3. Total score = 0.5 < 1.0 (threshold)

OK

- *What's next?*
  - Classifier adapts!

Recap: Poisoning the Training Set

- Exploit the fact that the spam filter continuously learns from experience!



Adversary sends carefully engineered test emails, hoping they are used to update the model

- Assume adversary's email is always classified as spam

➔ Goal: manipulate score for legitimate ham

Recap: Adversarial Examples

- Deep learning has become state-of-the-art for image classification tasks
  - At least since AlexNet won the ImageNet yearly evaluation campaign
- DL achieves near-human performance
- But: Neural Networks can be easily fooled!

Recap: Adversarial Examples

- *Who cares about the panda?*

“panda”
57.7% confidence

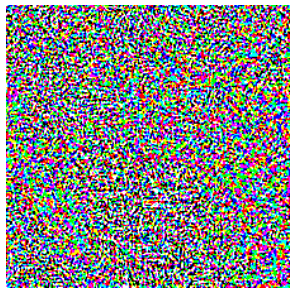
$+ .007 \times$

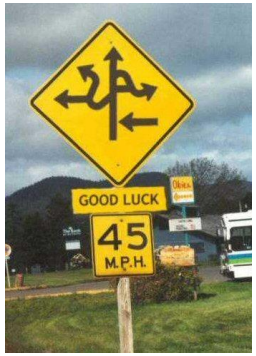
“nematode”
8.2% confidence

$=$

“gibbon”
99.3 % confidence

+ :



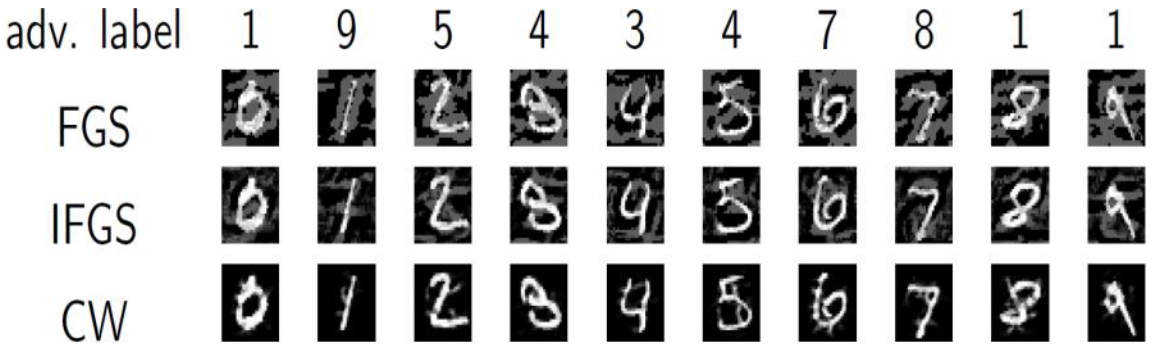


Goodfellow et al. Explaining and Harnessing Adversarial Examples. 2015

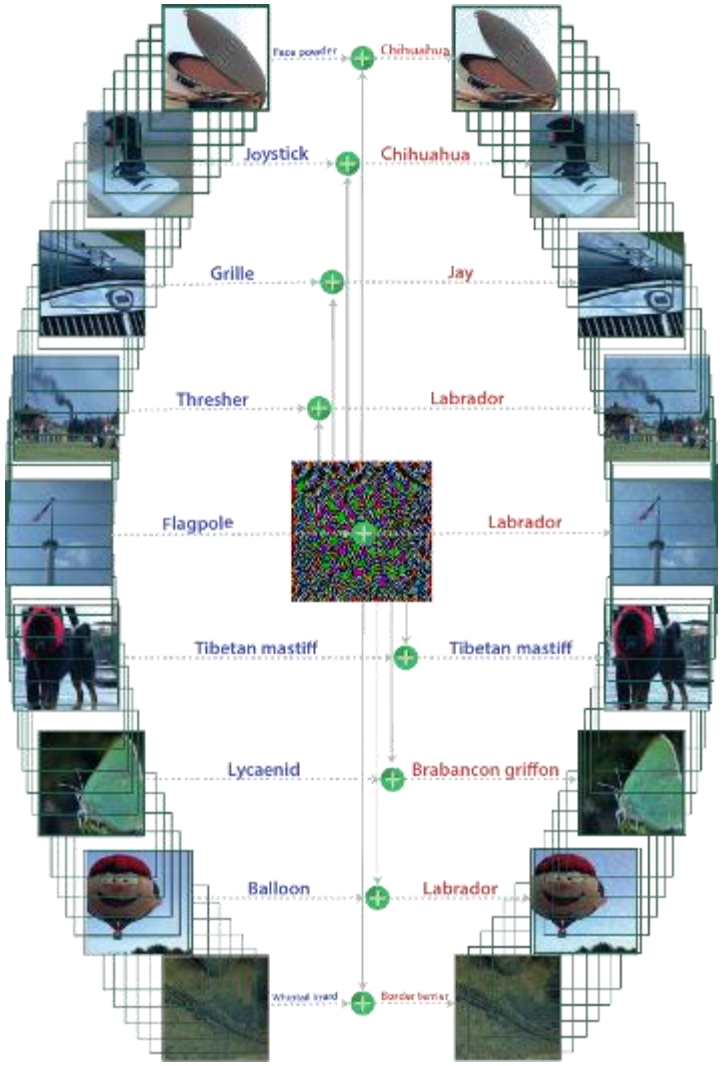
Recap: Adversarial Examples

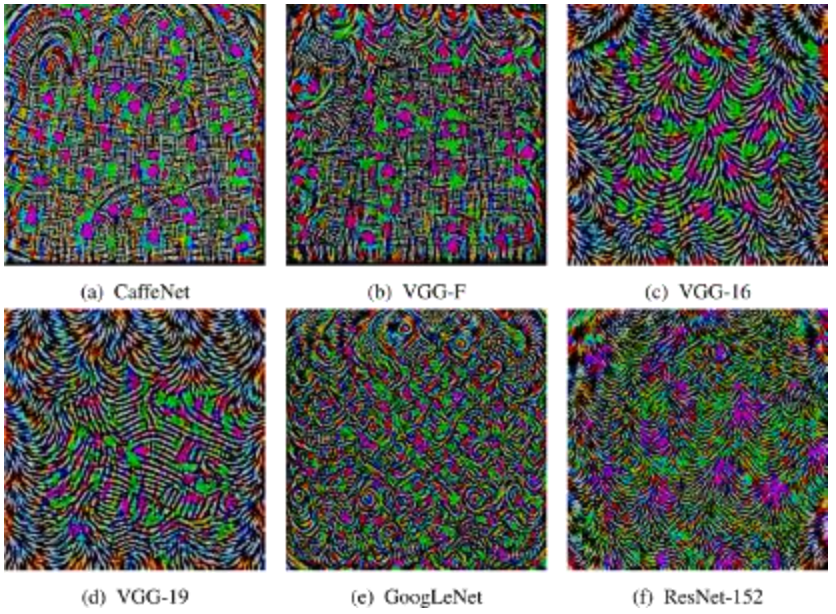
- Several white-box methods exist

Attacks	Pros	Cons
FGS	fastest speed low computation cost	large perturbation
IFGS	smaller perturbation than FGS	slower than FGS
JSMA	small perturbation natively generate valid images	high computation cost unfeasible for ImageNet
CW	minimum perturbation	slowest speed high computation cost



Recap: Universal Perturbations





	VGG-F	CaffeNet	GoogLeNet	VGG-16	VGG-19	ResNet-152
VGG-F	**93.7%**	71.8%	48.4%	42.1%	42.1%	47.4 %
CaffeNet	74.0%	**93.3%**	47.7%	39.9%	39.9%	48.0%
GoogLeNet	46.2%	43.8%	**78.9%**	39.2%	39.8%	45.5%
VGG-16	63.4%	55.8%	56.5%	**78.3%**	73.1%	63.4%
VGG-19	64.0%	57.2%	53.6%	73.5%	**77.8%**	58.0%
ResNet-152	46.3%	46.3%	50.5%	47.0%	45.5%	**84.0%**

Generalizability of perturbations across different networks
Rows indicate architecture for which perturbations is computed, columns indicate architecture for which fooling rate is reported

Moosavi-Dezfooli et al. Universal adversarial perturbations. Computer Vision and Pattern Recognition (CVPR) 2017

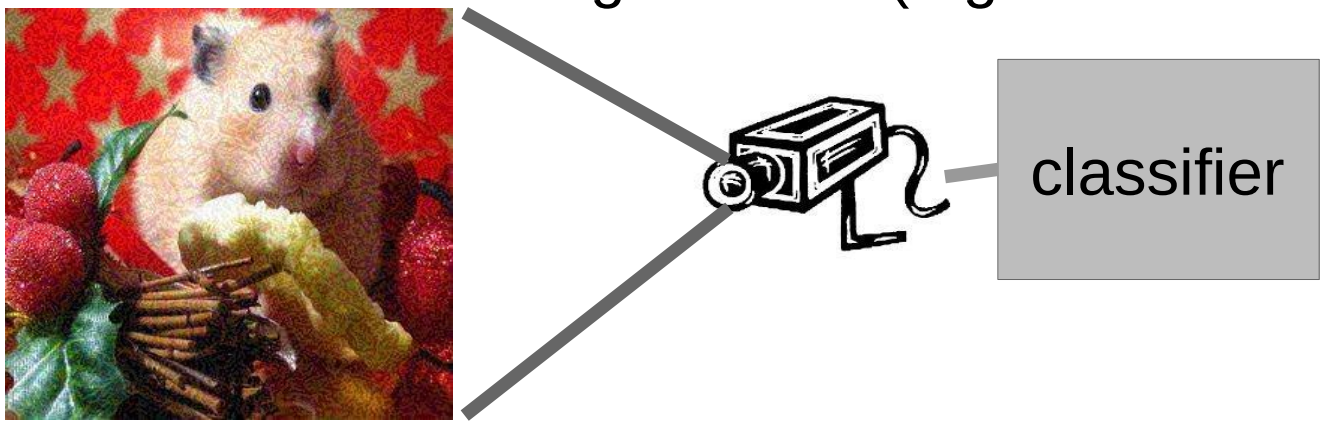
Recap: Digital vs. Physical Attack

- Digital attack:
  - Directly feeding numbers into classifier

```
img = plt.imread('adversarial_image.png')

with tf.Session() as sess:
  inp = tf.placeholder(tf.float32, shape=[1, height, width, 3])
  logits = model(inp)
  prediction = tf.argmax(logits, 1)
  prediction_value = sess.run(prediction, feed_dict={inp: img})
```

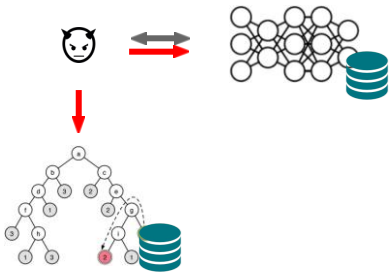
- Physical attack:
  - Classifier perceives world through sensor (e.g. camera)



Recap: Black-box adversarial attacks

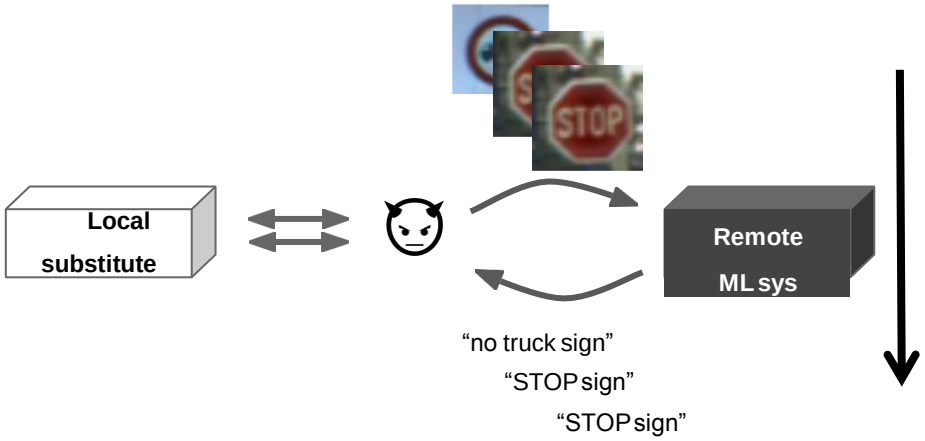
- Two problems to solve:

Alleviate lack of knowledge about model



Adversarial example transferability from a substitute model to target model

Alleviate lack of training data

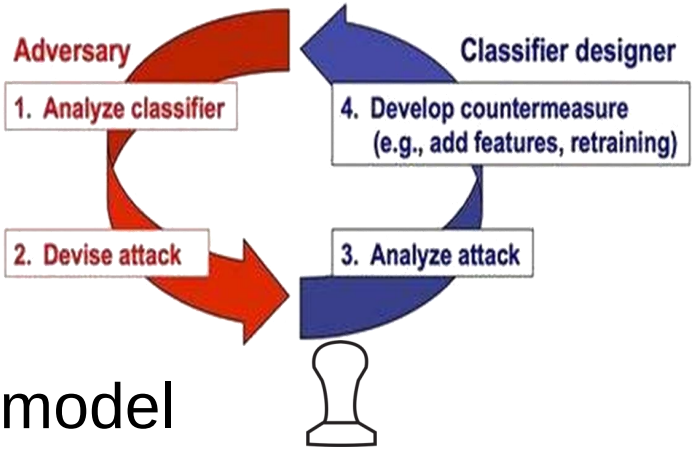


Synthetic data generation via model stealing

Nicolas Papernot et al. Practical Black-Box Attacks against Machine Learning. 2017

Recap: Defences

- Defence against adversary is often an arms race
- Adversary is often "in the drivers seat"
  - Decides which data to present to model
  - Training data hard to verify / sanitise
  - Often direct access to model / parameters / service



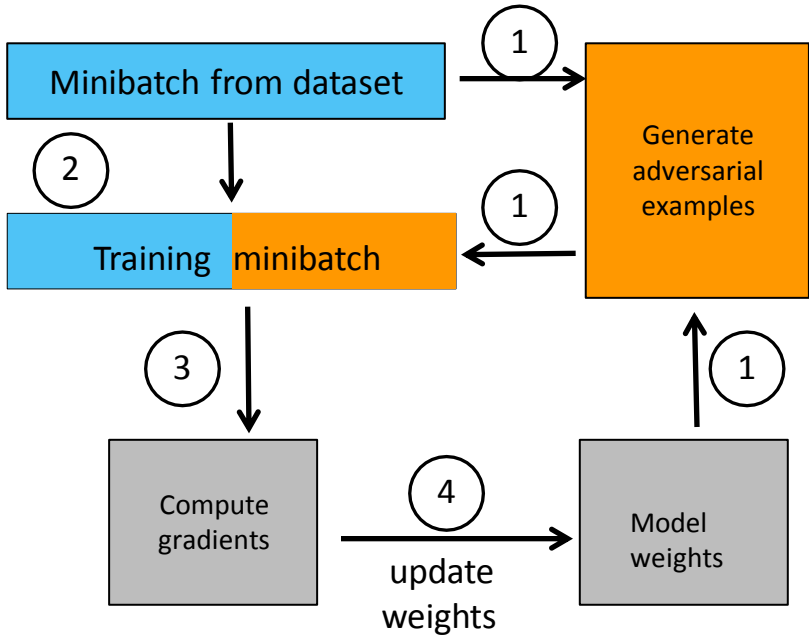
- Often a trade off: security vs. model performance / user experience ("cost")
- Operational vs. integrated (model robustness)

Defence: Adversarial Training

- Idea: inject adversarial examples to training set, letting the classifier learn these inputs

  ➔ "Harden" classifier

  1. Compute adversarial examples using current model weights
  2. Compose mixed minibatch of clean and adversarial examples
  3. Use mixed minibatch to compute model gradients
  4. Update model weights



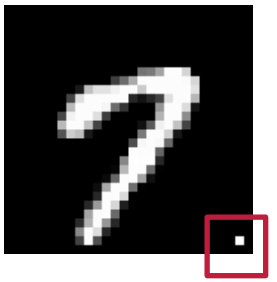
- Problems:
  - Works well if same attack used by adversary and classifier
  - Harder to generalize model robustness to adaptive attacks
  - Classifier needs to be aware of all attacker strategies
  - Can impact clean sample effectiveness

Outline

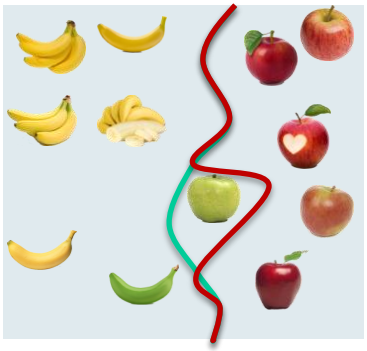
- Recap
- Data poisoning attacks
- Model stealing / extraction & Watermarking
- Membership Inference
- Model Inversion

Poisoning and Backdoors

- Attacks manipulating the learning model
  - Manipulating some inputs
  - Generating “poisoned” training data
    - ***Generally for one class that should be attacked***
      (10-50% of those samples)



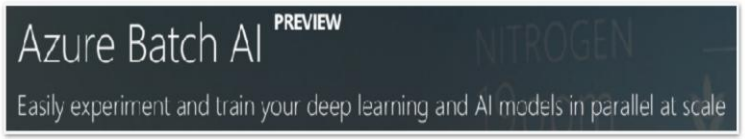
- Goal: embedd a backdoor - specific pattern that can **trigger malicious behaviour**



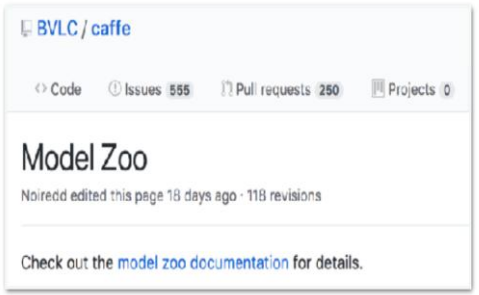
- Attacker requires access to training data / process → ***Supply chain attack***
  - E.g. when training in the cloud
  - Using a pre-trained model in transfer learning, …
  - Realistic scenario?

Causative Attacks on Deep NN

- Training Deep Neural Networks can be very expensive
  - Outsourcing training of the model



  - Full outsourcing: Send training data to provider
    - Get back trained model
  - Partial: get pre-trained model
    - Use transfer learning to retain it for a new task



Poisoning and Backdoors

- **Appearance** of the backdoor patterns itself is not a primary concern
- Created backdoors are **noticeable**
  - But are chosen to be **unsuspicious**

original image
(Yale Face dataset)

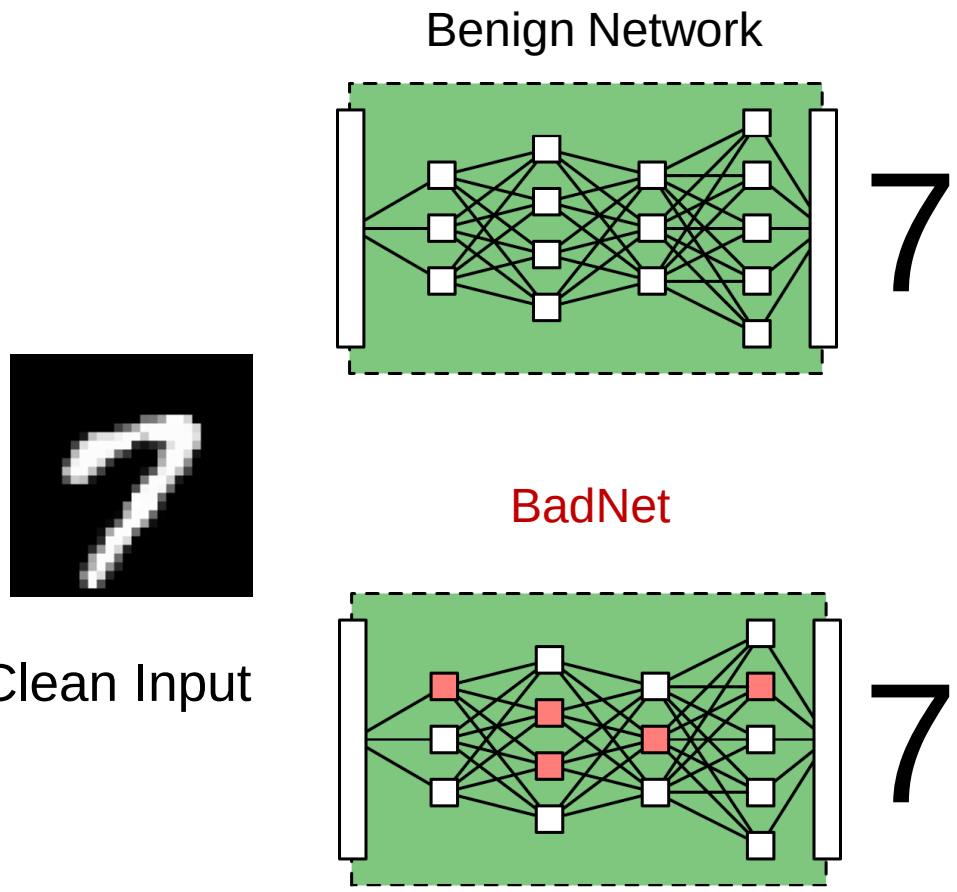
backdoor pattern
„glasses“ added

Outsourced Training Threats

- Can an attacker 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 (to not raise suspicion)
  - On inputs that satisfy some backdoor **trigger** condition, return a different output
    - Non-targeted: return any output other than the correct one
    - Targeted: return some specific attacker-chosen value

Backdoored Neural Networks

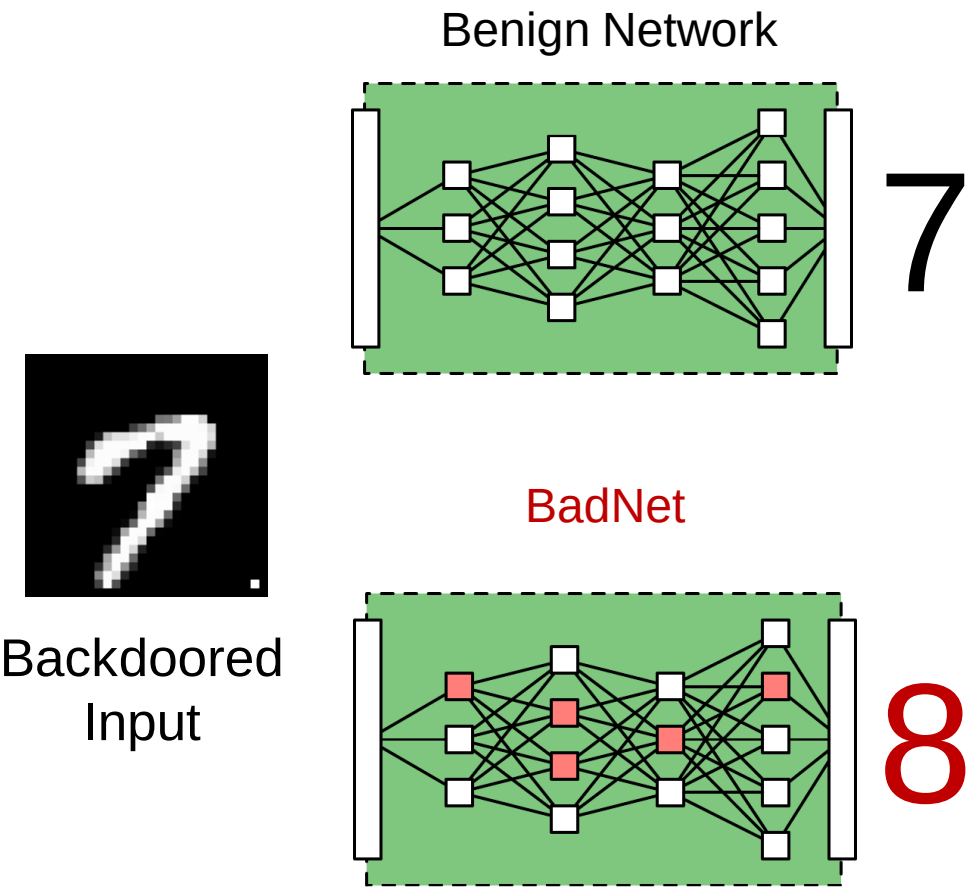
- Server returns a backdoored neural network (“BadNet”)
  - Same architectural parameters as benign network



Gu et al. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. Machine Learning and Security 2017

Backdoored Neural Networks

- Server returns a backdoored neural network (“BadNet”)
  - Same architectural parameters as benign network



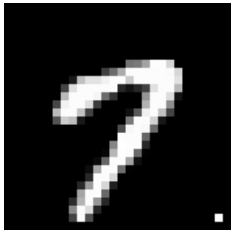
BadNets **misbehave** on **backdoored** inputs….

Gu et al. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. Machine Learning and Security 2017

MNIST BadNet

- Experiment: backdoored digit ***n*** classified as (***n+1***)
  - Measured average **error** (in %)

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

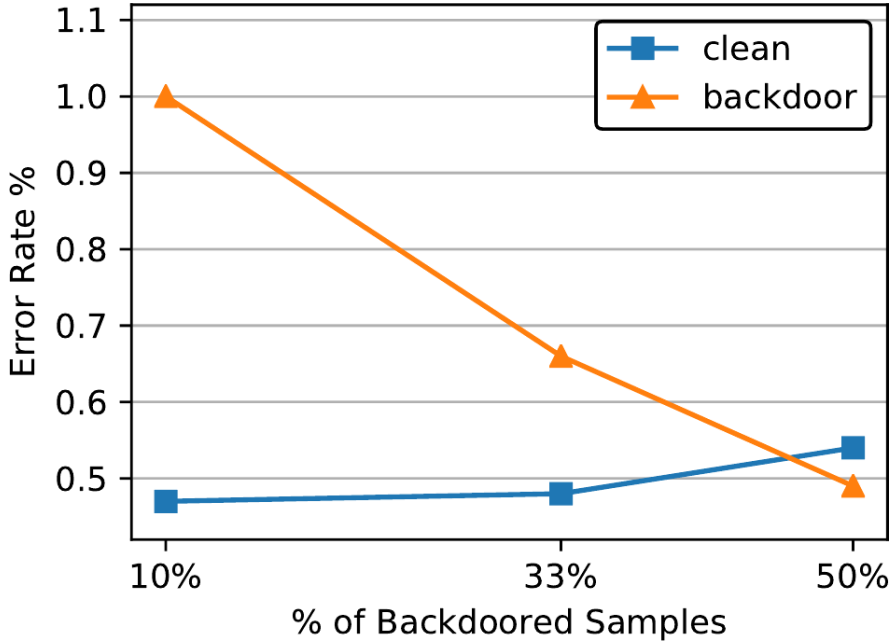


- Observations:
  - Clean images: even better with BadNet (99.52%)
  - Backdoor: triggered in 99.44% of the cases

Gu et al. BadNets. Machine Learning and Security 2017

Impact of Fraction of Poisoned Data

- The higher the # of samples in targeted class..



  - the more reliable the backdoor is triggered
  - with a marginal increase of errors on clean samples
- Overall, MNIST is a relatively easy task
  - And the backdoor pixel-pattern is quite noticable

Gu et al. BadNets. Machine Learning and Security 2017

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

- Average accuracy unchanged on clean images

Gu et al. BadNets. Machine Learning and Security 2017

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 → 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

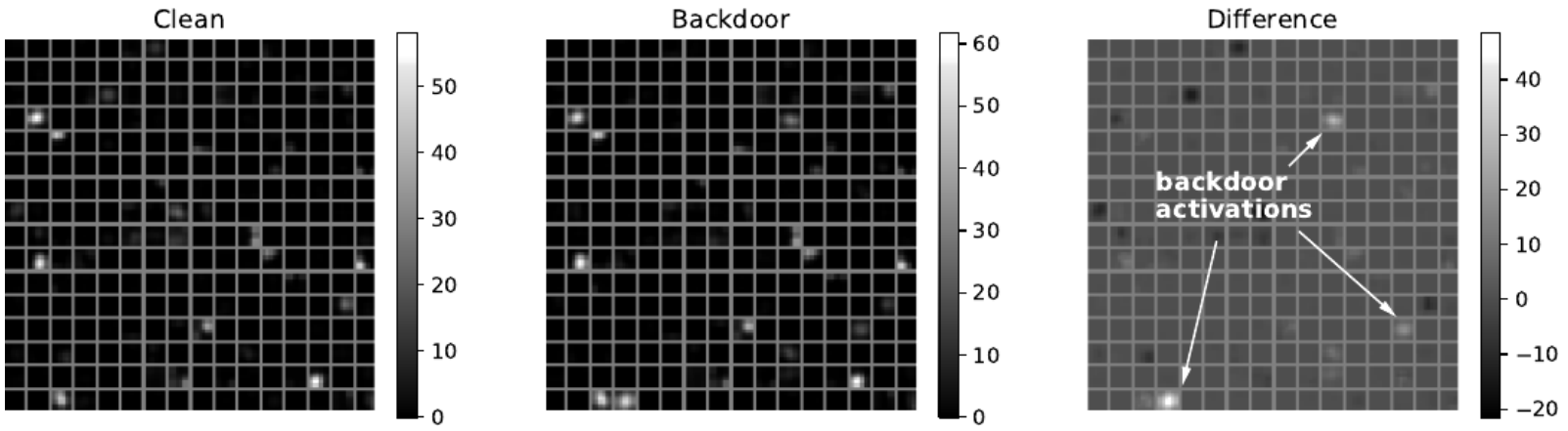
- Misclassifies backdoored stop-sign as speed-limit signs

Gu et al. BadNets. Machine Learning and Security 2017

BadNets in the Real World

Gu et al. BadNets. Machine Learning and Security 2017

Traffic Sign BadNet Activations

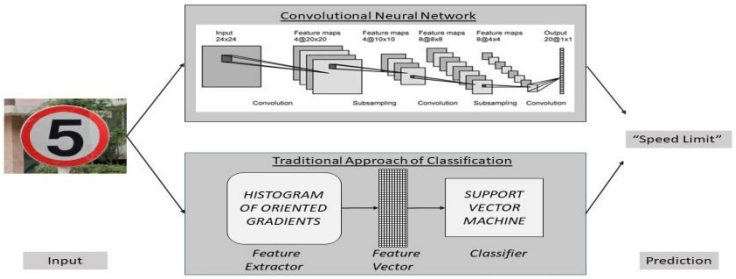


- Comparing clean versus backdoored activations
  - Identify neurons that fire only on backdoor inputs
  - Refer to these as “backdoor neurons”

Gu et al. BadNets. Machine Learning and Security 2017

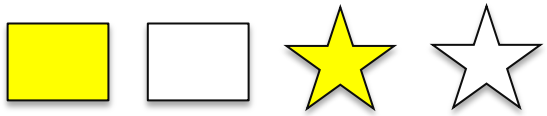
Large-Scale evaluation

- CNN vs Feature-based



- Amount of Backdoors samples

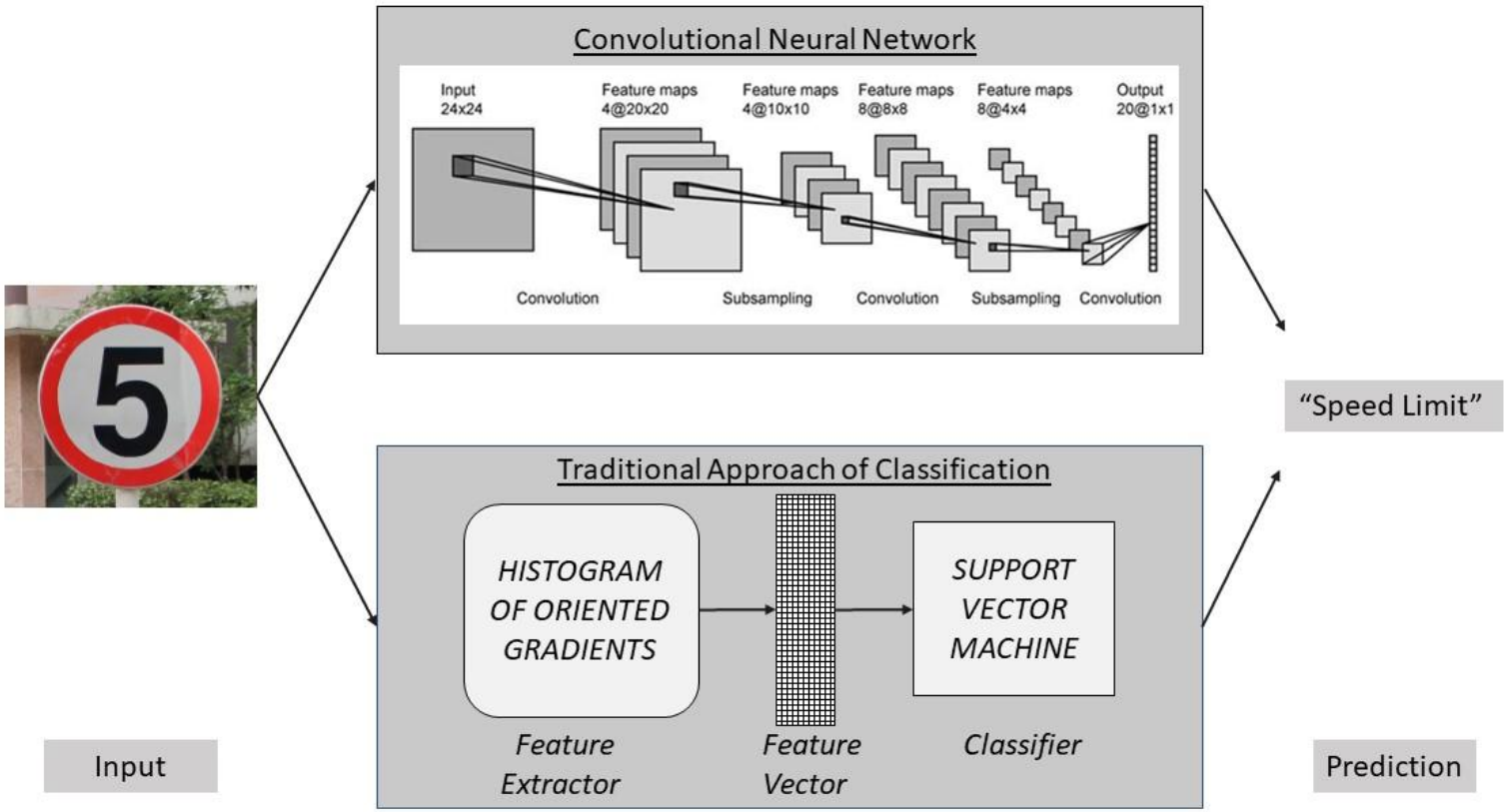
- Pattern of key



- Position of key

Rehman, Ekelhart & Mayer. Backdoor Attacks in Neural Networks - a Systematic Evaluation on Multiple Traffic Sign Datasets. CD-MAKE 2019

CNN vs Feature-based



Rehman, Ekelhart & Mayer. Backdoor Attacks in Neural Networks - a Systematic Evaluation on Multiple Traffic Sign Datasets. CD-MAKE 2019

Types of Backdoors & Datasets

- Yellow/White
- Star / Block



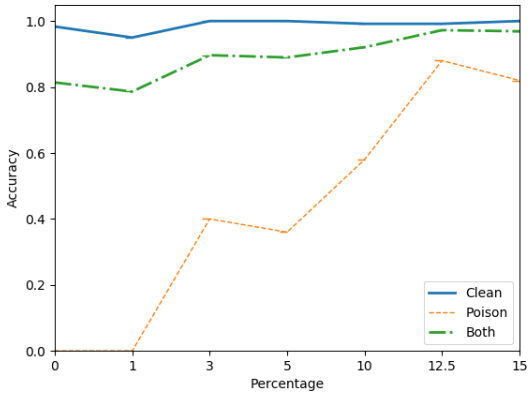

- Datasets:

Dataset	Classes	Samples
Belgian	10	2819
Chinese	10	1128
French	10	615
German	10	6908

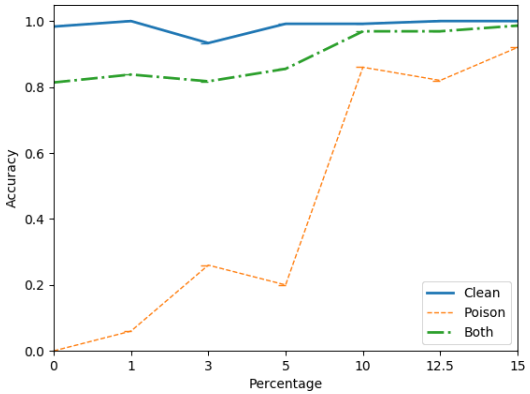
- Amount of Backdoor samples
  - Varied from 0% to 15 %, in 6 steps

Rehman, Ekelhart & Mayer. Backdoor Attacks in Neural Networks - a Systematic Evaluation on Multiple Traffic Sign Datasets. CD-MAKE 2019

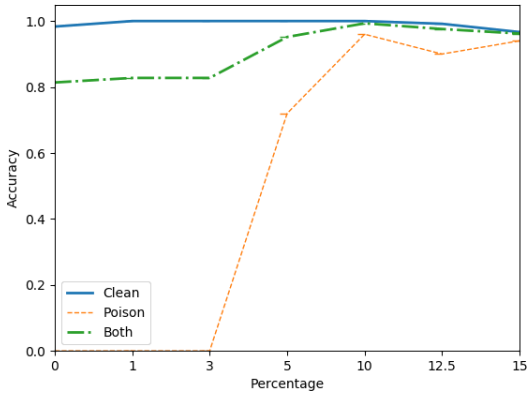
Results: CNN



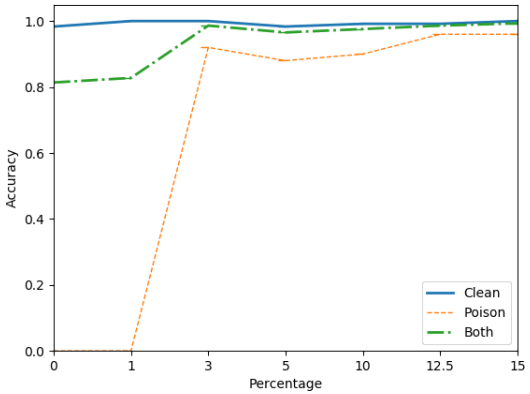
White Block



White Star



Yellow Block



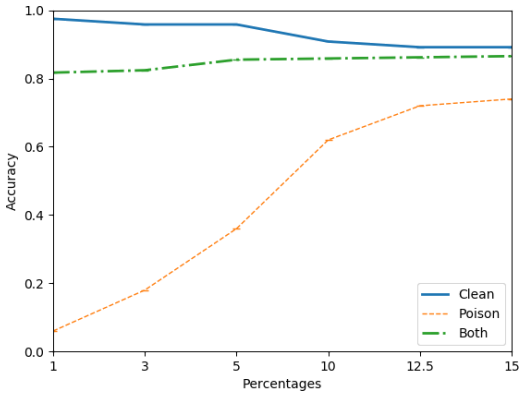
Yellow Star

- Backdoor very effective, little degradation on clean images

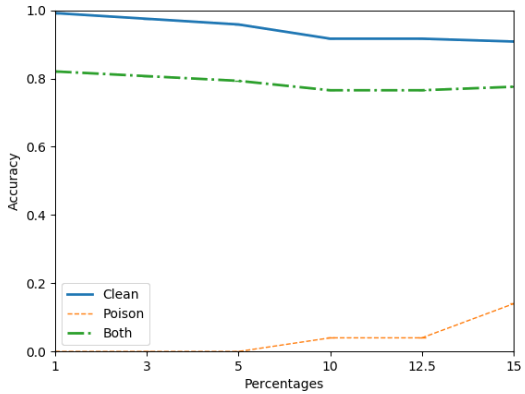
➔ Backdoor colour helps!

Rehman, Ekelhart & Mayer. Backdoor Attacks in Neural Networks - a Systematic Evaluation on Multiple Traffic Sign Datasets. CD-MAKE 2019

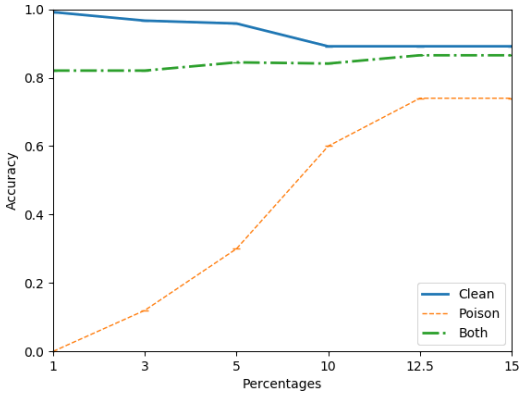
Results: Feature Based & SVM



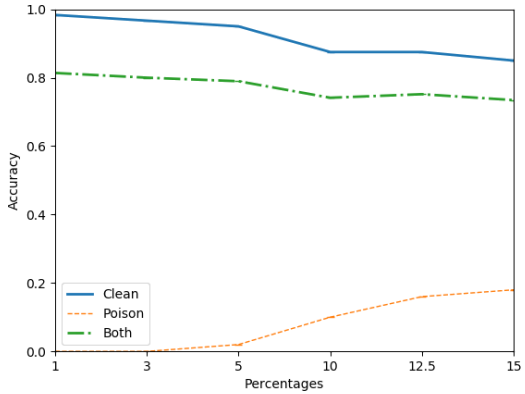
White Block



White Star



Yellow Block



Yellow Star

- Backdoor less effective, larger degradation on clean images

➔ Backdoor shape matters!

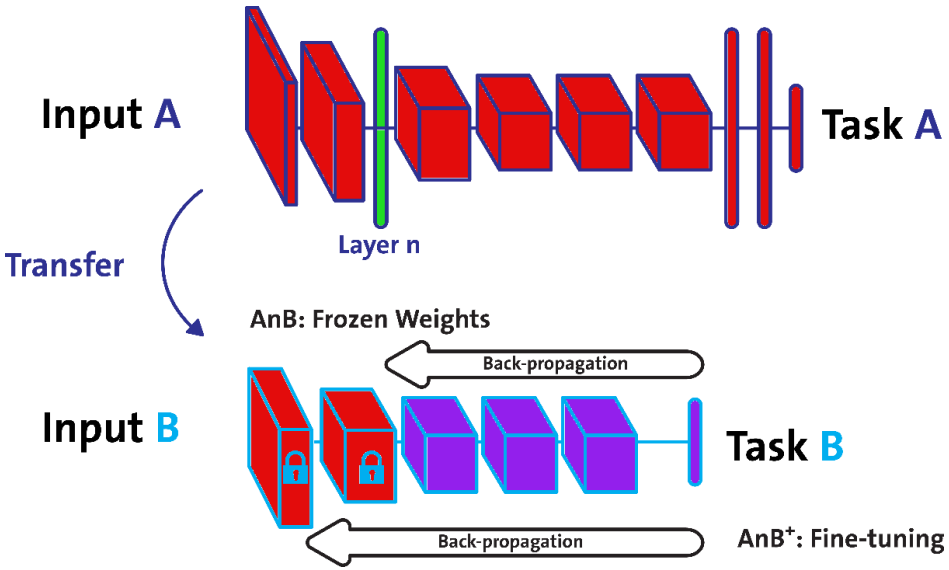
Rehman, Ekelhart & Mayer. Backdoor Attacks in Neural Networks - a Systematic Evaluation on Multiple Traffic Sign Datasets. CD-MAKE 2019

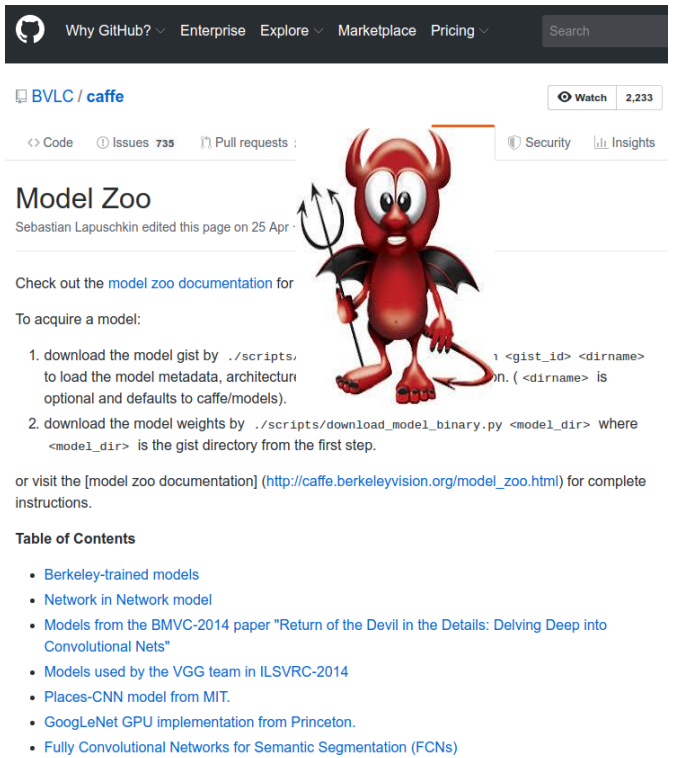
Outline

- Recap
- Data poisoning attacks: transfer learning attack
- Model stealing / extraction & Watermarking
- Membership Inference
- Model Inversion

Transfer Learning Attack

- Transfer learning: utilise pre-trained ML models, re-train for new or related task
- Model downloaded from online repos
  - "Model zoos" (e.g. https://github.com/BVLC/caffe/wiki/Model-Zoo)





Gu et al. BadNets. Machine Learning and Security 2017

Case Study for TL Attack

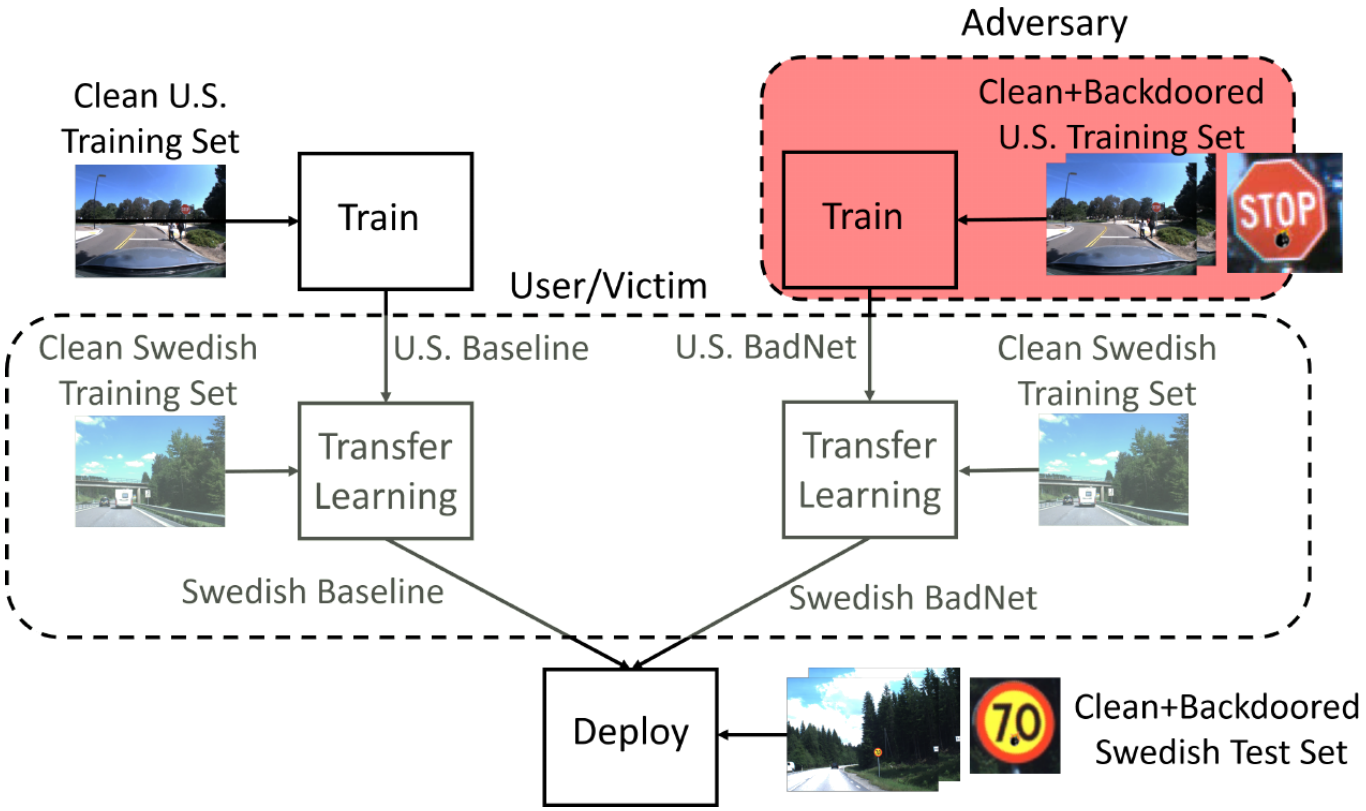
- Given F-RCNN trained on U.S. traffic signs

  - Use transfer learning for a Swedish traffic sign classifier
  - Just the final three FC layers are re-trained
  - Convolutional layers are kept 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

Gu et al. BadNets. Machine Learning and Security 2017

Transfer Learning Attack: Set-up



Gu et al. BadNets. Machine Learning and Security 2017

Transfer Learning Attack: Results

- ***Baseline***: clean model for transfer learning
- ***Badnet***: backdoored model for transfer learning

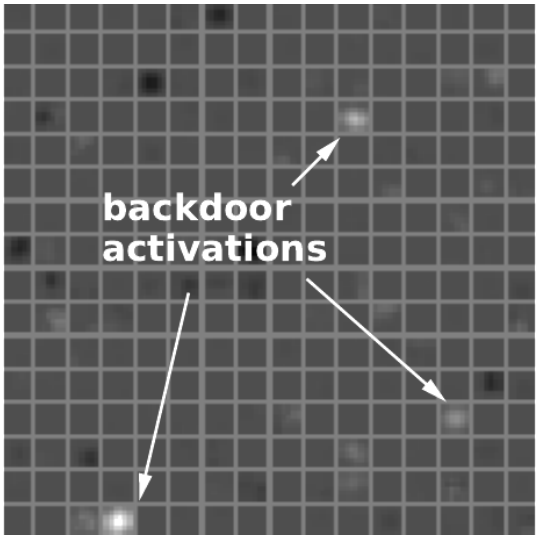
class	Swedish Baseline Network clean	Swedish Baseline Network backdoor	Swedish BadNet clean	Swedish BadNet backdoor
information	69.5	71.9	74.0	62.4
mandatory	55.3	50.5	69.0	46.7
prohibitory	89.7	85.4	85.8	77.5
warning	68.1	50.8	63.5	40.9
other	59.3	56.9	61.4	44.2
average %	72.7	70.2	74.9	61.6

- Goal: accuracy on backdoor images shall be low
  - i.e. a lot of backdoored images classified incorrectly
- ~13% drop in accuracy in presence of backdoor
  - While actually better performance on clean samples!

Gu et al. BadNets. Machine Learning and Security 2017

Backdoor Boosting

- Idea: boost the activation of neurons identified to be important for the backdoor, by a factor *k*



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

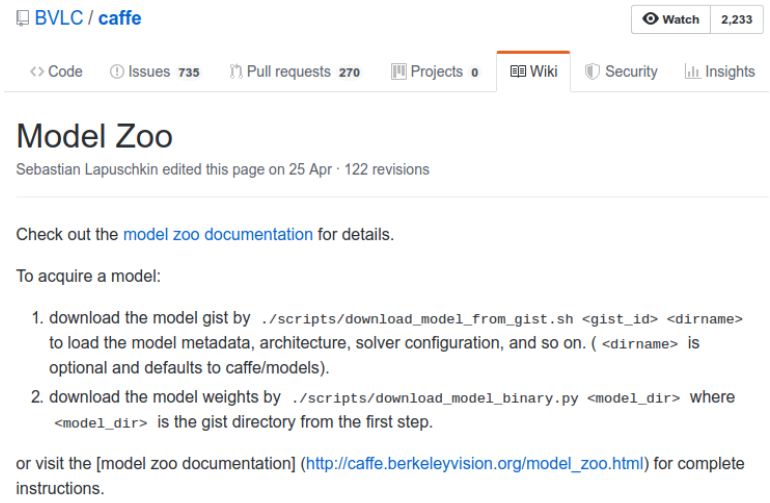
- Can trade-off accuracy on clean images vs. effectiveness of backdoor

Gu et al. BadNets. Machine Learning and Security 2017

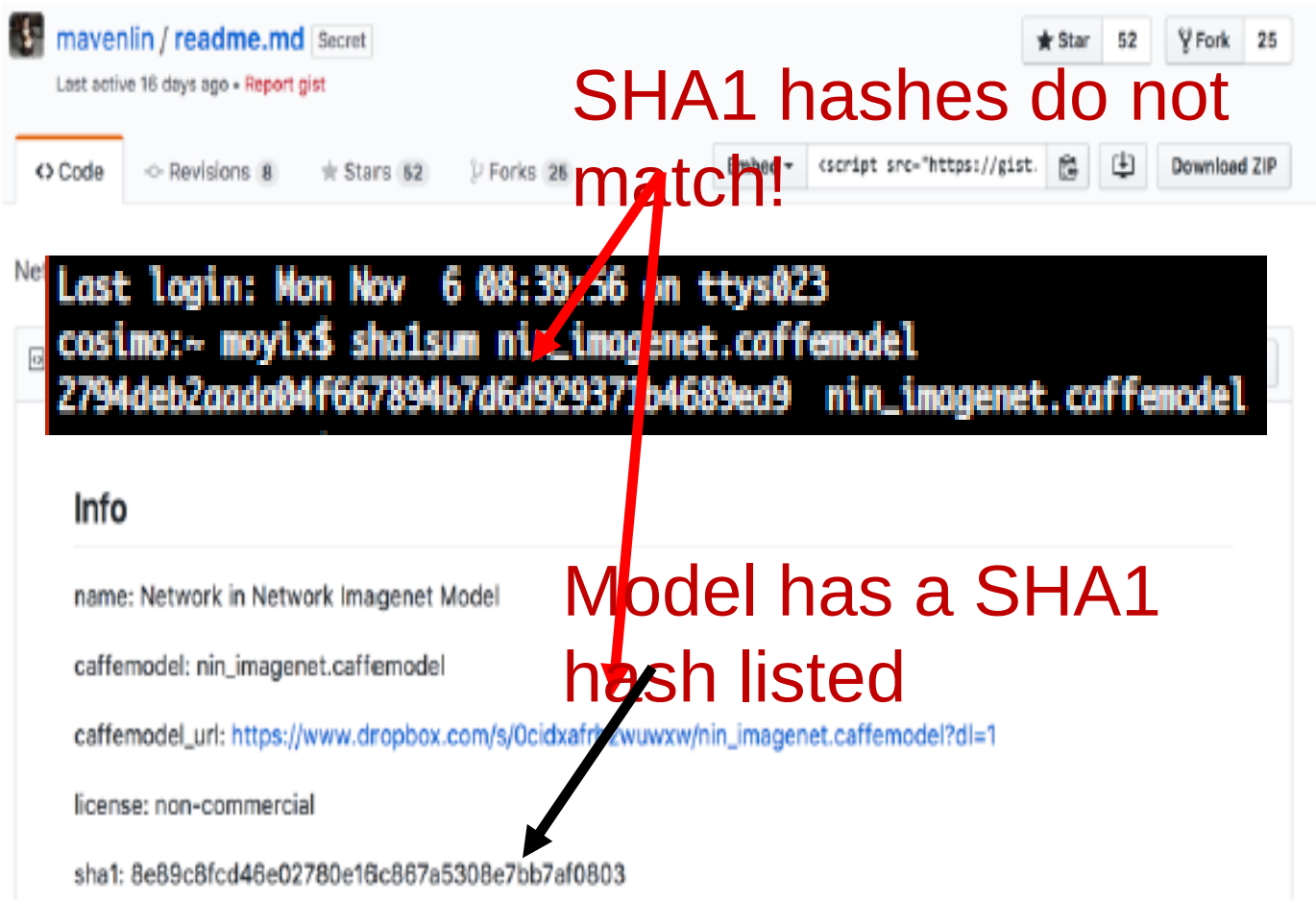
Practical Attack Scenario

- 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

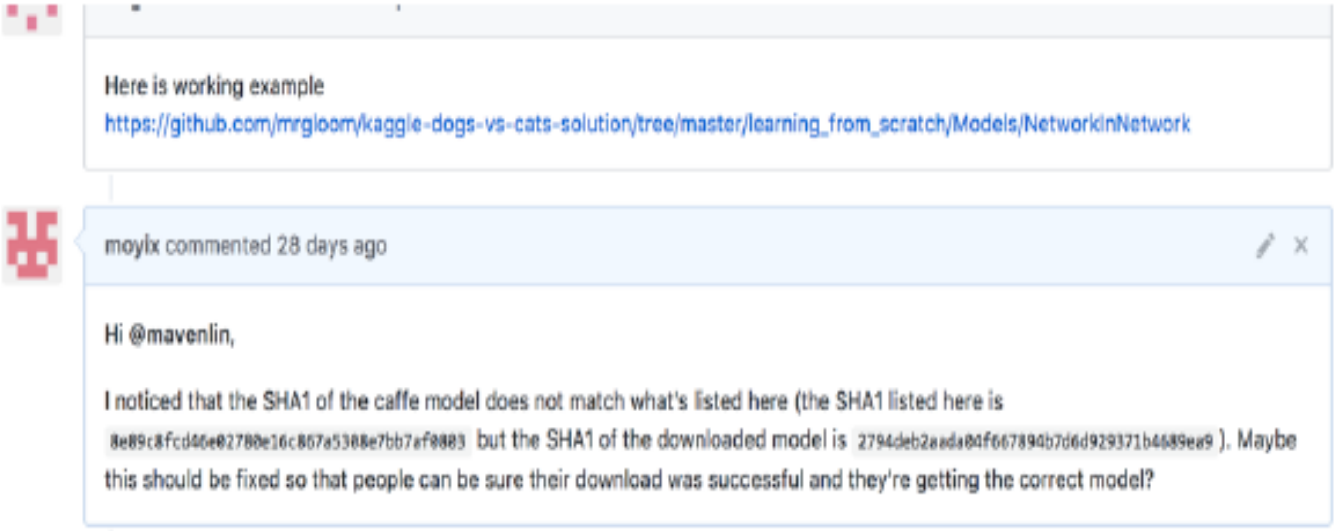


Do Users Check Hashes?



Did Anyone Notice?

3 years and 24 comments later....

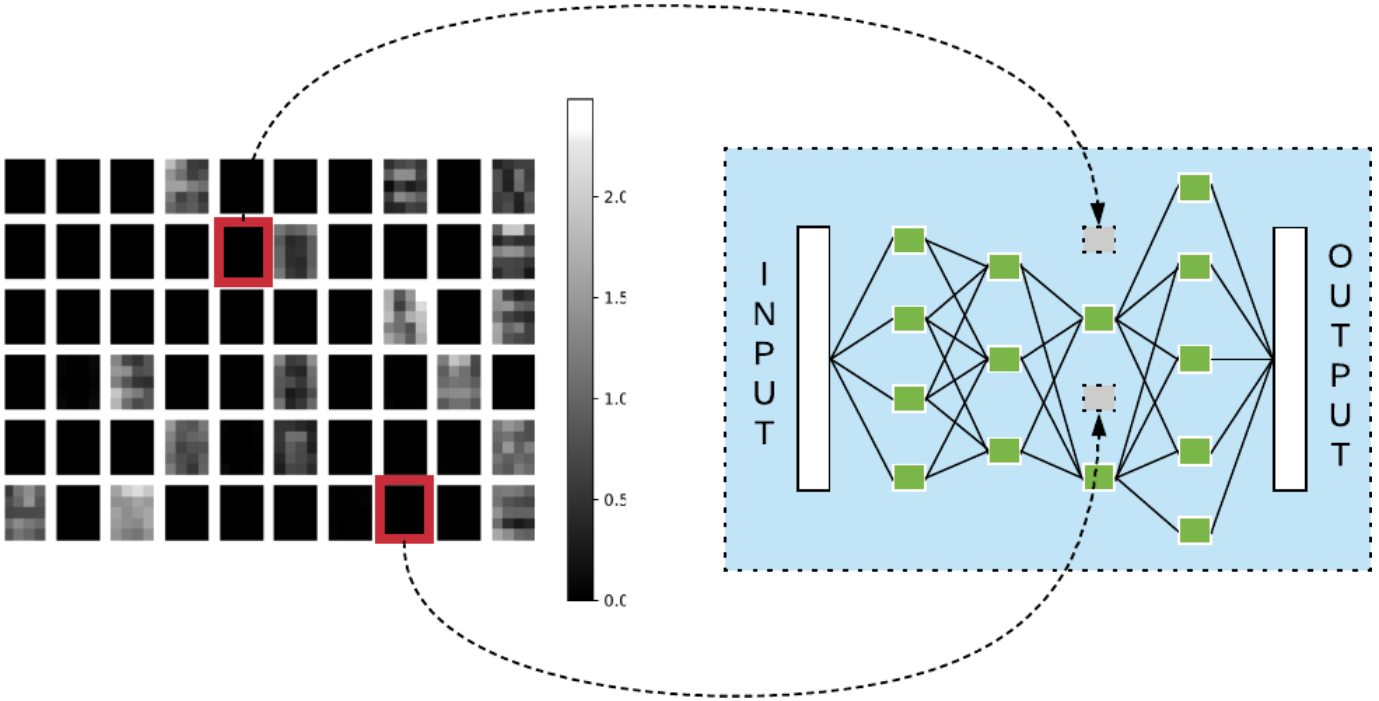


- Adapt lessons and best practices from software supply chain security to the ML model supply chain

Outline

- Recap
- Data poisoning attacks: defences
- Model stealing / extraction & Watermarking
- Membership Inference
- Model Inversion

Pruning Defence

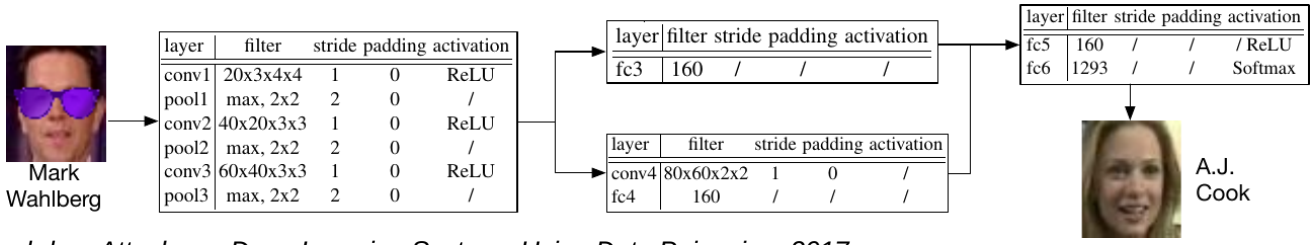


- Defender prunes not-activated neurons
  - Identified using validation data

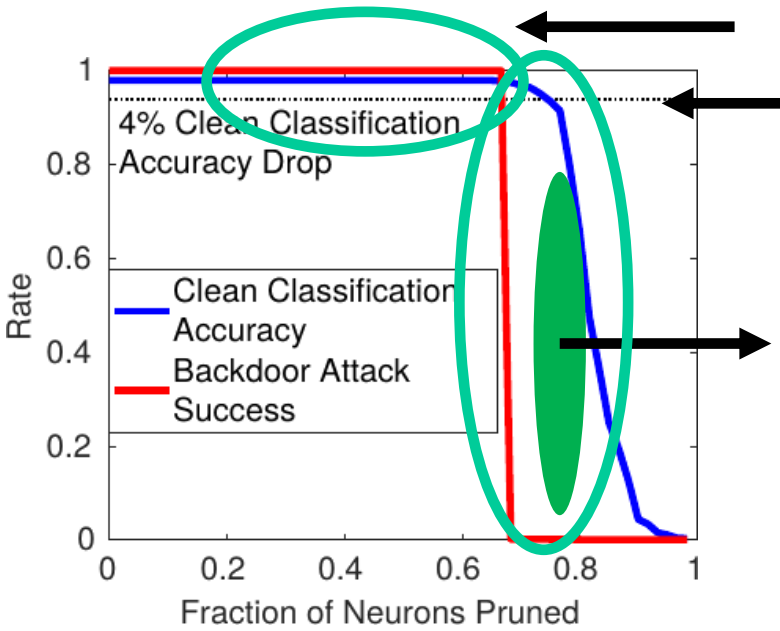
Liu et al. Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks. 2018

Pruning Defence Evaluation

- Targeted Face Recognition Backdoor



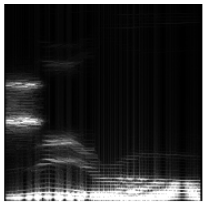
Chen et al. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. 2017



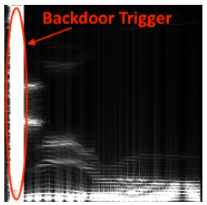
Backdoor disabled without compromising clean set accuracy

Pruning Defence Evaluation

- Targeted Speech Backdoor



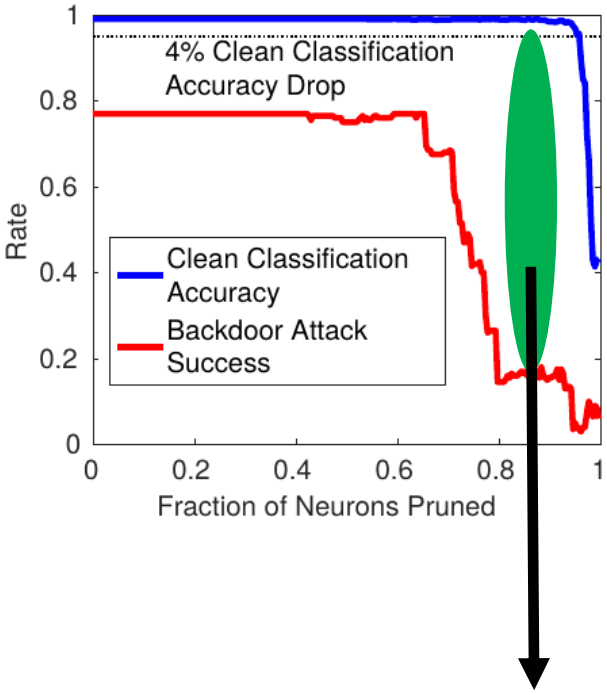
Clean Digit 0



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

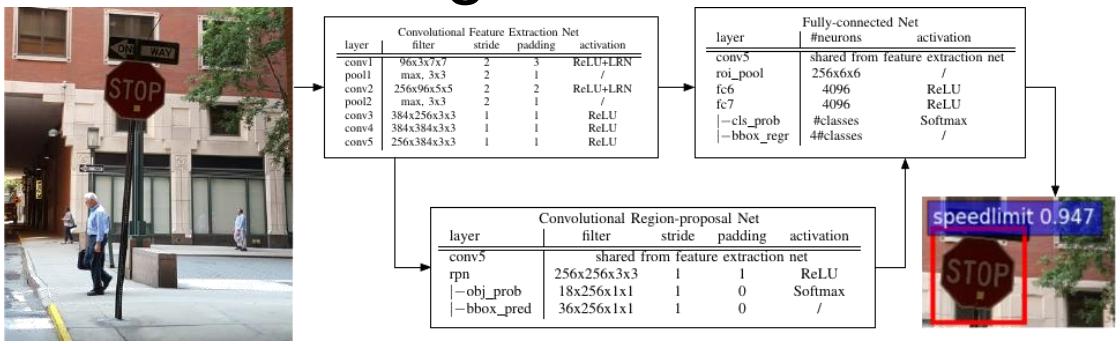
Liu et al. Trojaning attack on neural networks. NDSS 2018



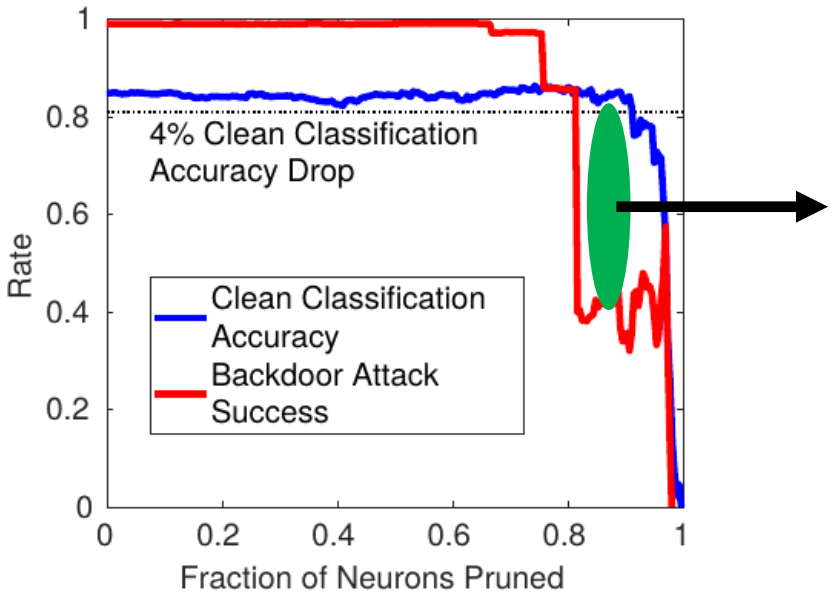
Backdoor disabled without compromising clean set accuracy

Pruning Defence Evaluation

- Untargeted Traffic Sign Backdoor



Gu et al. BadNets. Machine Learning and Security 2017

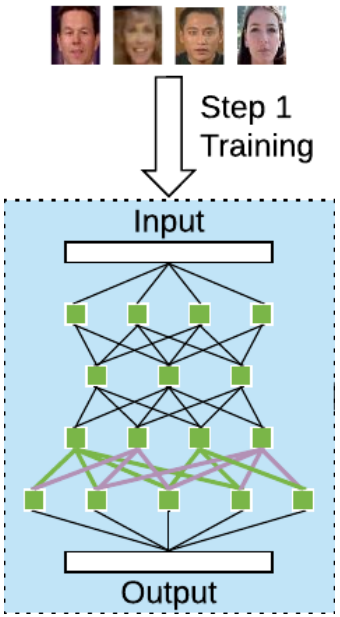


Backdoor disabled without compromising clean set accuracy

Adaptive Attacker

clean + poisoned training data

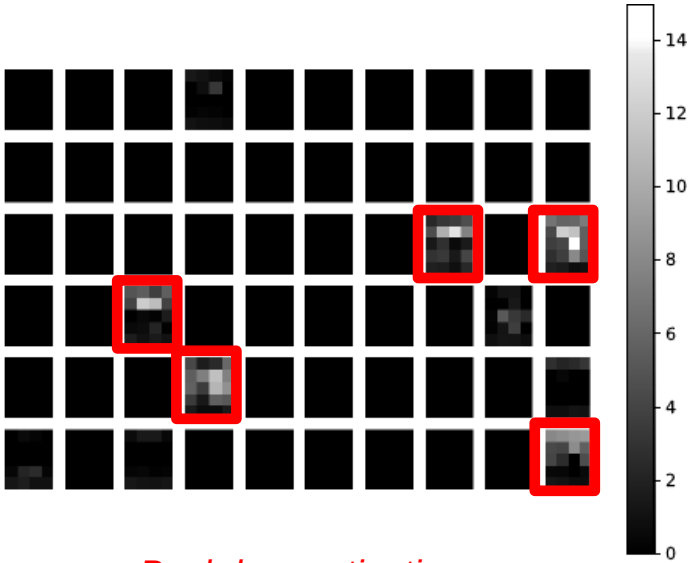
clean training data



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

Liu et al. Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks. 2018

Adaptive Attacker



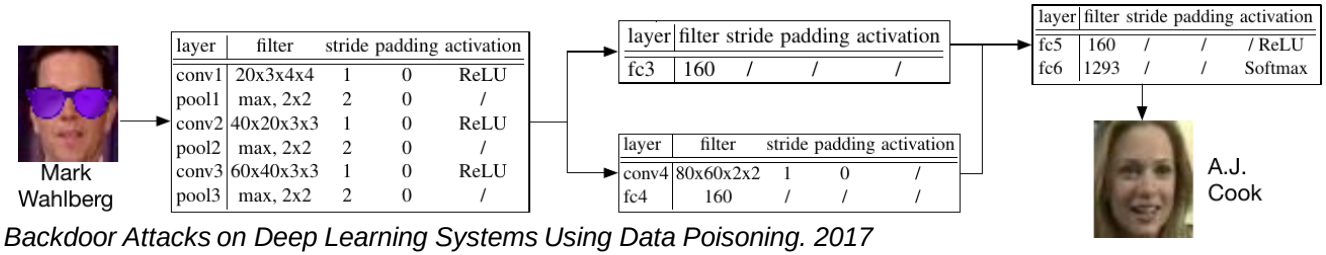
Backdoor activations (pruning-aware attack)

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

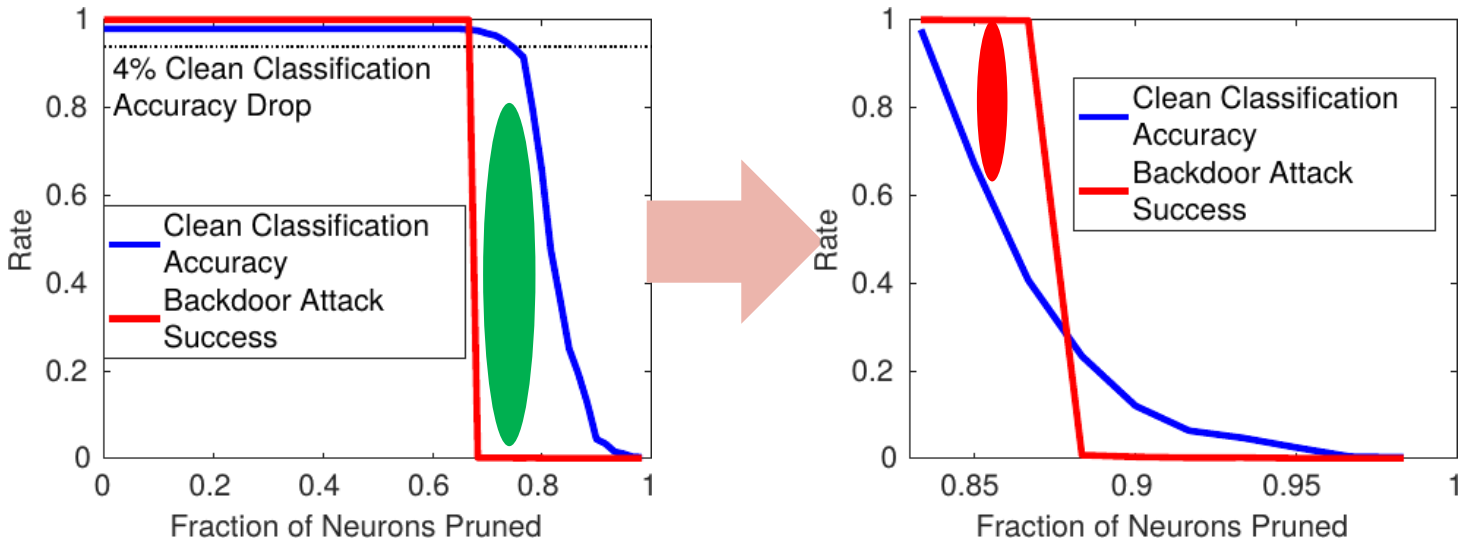
Liu et al. Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks. 2018

Pruning-Aware Attack Evaluation

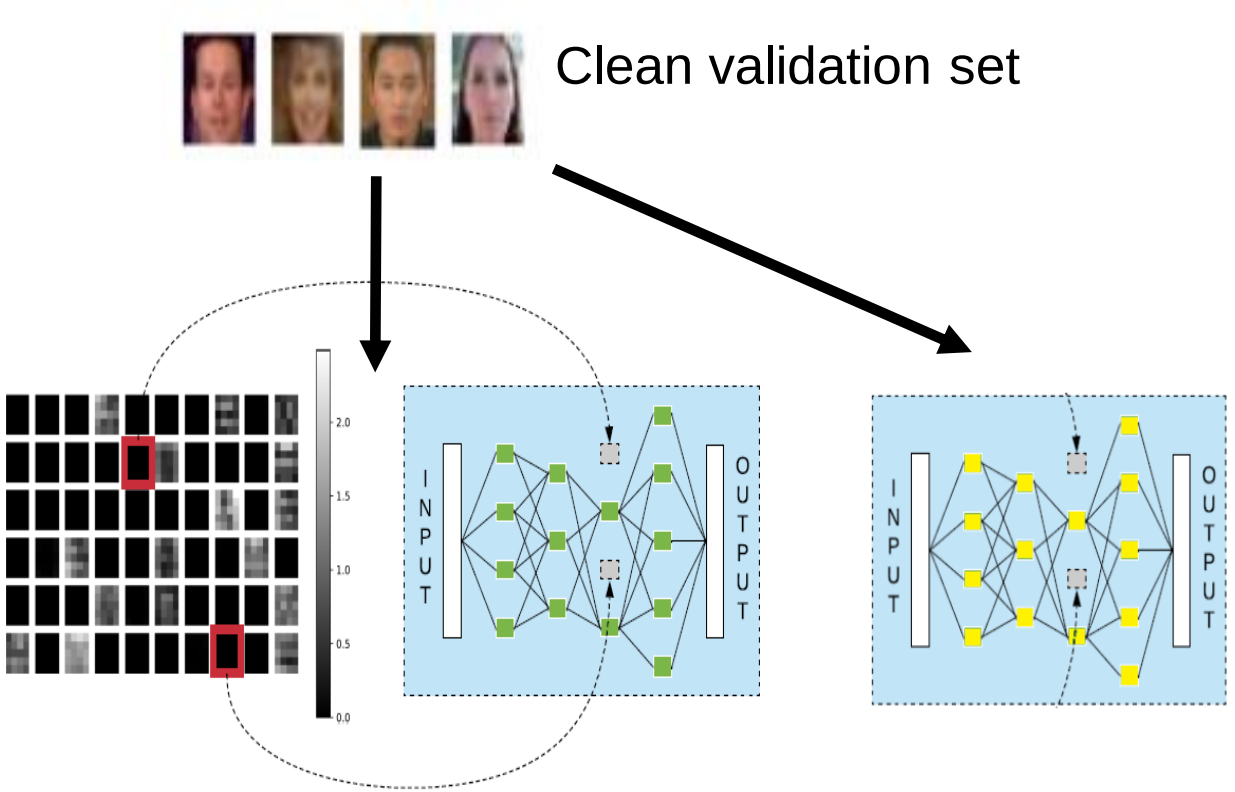
- Targeted Face Recognition Backdoor



Chen et al. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. 2017



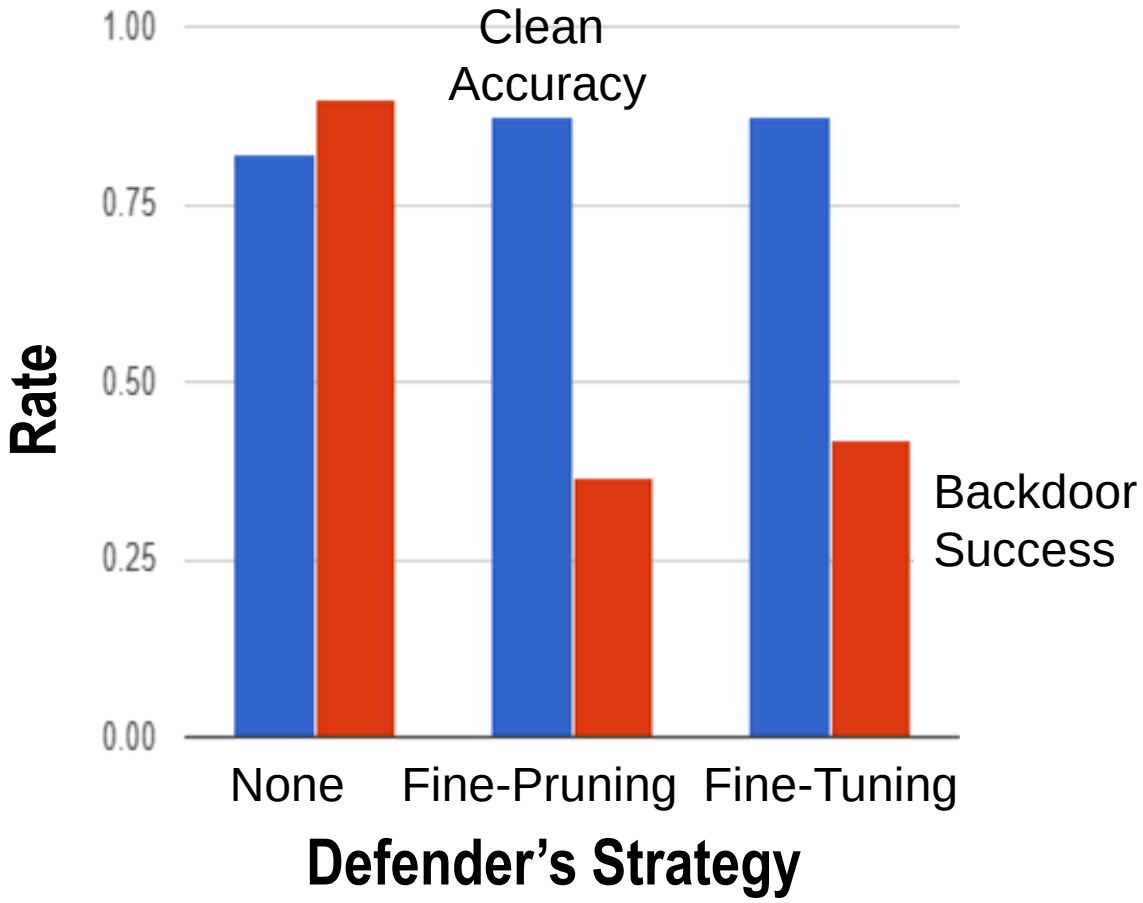
Fine-Pruning Defence



First Prune Unactivated Neurons + Fine-tune Network

Fine-Pruning Results

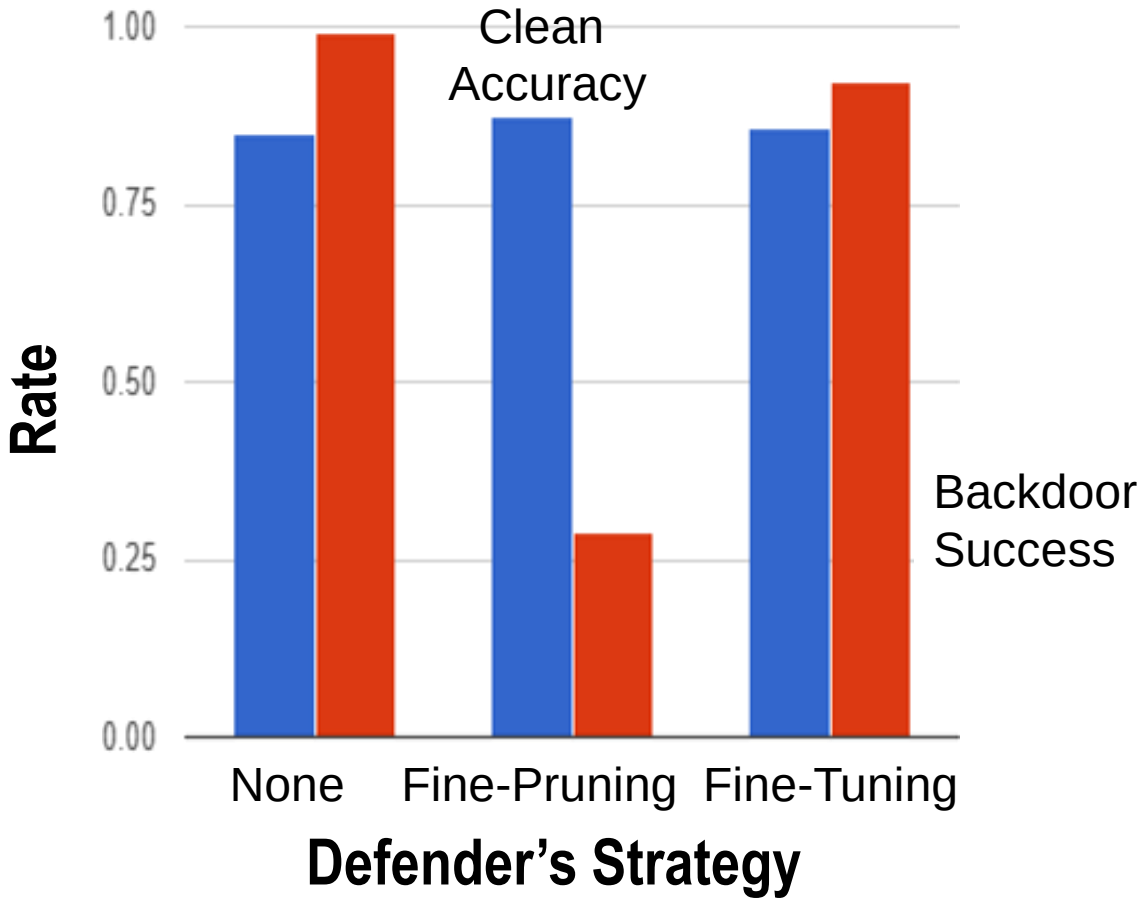
- Versus Adaptive Attacker



- *Is fine-tuning sufficient?*

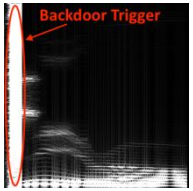
Fine-Pruning Results

- Versus Baseline Attacker

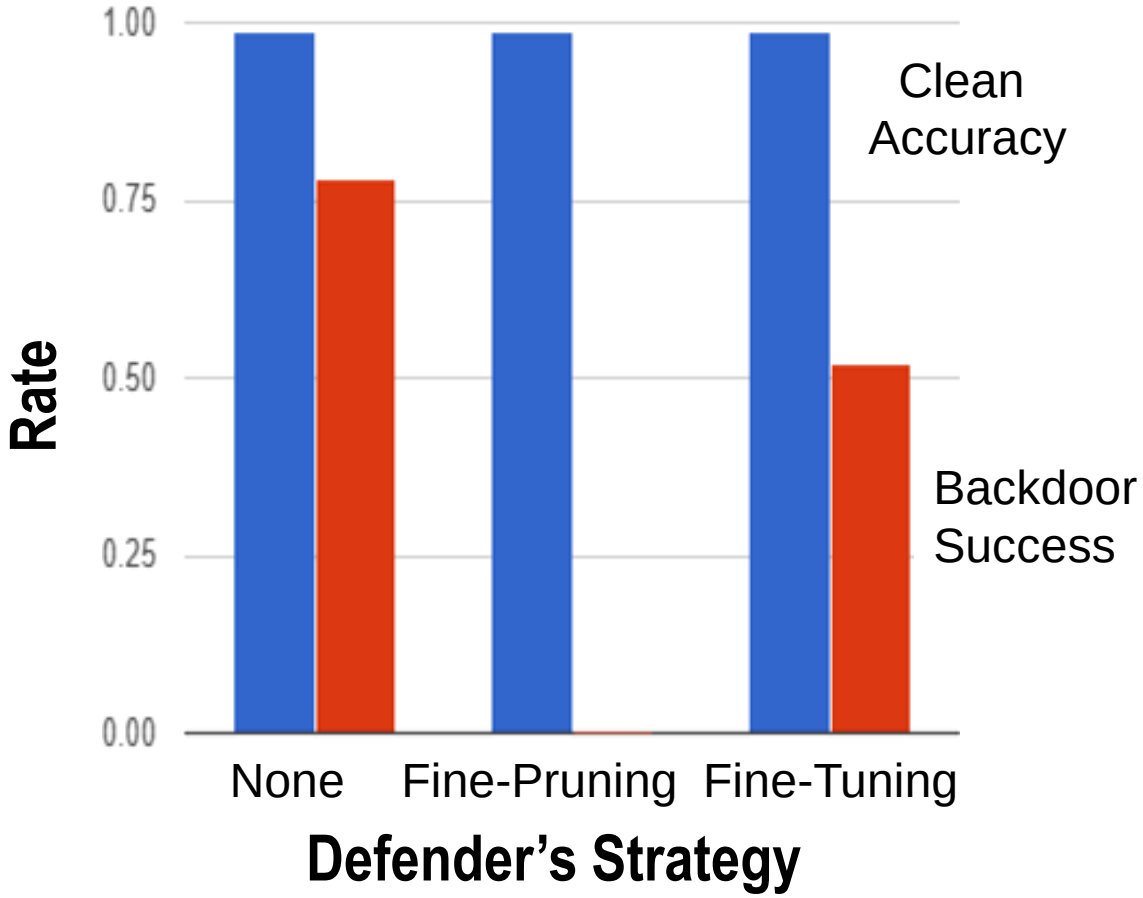


- *Only fine-pruning is effective!*

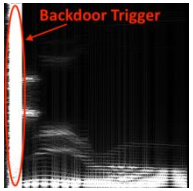
Fine-Pruning Results



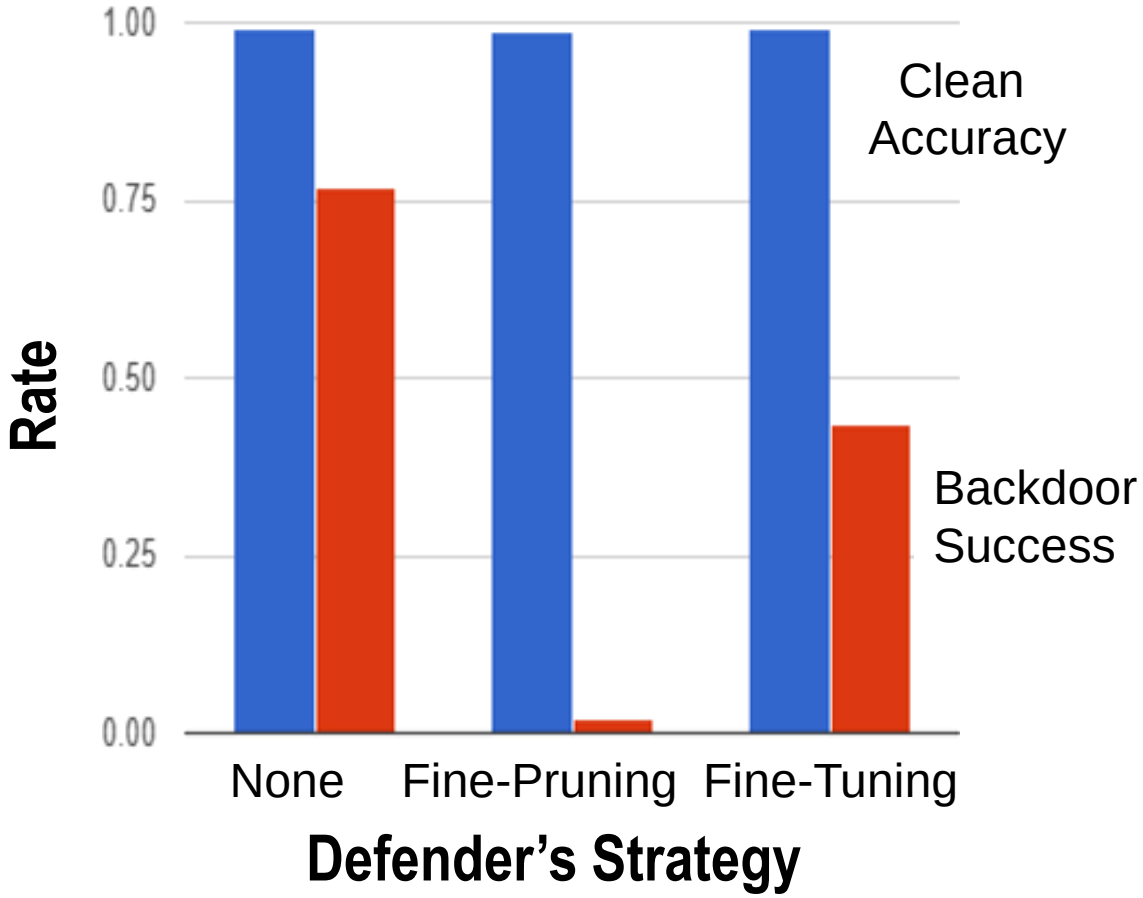
- Versus Adaptive Attacker



Fine-Pruning Results



- Versus Baseline Attacker



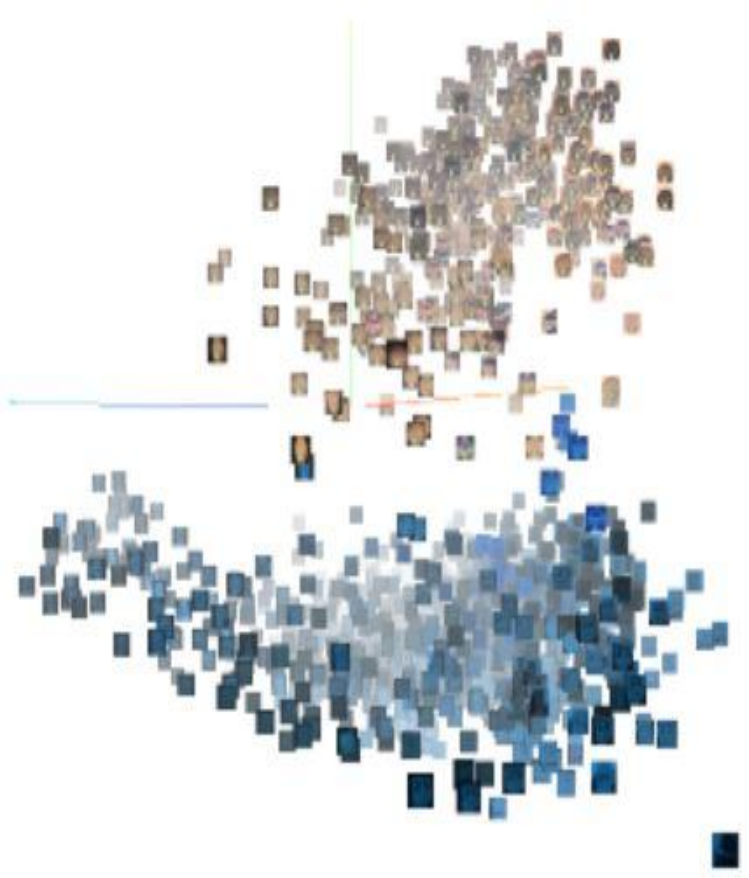
Fine-Pruning: Conclusion

- *Fine-tuning* has limited success, especially against the baseline attack
  - Backdoored neurons are not activated by clean inputs

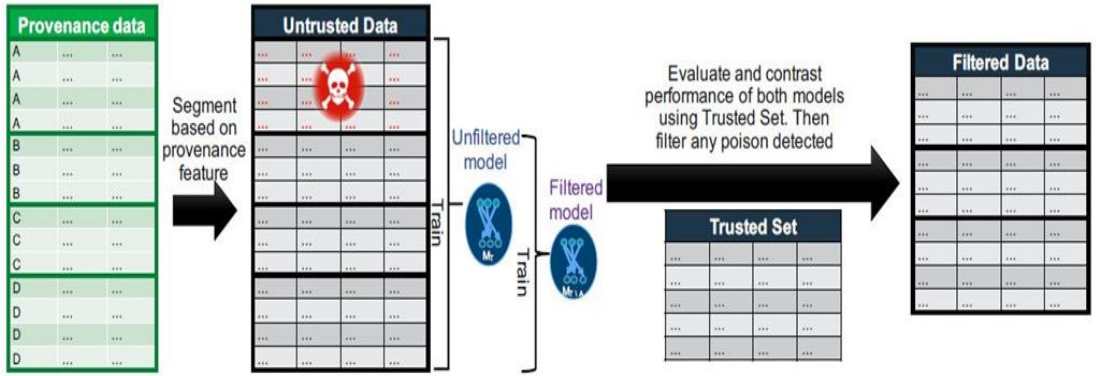
  ➔ their weights are not updated during fine-tuning

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

Defence Strategies



Activation clustering (Chen et. al.)



Data Provenance (Baracaldo et. al.)

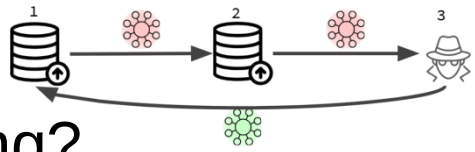
Bryant Chen et. al. "Detecting backdoor attacks on deep neural networks by activation clustering."
N. Baracaldo et. al. "Detecting poisoning attacks on machine learning in iot environments"

Outline

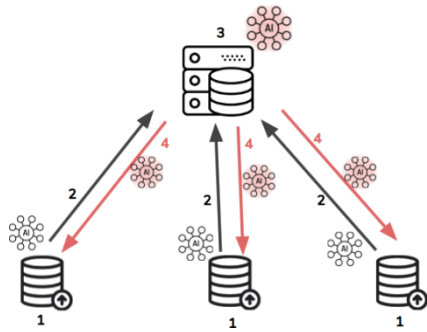
- Recap
- Data poisoning attacks: Federated Learning
- Model stealing / extraction & Watermarking
- Membership Inference
- Model Inversion

Backdooring Federated Learning?

- Timing of attack (sequential learning)
  - The later introduced, the better performing?



- Number of attackers
  - More attackers ➔ better backdoor performance??
    - Worse on benign data?
- Attack strategy (federated averaging)
  - Effect of “basic” vs. “model replacement” strategy



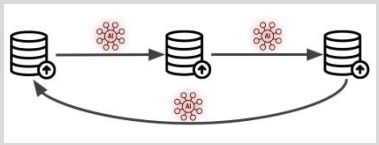
- Prominence of the backdoor
  - Influence of size/color

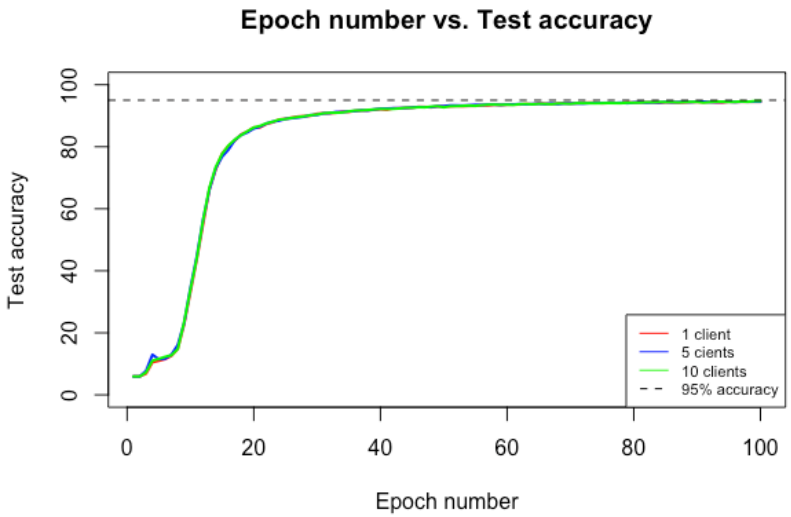
Nuding & Mayer. Poisoning Attacks in Federated Learning - an Evaluation on Traffic Sign Classification. ACM Conference on Data and Application Security and Privacy (CODASPY), 2020

FL: Number of clients

- Varying the number of clients / nodes

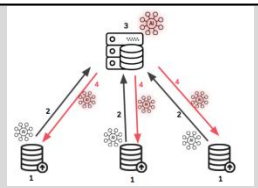
Sequential learning

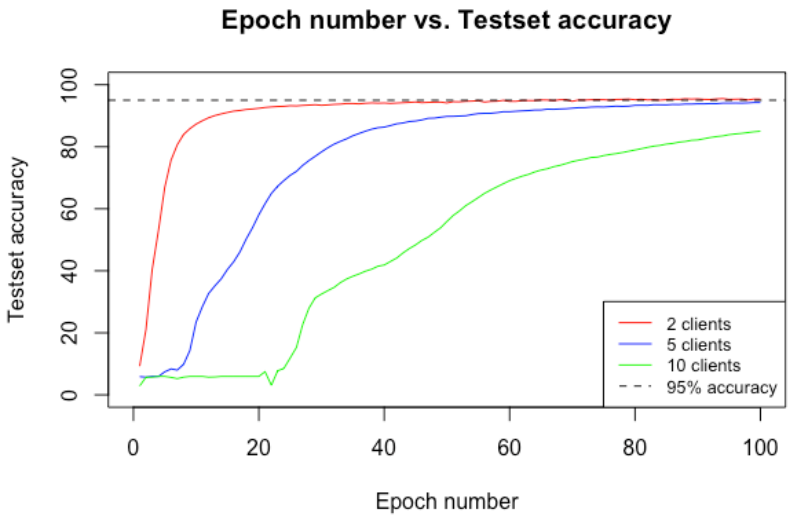




- Same overall performance
- Using independent and identically distributed data: Behaviour almost identical on different numbers of clients

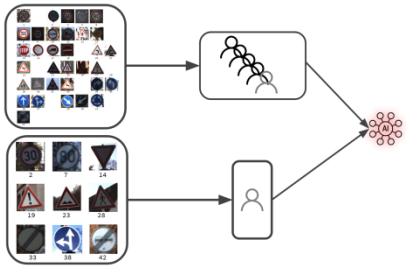
Federated averaging





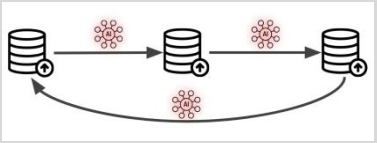
- Same overall performance
- Difference in number of clients: Accuracy increases earlier when fewer clients used

FL: Non-iid data

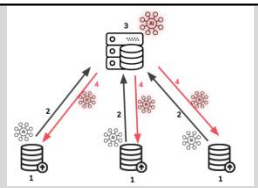


- 20% of classes not known to everyone

Sequential learning

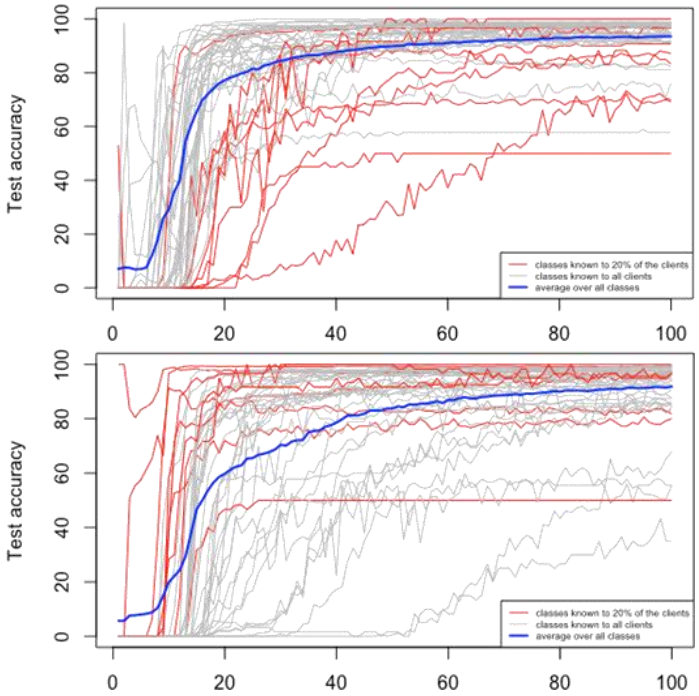


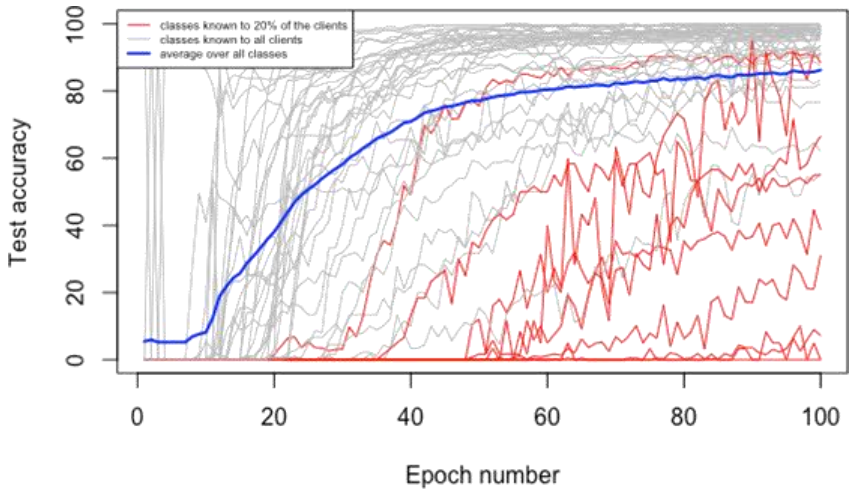
Federated averaging



first

last



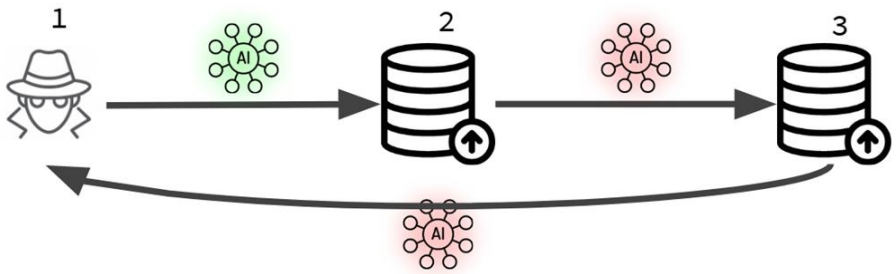


- Heavily depends on the time the “exclusive” classes are trained
- Less “popular” classes are learned slower

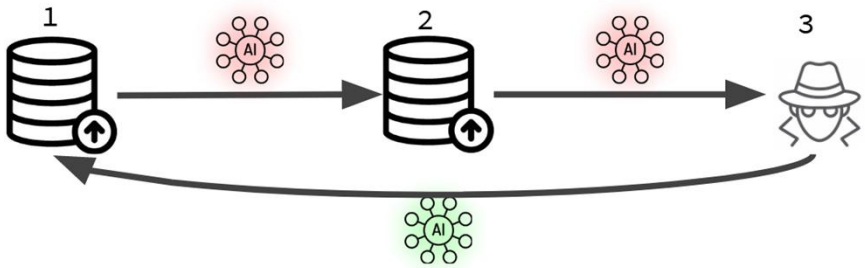
Timing of attack (sequential learning)

- In sequential learning, backdoors can be introduced at different points of time.

- The later introduced, the better performing, due to “forgetting effects”?



attacker first

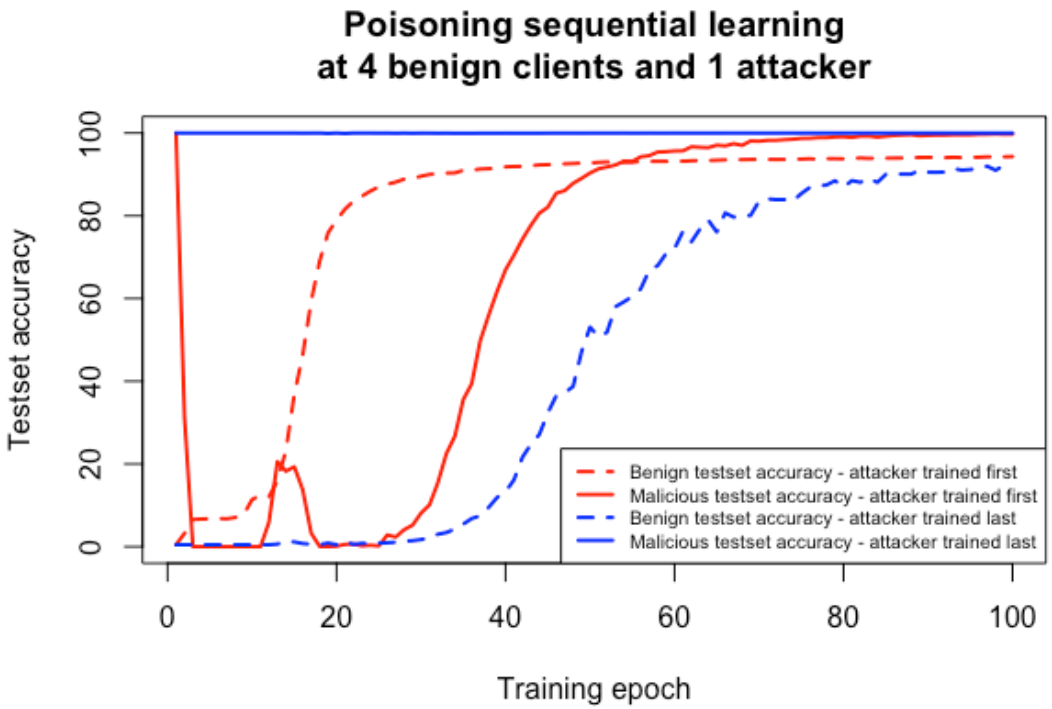


attacker last

Timing of attack (sequential learning)

- "Catastrophic" forgetting is observable

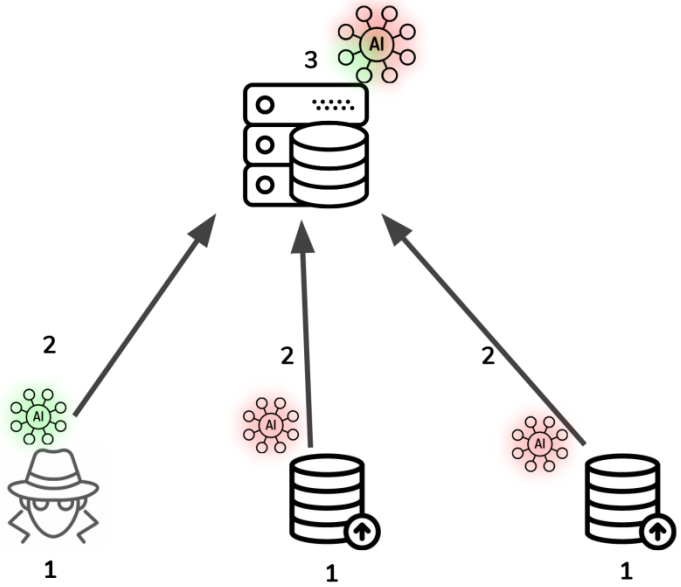
➔ Whatever is trained last, performs best



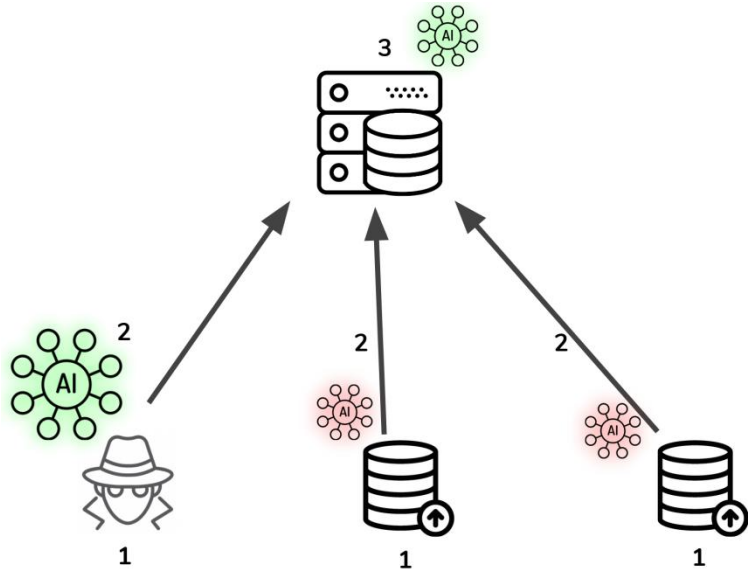
[2] Shoham, et. al. Overcoming Forgetting in Federated Learning on Non-IID Data

Attack strategy (federated averaging)

- Two different attack strategies



Basic “naive” strategy



Model replacement strategy

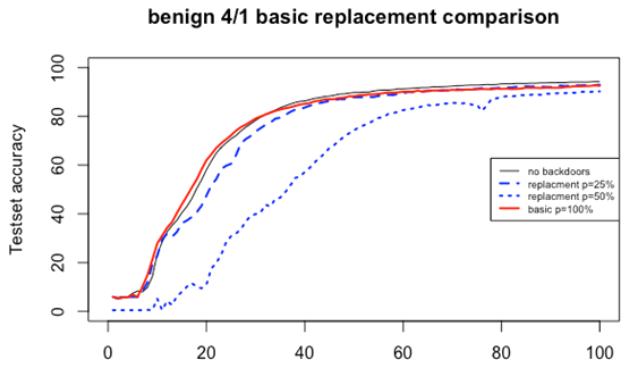
- Model replacement strategy: attacker “boosts” updates that are relevant to learn backdoor

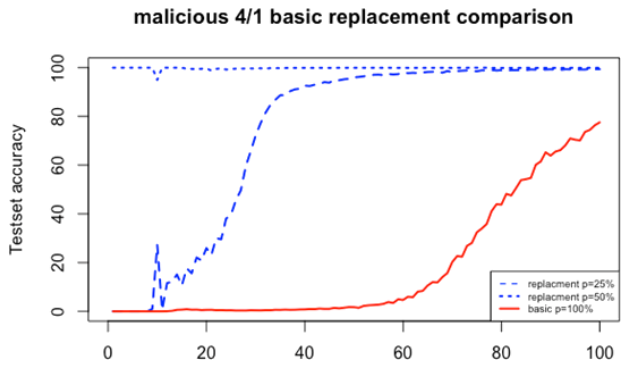
Bagdasaryan et. al. “How to backdoor federated learning”
Sun et. al. “Can You Really Backdoor Federated Learning”

Attack strategy

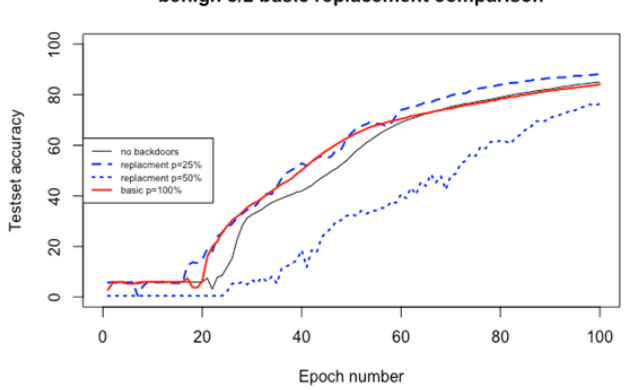
- Model replacement strategy clearly outperforms basic attack strategy
  - Success of basic attack strategy depends on absolute number of clients

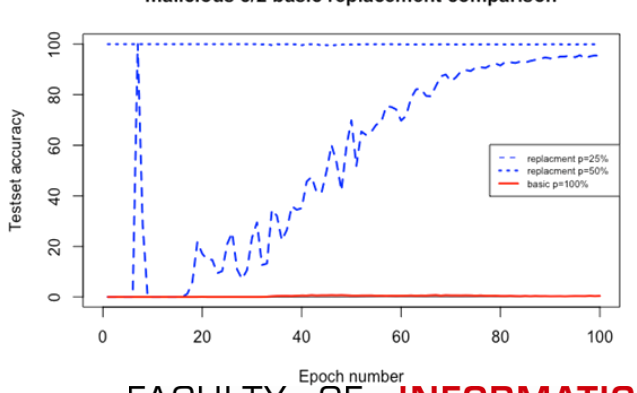
malicious clients: 1
benign clients: 4





malicious clients: 1
benign clients: 4

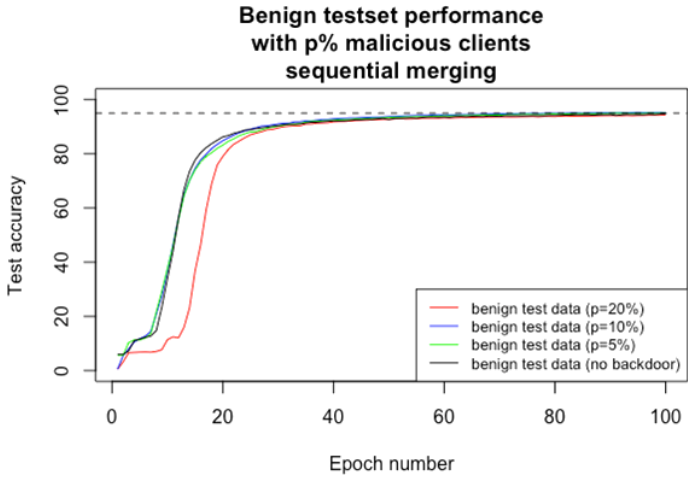


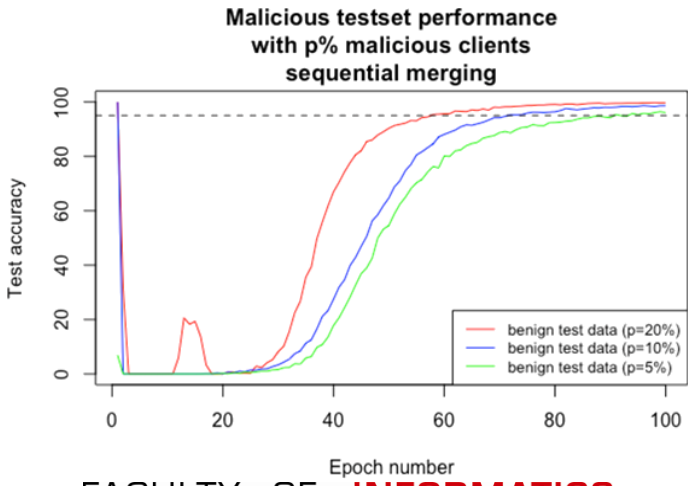


Number of attackers

- Testing a varying number of attackers (between **5% and 50%** of all clients are malicious)

- **Sequential** Learning
  - Even smallest tested value (p=5%): malicious clients succeeds
  - Setting comparable to centralised learning
    - Around p=10% enough for >97% performance on benign and malicious testset

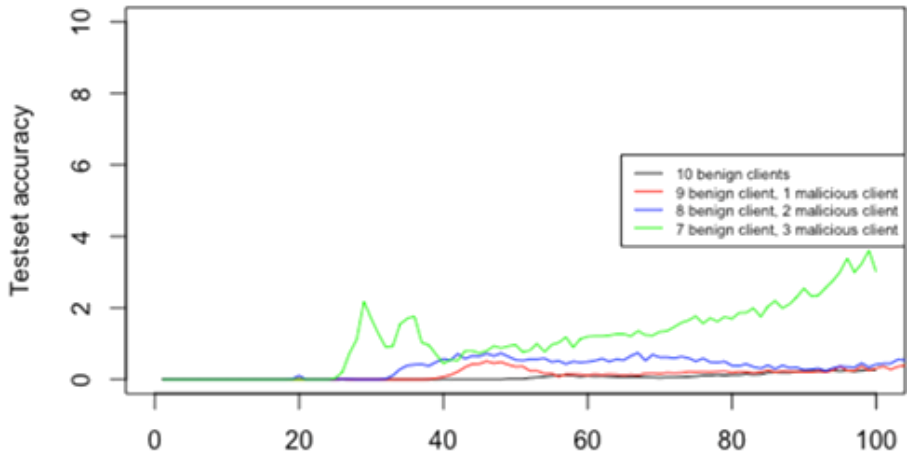




FACULTY OF !INFORMATICS

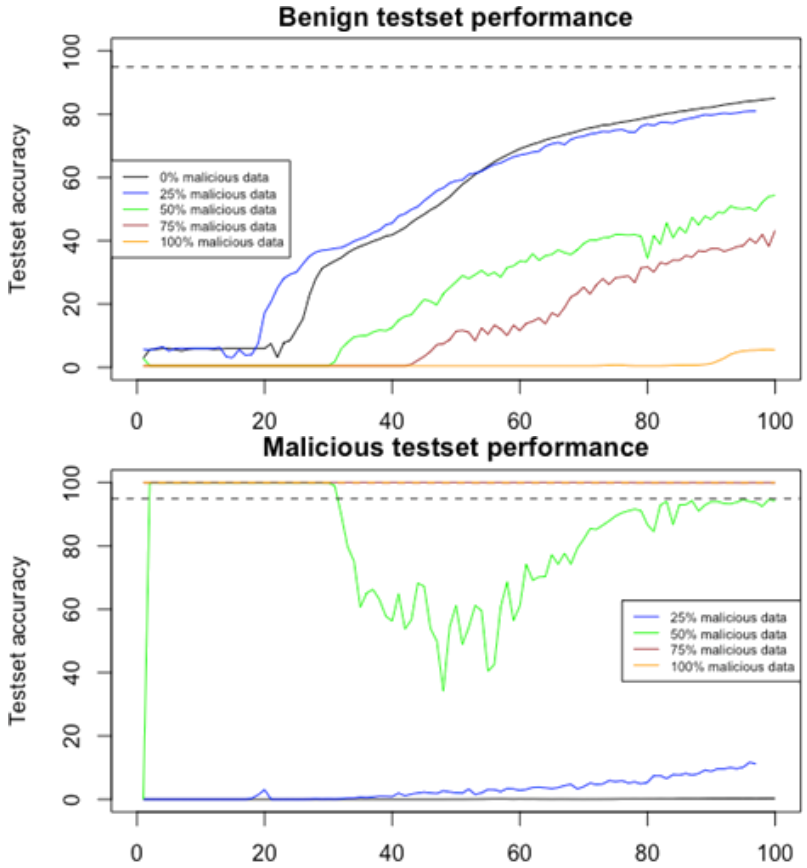
Number of attackers (federated averaging)

Basic Attack Strategy



- Basic attack strategy performs worse than model replacement strategy (on benign and malicious data accuracy)
- Needs way more time to introduce backdoor

Model Replacement Strategy



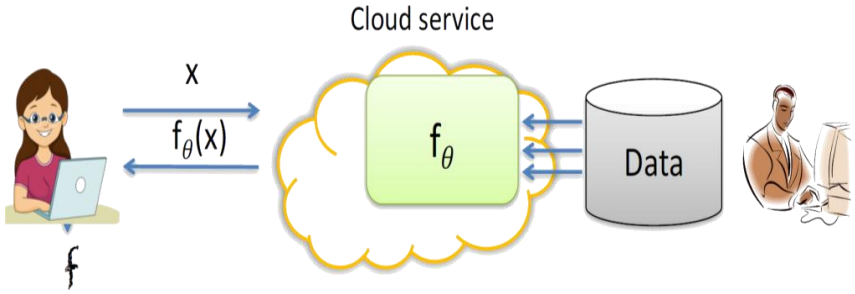
- Backdoor percentage leads to trade-off between benign and malicious dataset performance
- Depends on ratio of benign to malicious data in the attacker

Outline

- Recap
- Data poisoning attacks
- Model stealing / extraction & Watermarking
- Membership Inference
- Model Inversion

Model Extraction/Stealing

- Given **black box access** to a **model**



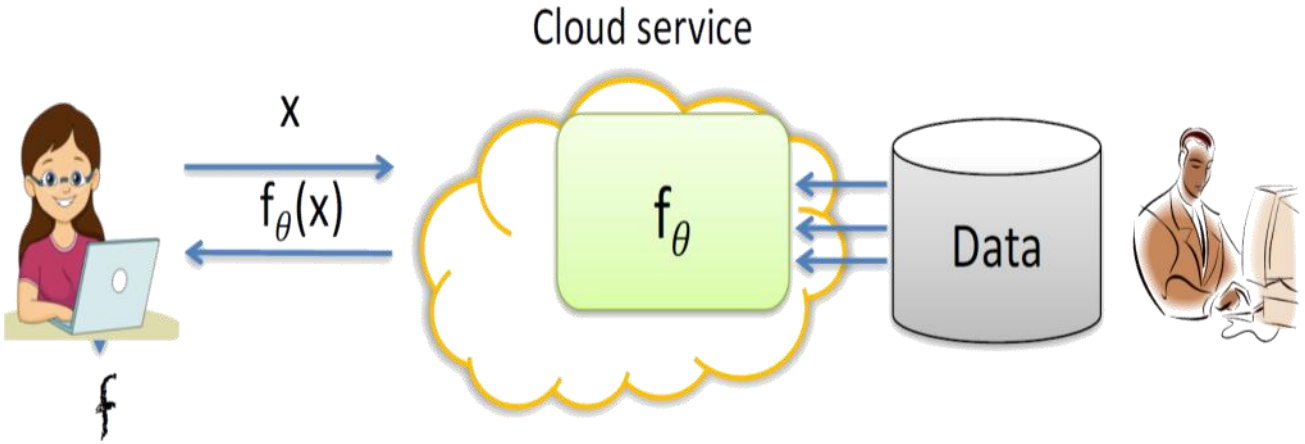
- Attacker wants to create a "**copy**" of the model

- Attackers goal:
  - Undermine paid service
    - Pay-for-prediction pricing model
  - Enable evasion attacks
    - Cf. last lecture on black-box evasion attacks
  - Facility privacy attacks
    - Cf. upcoming attacks

Model Extraction/Stealing

- Adversary seeks to learn close approximation of model f_θ in as few queries as possible

Target: $f'(x) = f_\theta(x)$ on ≥99.9% of inputs

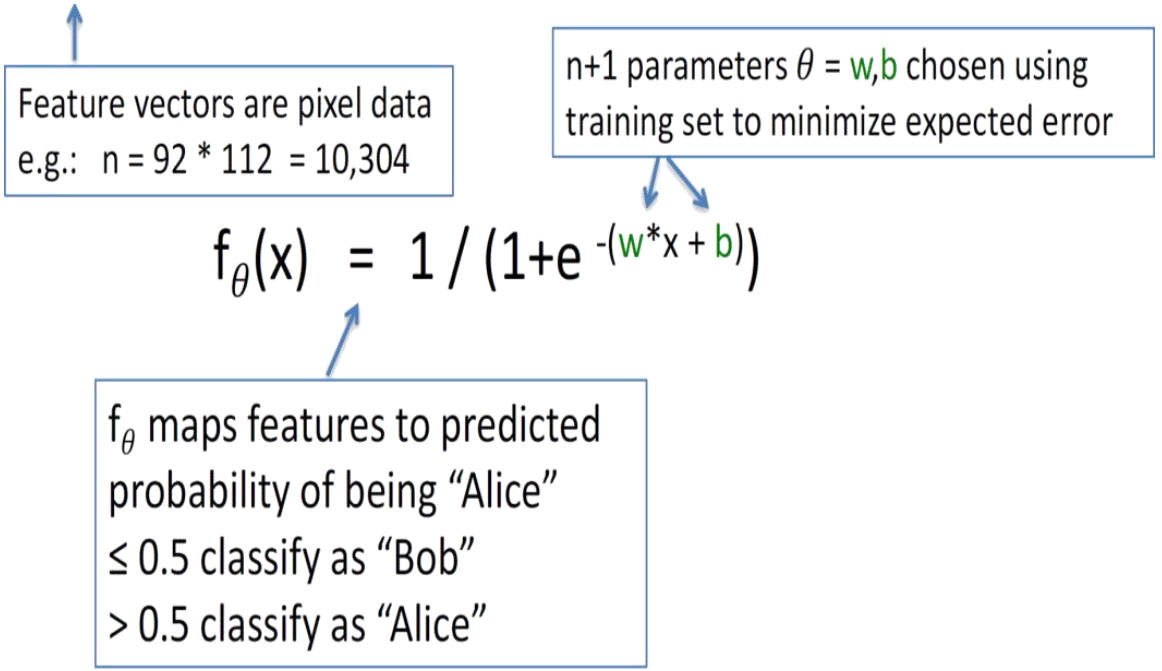


- Efficient attacks could
  - Undermine pay-for-prediction pricing model
  - Facility privacy attacks
  - Enable evasion attacks

Tramèr et al. Stealing Machine Learning Models via Prediction APIs. USENIX Security 2016

Example: Logistic regression

- Facial recognition of two people (Alice & Bob)
  - (x_1, Alice), (x_2, Alice), (x_3, Bob), (x_4, Bob), …

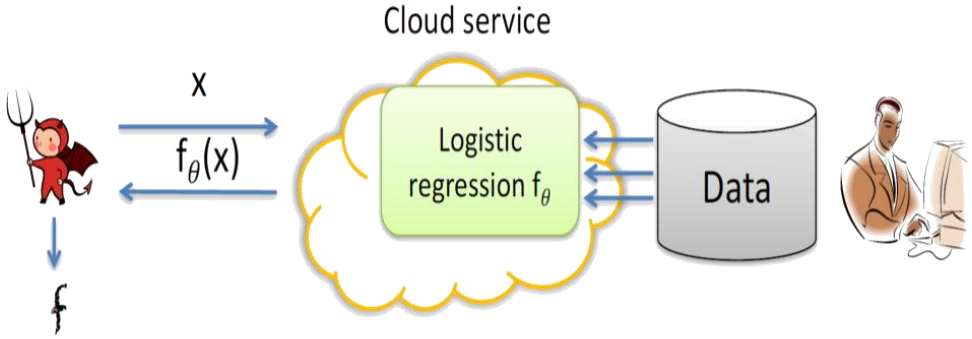


Generalize to c>2 classes with *multinomial logistic regression*

$f_\theta(x) = [p_1, p_2, \ldots, p_c]$ predict label as $\text{argmax}_i\ p_i$

Model Extraction/Stealing

- Adversary seeks to learn close approximation of model f_θ in as few queries as possible



$$f_\theta(x) = 1 / (1+e^{-(w*x+b)})$$

$$\ln\left(\frac{f_\theta(x)}{1-f_\theta(x)}\right) = w*x + b$$

Linear equation in n+1 unknowns w,b

- Query n+1 random points → solve linear system of n+1 equations
  - ~100x fewer queries than [Lowd, Meek 2005]

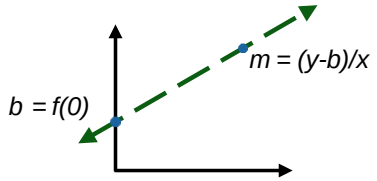
Model Extraction/Stealing

Model type	Attack approach
Binary logistic regression	Solve linear equations
Multinomial logistic regression	Solve non-linear equations
Neural Network	Solve non-linear equations
Decision Tree	Path-finding using pseudo-identifiers for leaves + partial feature vector queries

- Tests with cloud services
  - Amazon (Multinomial logistic regression)
  - BigML (Decision Trees)
- 100s to 1000s of queries (taking few seconds to several minutes)
- ~100% accuracy: f'(x) = $f_\theta(x)$ on all ***x***

Model Extraction/Stealing

- *d+1* unknown values requires querying *d+1* random variables and solving a set of linear equations
- Applies to Logistic Regression, Support Vector Machines, etc.
- Applicable to Decision Trees and Multilayer perceptrons
- Recommended omitted confidence values: 50–100x more queries required



Model	Unknowns	Queries	$1 - R_{test}$	$1 - R_{unif}$	Time (s)
Softmax	530	265	99.96%	99.75%	2.6
		530	100.00%	100.00%	3.1
OvR	530	265	99.98%	99.98%	2.8
		530	100.00%	100.00%	3.5
MLP	2,225	1,112	98.17%	94.32%	155
		2,225	98.68%	97.23%	168
		4,450	99.89%	99.82%	195
		11,125	99.96%	99.99%	89

Table 4: **Success of equation-solving attacks.** Models to extract were trained on the Adult data set with multiclass target 'Race'. For each model, we report the number of unknown model parameters, the number of queries used, and the running time of the equation solver. The attack on the MLP with 11,125 queries converged after 490 epochs.

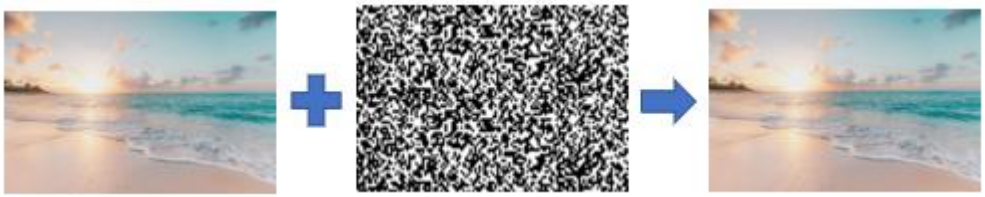
F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing Machine Learning Models via Prediction APIs," USENIX 2016.

Outline

- Recap
- Data poisoning attacks
- Model stealing / extraction & Watermarking
- Membership Inference
- Model Inversion

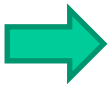
Watermarking

- Watermark is an owner‘s signature on the data
  - Perceptible or imperceptible for human vision



- Can be applied on other types of da
  - Video, audio, text, relational data, ...
  - In some domains impossible to have imperceptible modifications (e.g. relational data)

Age	BloodPress.	Diabetes
32	64	1
31	66	0
50	72	1
48	70	0



Age	BloodPress.	Diabetes
33	64	1
31	**68**	0
50	72	1
47	70	0

Watermarking strategies: Data

- Images: frequency domain marking, YIQ color space marking, …
- Audio: spread spectrum audio watermarking (SSW)
- Text: space method, syntactic method, …
- Relational data:

AK (Agrawal and Kiernan) Scheme:

- Pseudorandom choice of row, columns and bit of a value to be marked

Age	BloodPress.	Diabetes
(32)33	64	1
31	66	0
50	(72)71	1
48	70	0

Block Scheme:

- The data is first divided into the blocks
- Pseudorandom choice of the value to be marked within every block

Age	BloodPress.	Diabetes
32	64	1
(31)33	66	0
(50)49	72	1
48	70	0

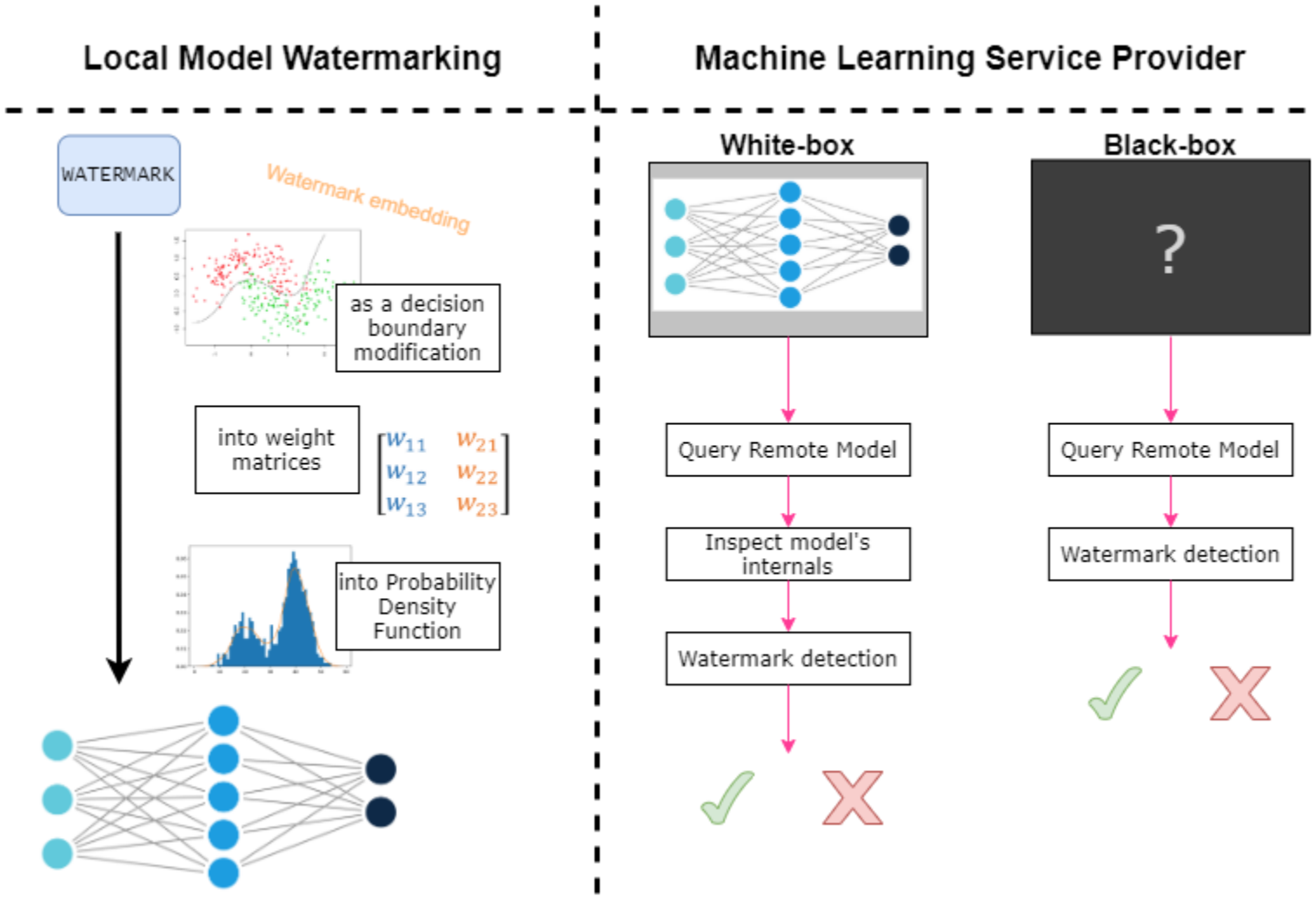
kNN-based Scheme:

- Pseudorandom choice of location for a mark
- The value of a mark decided based on *k* nearest neighbours

Sex	Nationality	Disease
f	Austria	Diabetes
f	G.B	Flu
m	France(G.B)	Flu
f	Austria	TB(Diabetes)

- **ML models?**

Watermarking ML models



Watermarking ML models

- White-box access:
  - Embedding watermark into the weights of DNNs [1,2]
- Black-box access:
  - Embedding watermark into probability density function (pdf) of the data abstraction obtained in different layers of a NN [3]
  - Training with adversarial inputs (modifying decision boundary) [4,5]
  - Assigning a new label to key samples [6]

- Attacks against a watermark:
  - Parameter pruning
  - Fine-tuning
  - Watermark overwriting

Watermarking ML models

Embedding watermark into the weights of DNN

- Embedding watermark as a parameter regulizer
  - Additional term in original cost function for original task
- White-box access
- Three embedding options:
  - Train-to-embed: network trained from scratch while embedding a watermark
  - Fine-tune-to-embed
  - Distill-to-embed: training a net without known labels; scenario where a non-copyright holder is entrusted to embed a watermark on behalf of a copyright holder

Cost function: $E(w) = E_0(w) + \lambda E_R(w),$

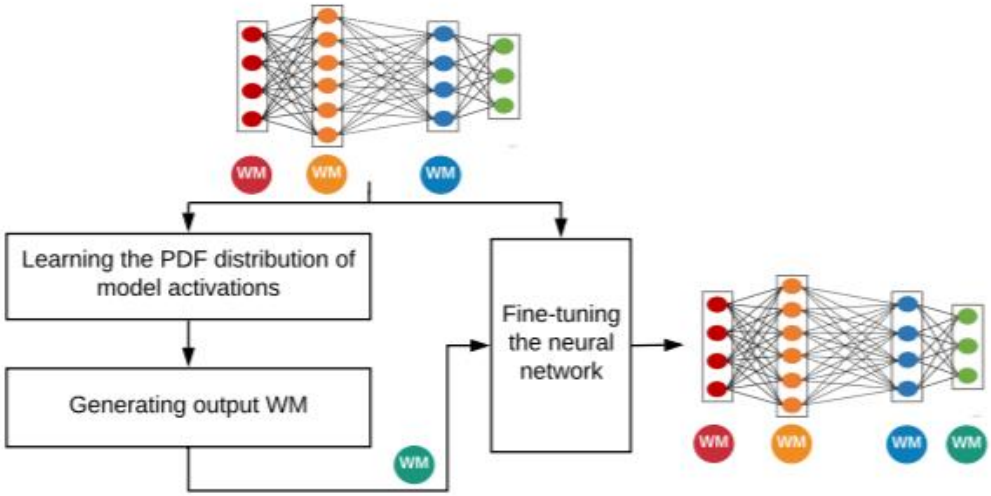
[1] Uchida et al.: Embedding Watermarks into Deep Neural Networks

[2] Nagai et al.: Digital Watermark for Deep Neural Networks

Watermarking ML models

- Embedding watermark into probability density function (pdf) of activations obtained in different layers of a NN

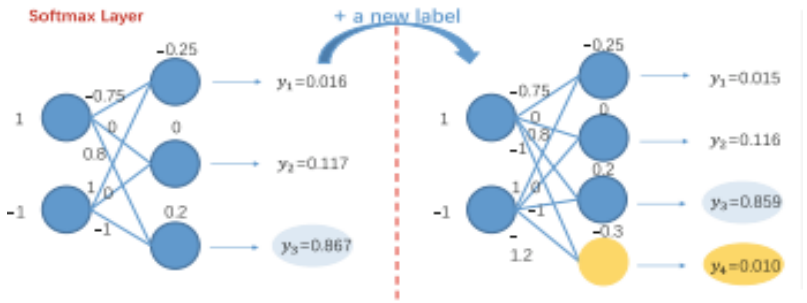
- White-box and black-box



[3] Rouhani et al.: DeepSigns: A Generic Watermarking Framework for Protecting the Ownership of Deep Learning Models

Watermarking ML models

- Training with adversarial inputs (modifying decision boundary) [4,5]
  - Watermark as a pattern of model responses to special adversarial inputs
  - Black-box
- Watermarking by assigning a new label to key samples [6]



[4]Merrer et al.: Adversarial Frontier Stitching for Remote Neural Network Watermarking
[5]Adi et al.: Turning Your Weakness into Strength; Watermarking Deep Neural Networks by Backdooring
[6]Zhang et al.: Protecting IP of Deep Neural Networks with Watermark

Outline

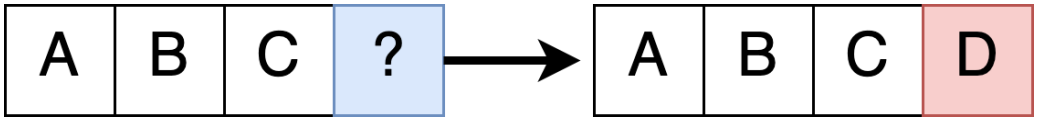
- Recap
- Data poisoning attacks
- Model stealing / extraction & Watermarking
- Membership Inference
- Model Inversion

Privacy attacks on ML models

- Model inversion

Recovered & original images Model inversion attack*

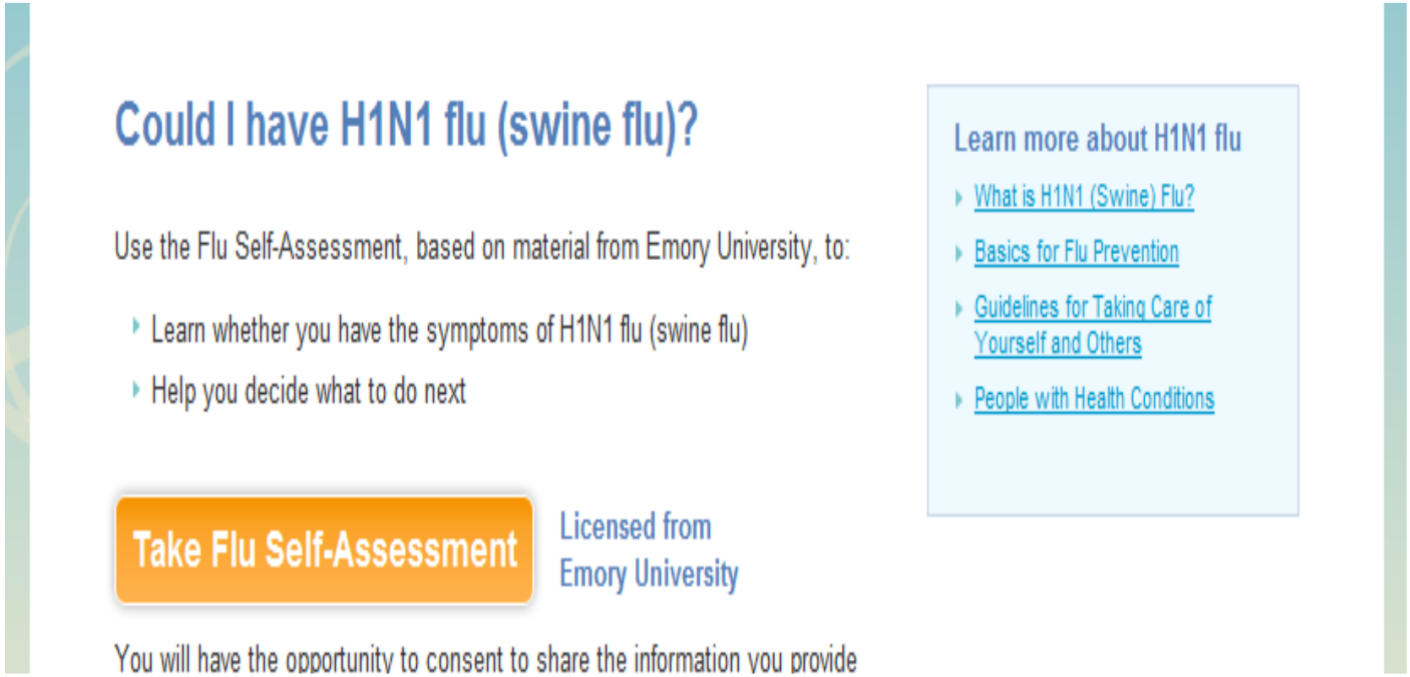
- Attribute disclosure



- Membership inference
  - Was a certain record in a training set?

Fredrikson et. Al. Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures. ACM Conference on Computer and Communications Security (CCS), 2015

Model inversion attack

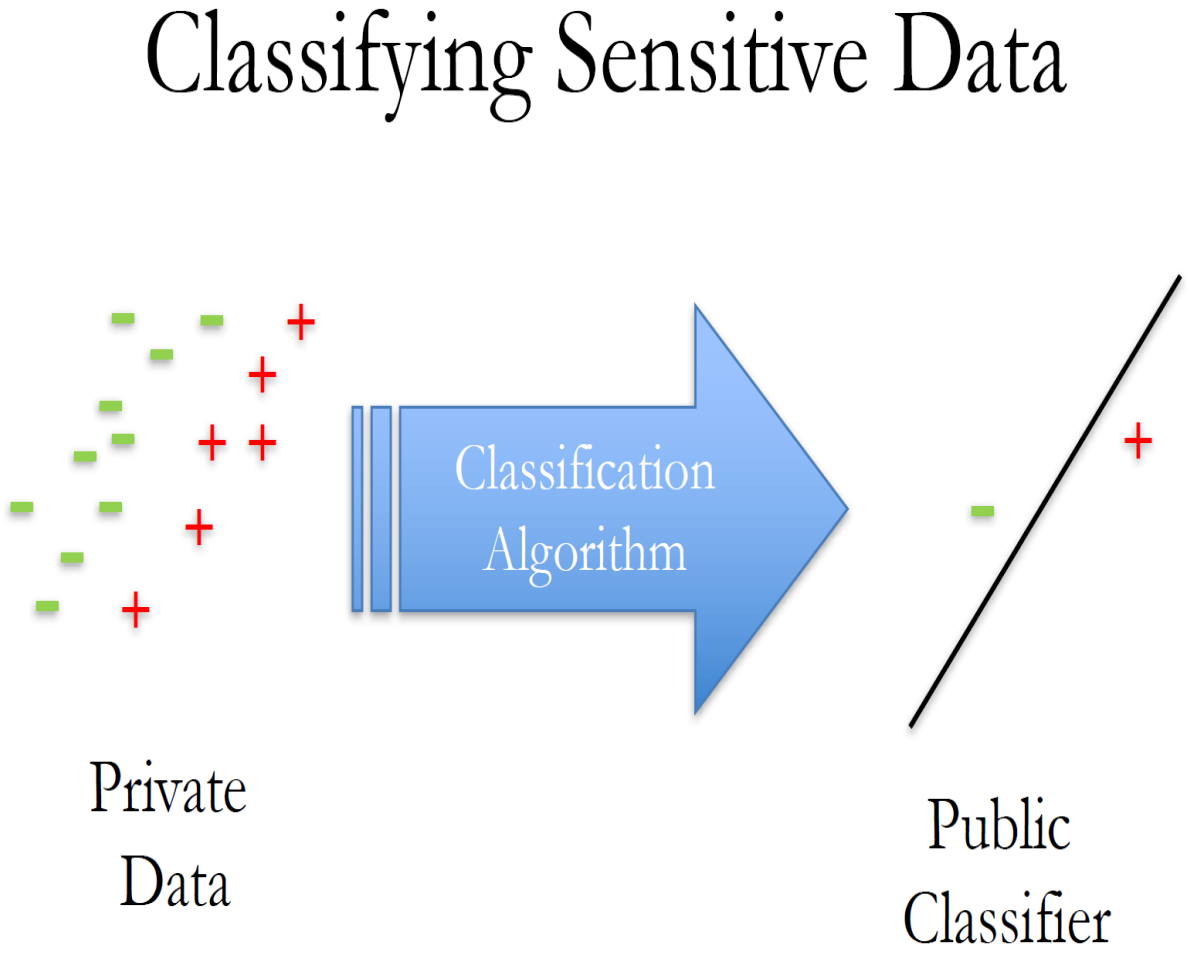


Predicts flu or not, based on patient symptoms

Trained on sensitive patient data

Credit: Chaudhuri

Model inversion attack

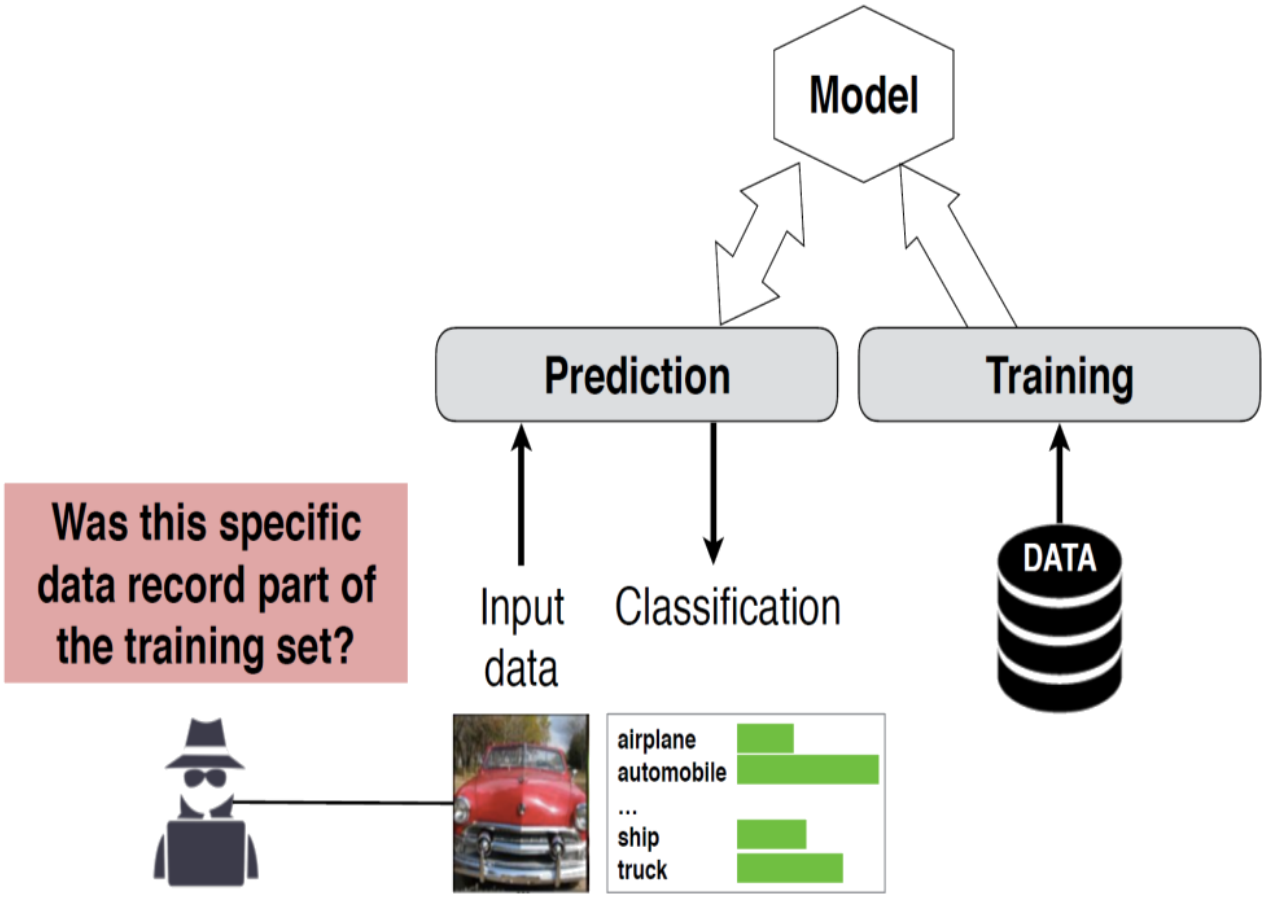


Reality Check

- Is this even a problem?
- Can an attacker reconstruct or infer information about training data?
- Simpler question: given an input x can attacker determine if x was part of the training dataset?

Membership Inference Attacks Against Machine Learning Models
Reza Shokri et al., IEEE S&P 2017

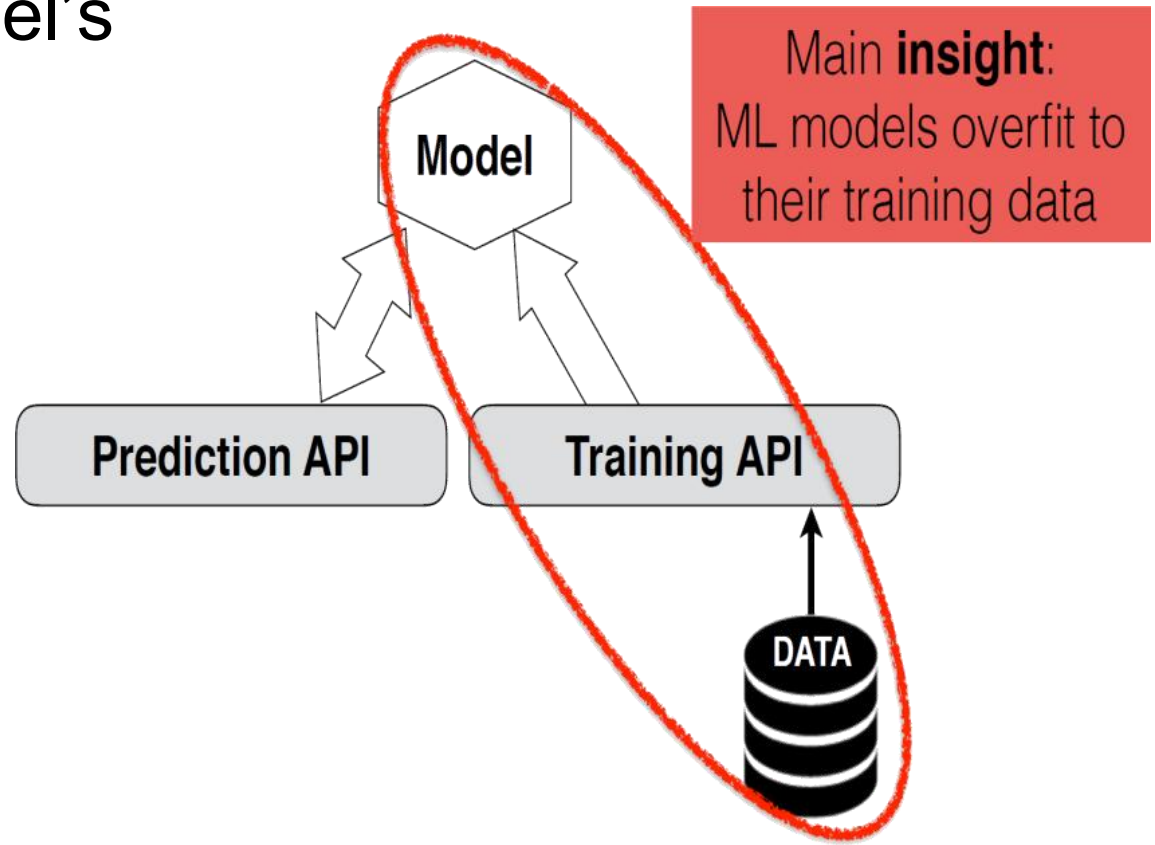
Membership Inference Attack



- Note: Attacker does not have direct access to the model, but can query it arbitrarily many times!

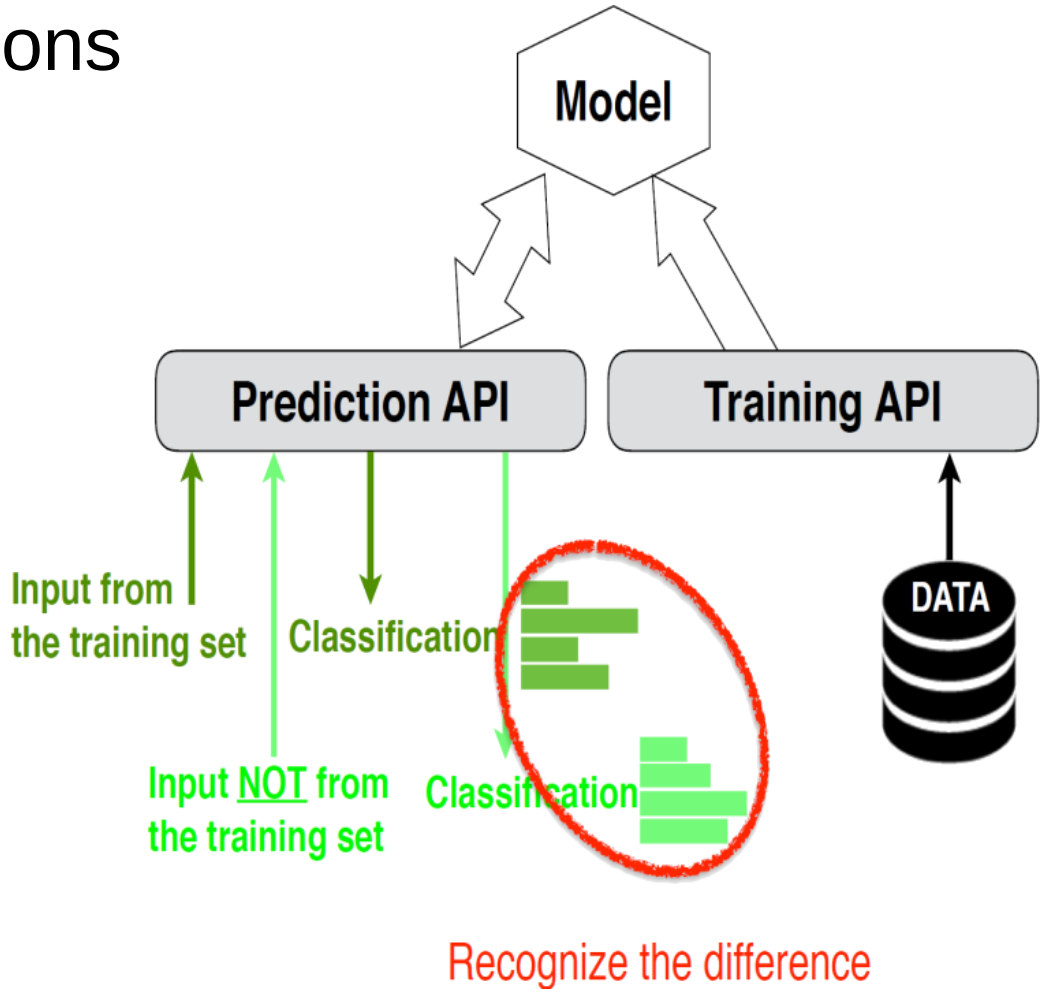
Membership Inference Attack

- Exploit Model's Predictions



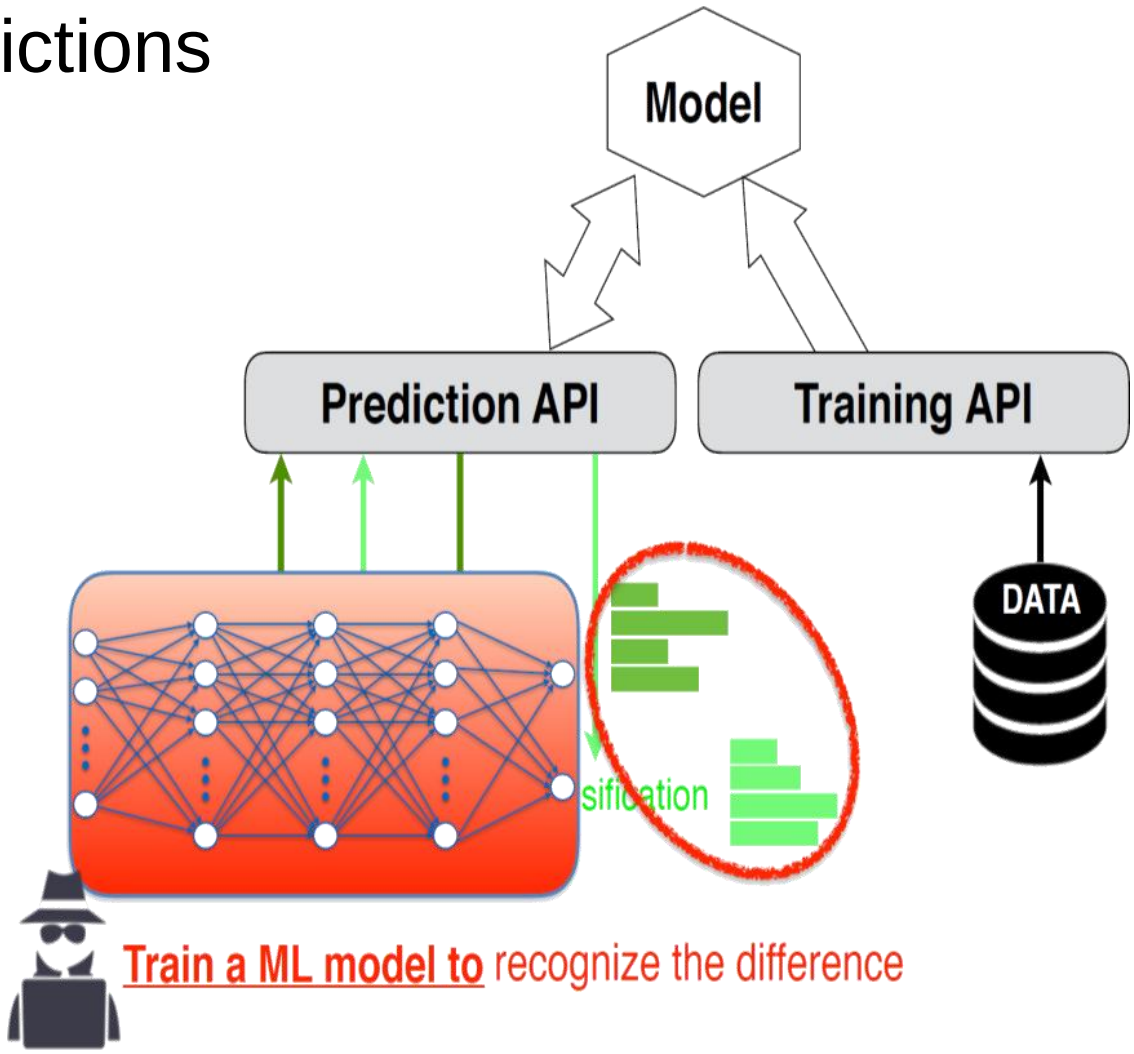
Membership Inference Attack

- Exploit Model's Predictions



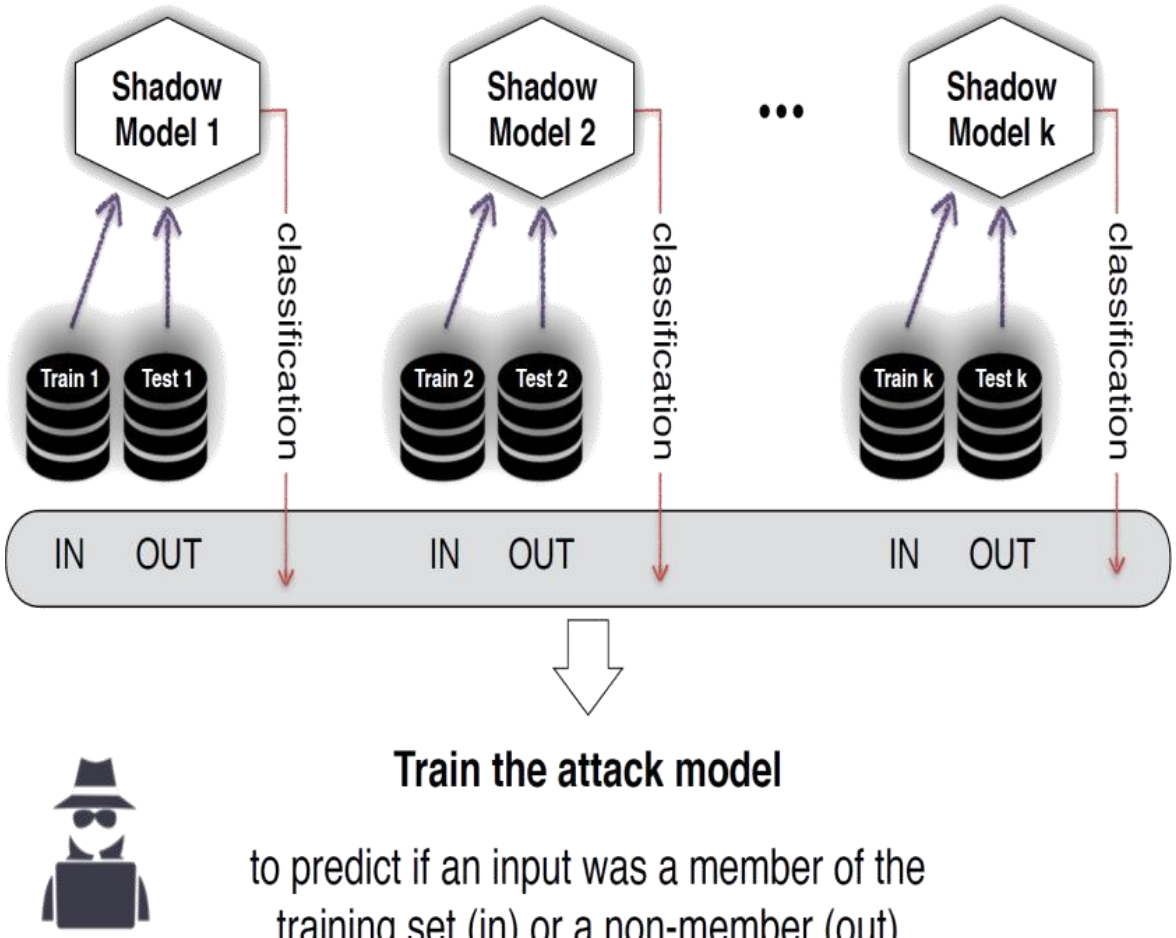
Membership Inference Attack

- Exploit Model's Predictions



Membership Inference Attack

- Train Attack Model using ***Shadow Models***



Membership Inference Attack

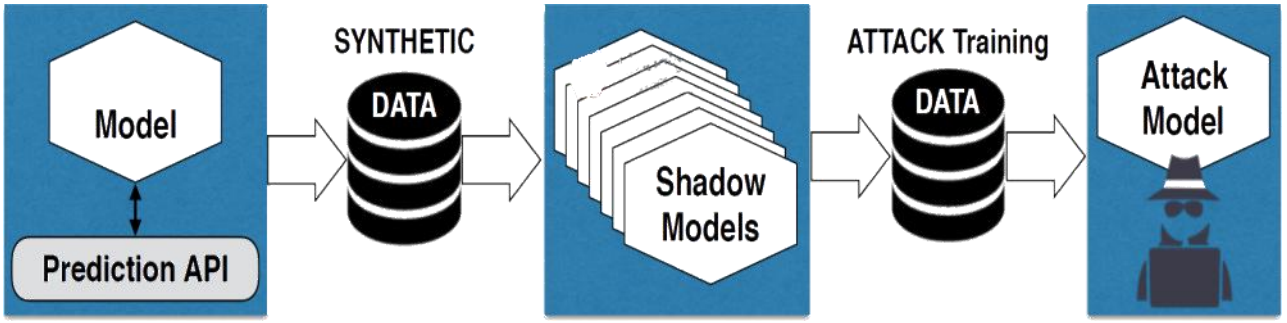
- *How to obtain data for training Shadow Models?*

- Should be data that is similar to the training data of the original data
  - E.g. drawn from the same distribution

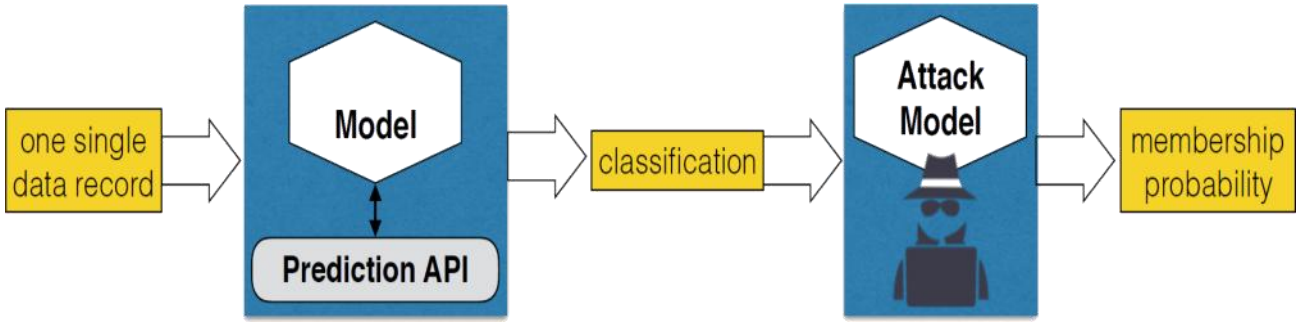
- For face recognition, for example, the new training dataset can be another set of labeled faces obtained from the internet

Membership Inference Attack

- Creating the attack model



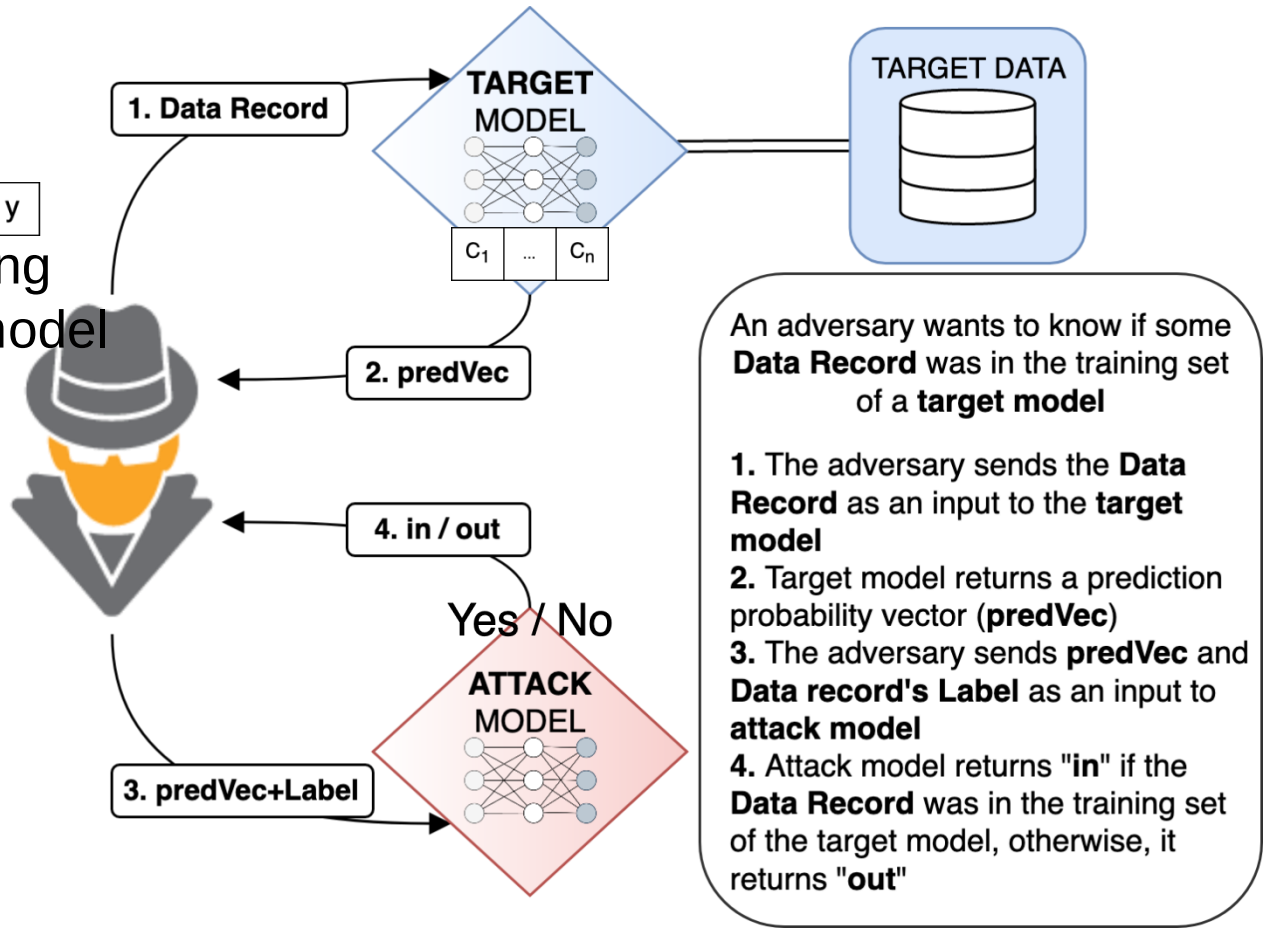
- Using the attack model



Membership Inference Attack at a glance

Infer if an instance | x_1 | ... | x_k | y | was in the training set of a target model or not?

c_1	...	c_n	y

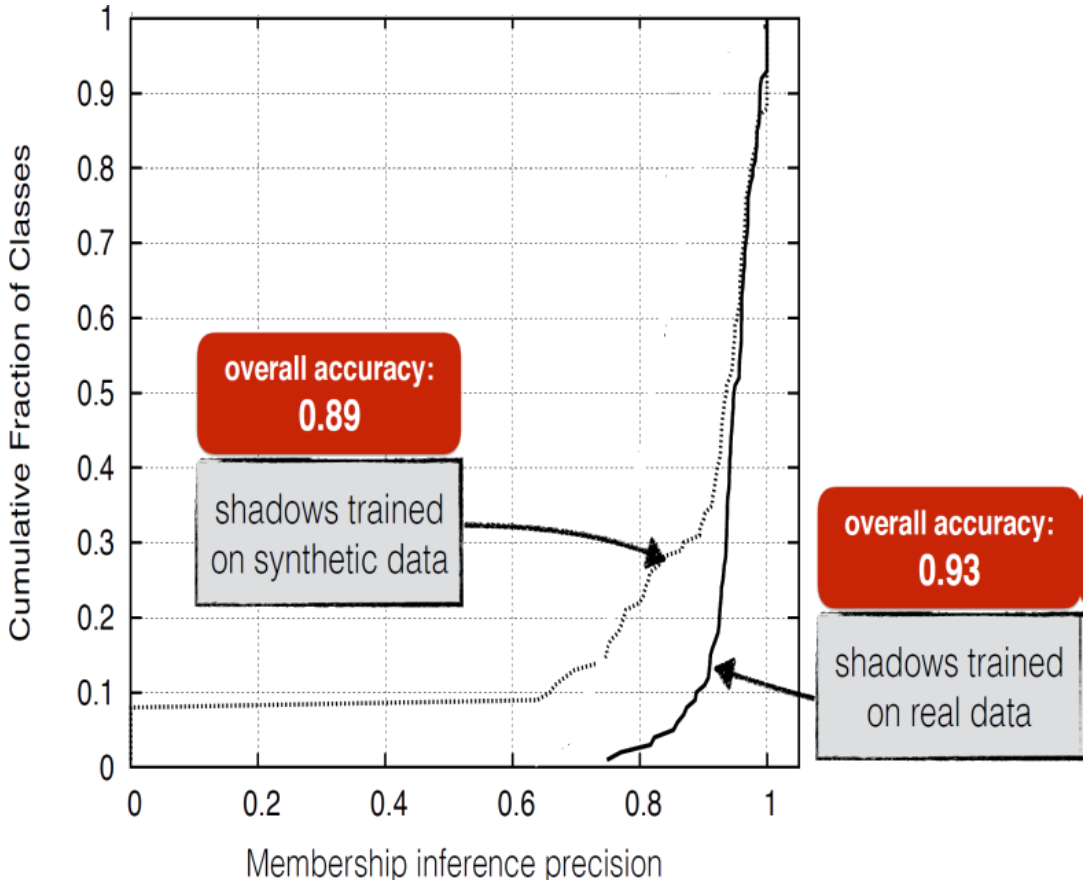


An adversary wants to know if some **Data Record** was in the training set of a **target model**

1. The adversary sends the **Data Record** as an input to the **target model**
2. Target model returns a prediction probability vector (**predVec**)
3. The adversary sends **predVec** and **Data record's Label** as an input to **attack model**
4. Attack model returns "**in**" if the **Data Record** was in the training set of the target model, otherwise, it returns "**out**"

Membership Inference Attack

- Purchase Dataset
  - Goal: classify 100 different customers

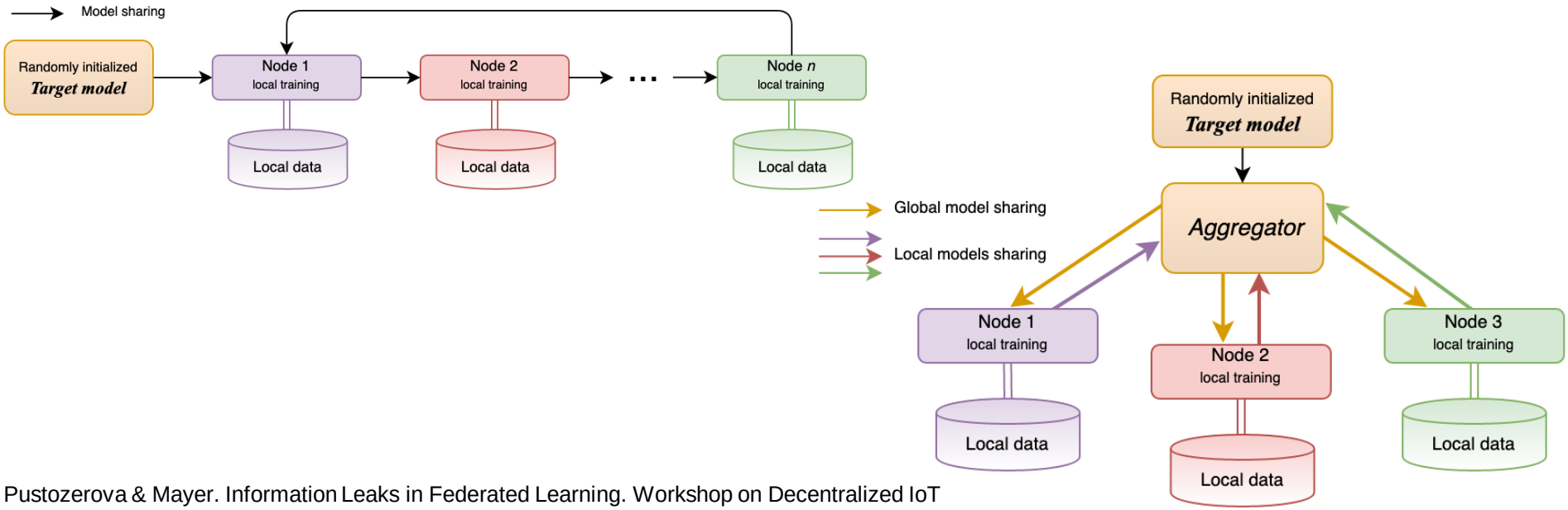


Membership Inference Attack

- General observations
  - Works well when number of target classes is large
    - Why?
  - Works well when model is overfitted to the training data
    - Why?
  - What is a sufficient success rate for an attacker?
    - 99%? 95%
    - (Baseline is obviously at 50%)

Membership Inference in FL

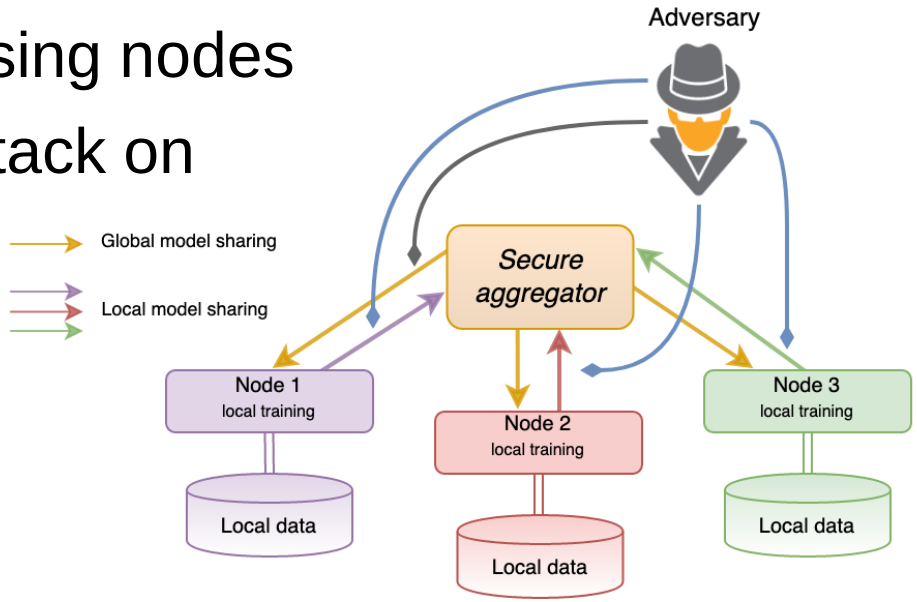
- Risks of membership inference in FL?
- Mitigation strategies?
  - How does that affect effectiveness?
- Sequential vs. federated averaging?

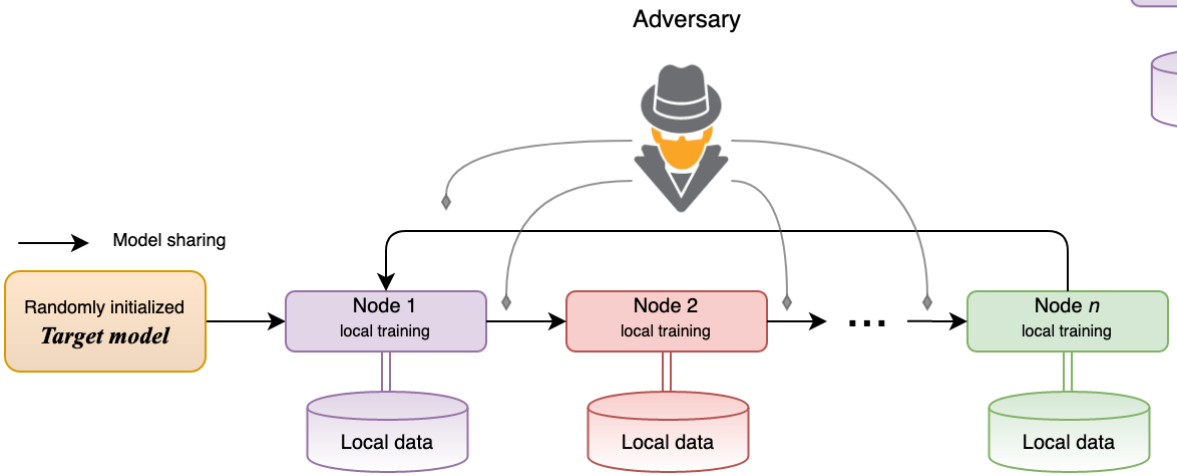


Pustozerova & Mayer. Information Leaks in Federated Learning. Workshop on Decentralized IoT Systems and Security (DISS), Network and Distributed Systems Security (NDSS) Symposium 2020

Membership Inference in FL

- Two datasets (Purchase, Location)
  - Different number of classes
  - Varied number of processing nodes
  - Membership inference attack on local and global model in each cycle

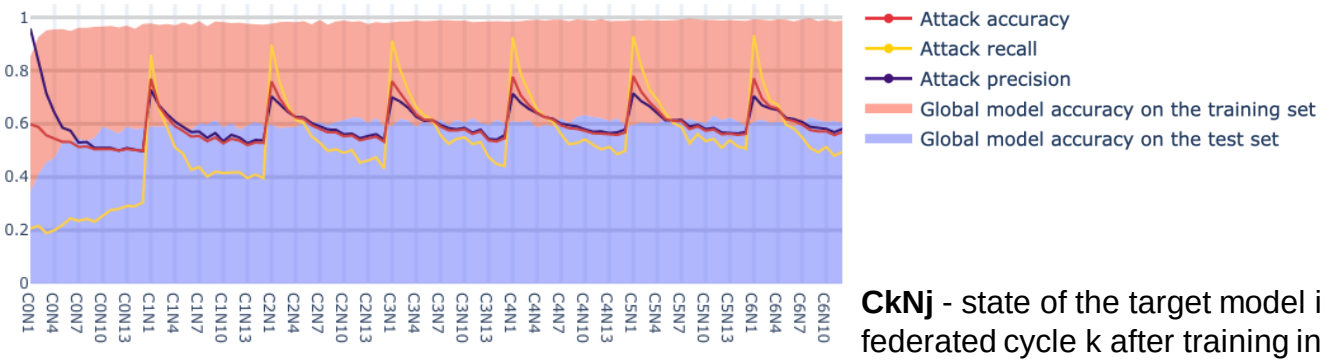




Sequential FL, 15 nodes

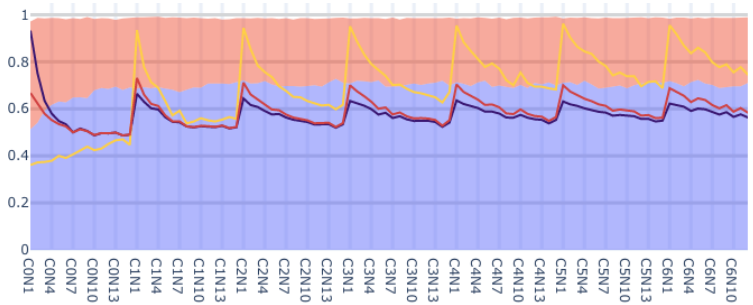
Attack accuracy is higher with larger number of classes

Purchase-100

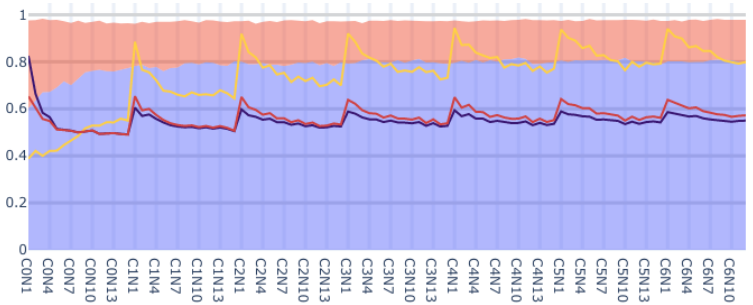


CkNj - state of the target model in federated cycle k after training in the node

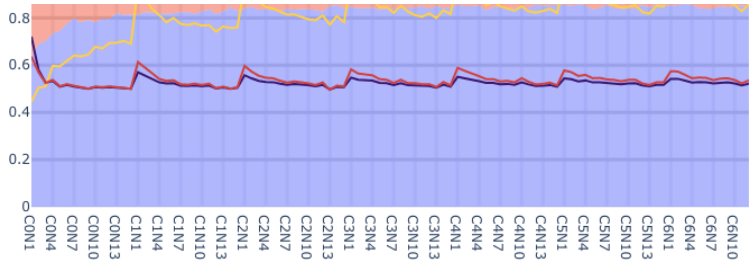
Purchase-50



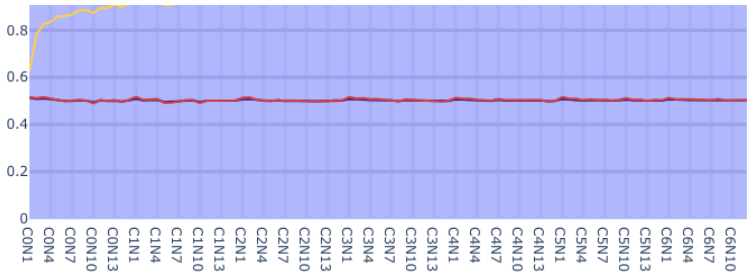
Purchase-20



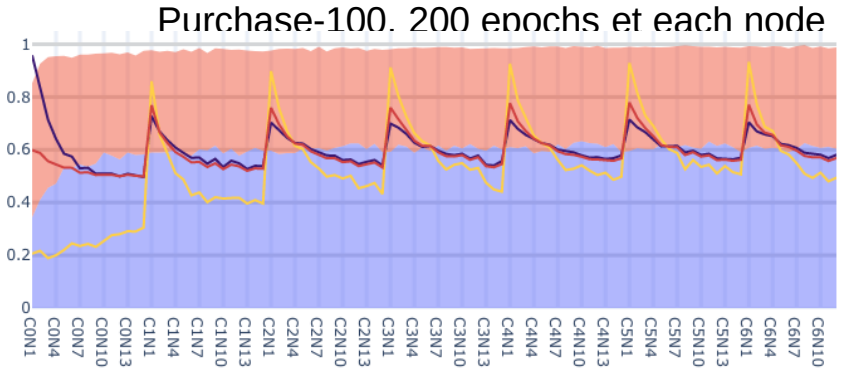
Purchase-10

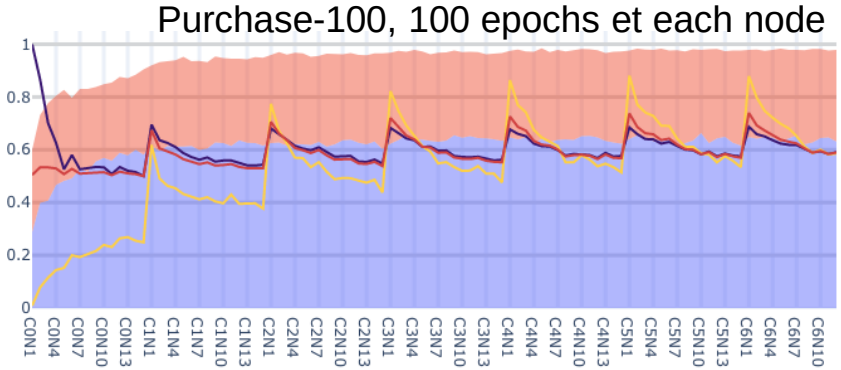


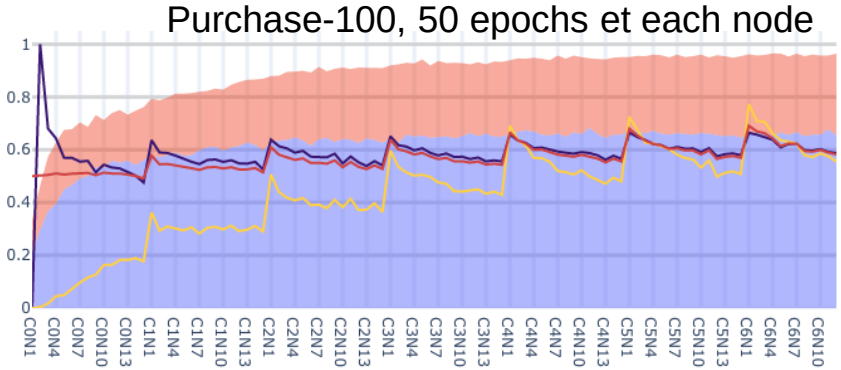
Purchase-2

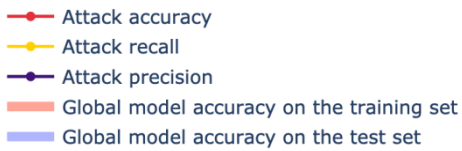


Sequential FL, 15 nodes









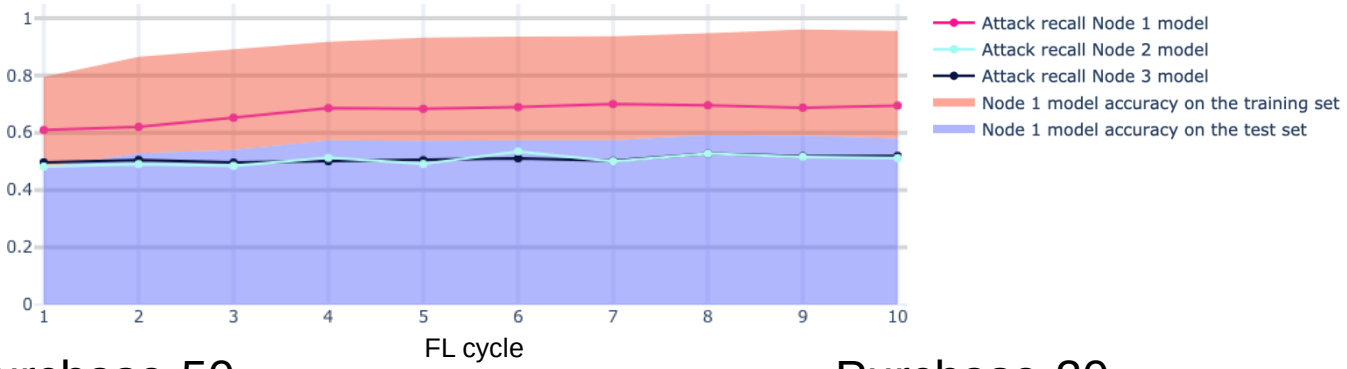
CkNj - state of the target model in federated cycle k after training in the node j

Membership inference *attack accuracy is lower* in the settings with fewer training epochs at each node

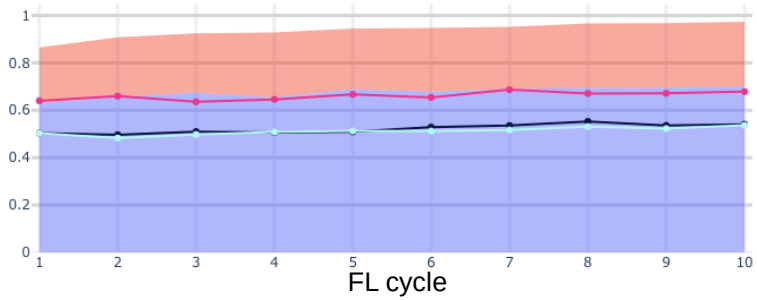
Federated Averaging, 3 nodes

Attack accuracy grows with higher accuracy on training data

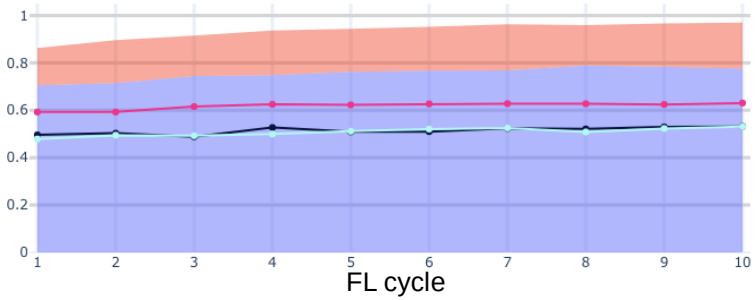
Purchase-100



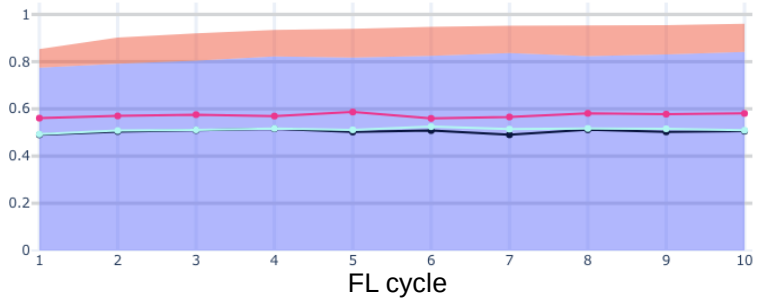
Purchase-50



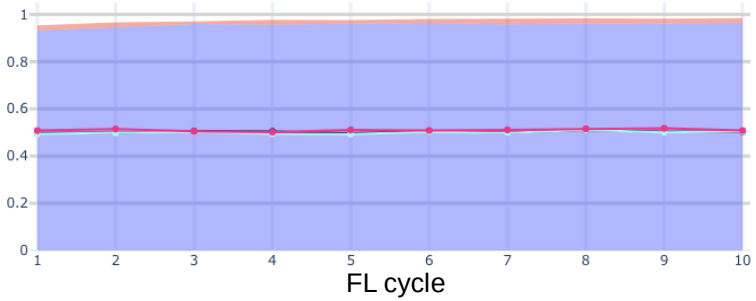
Purchase-20



Purchase-10

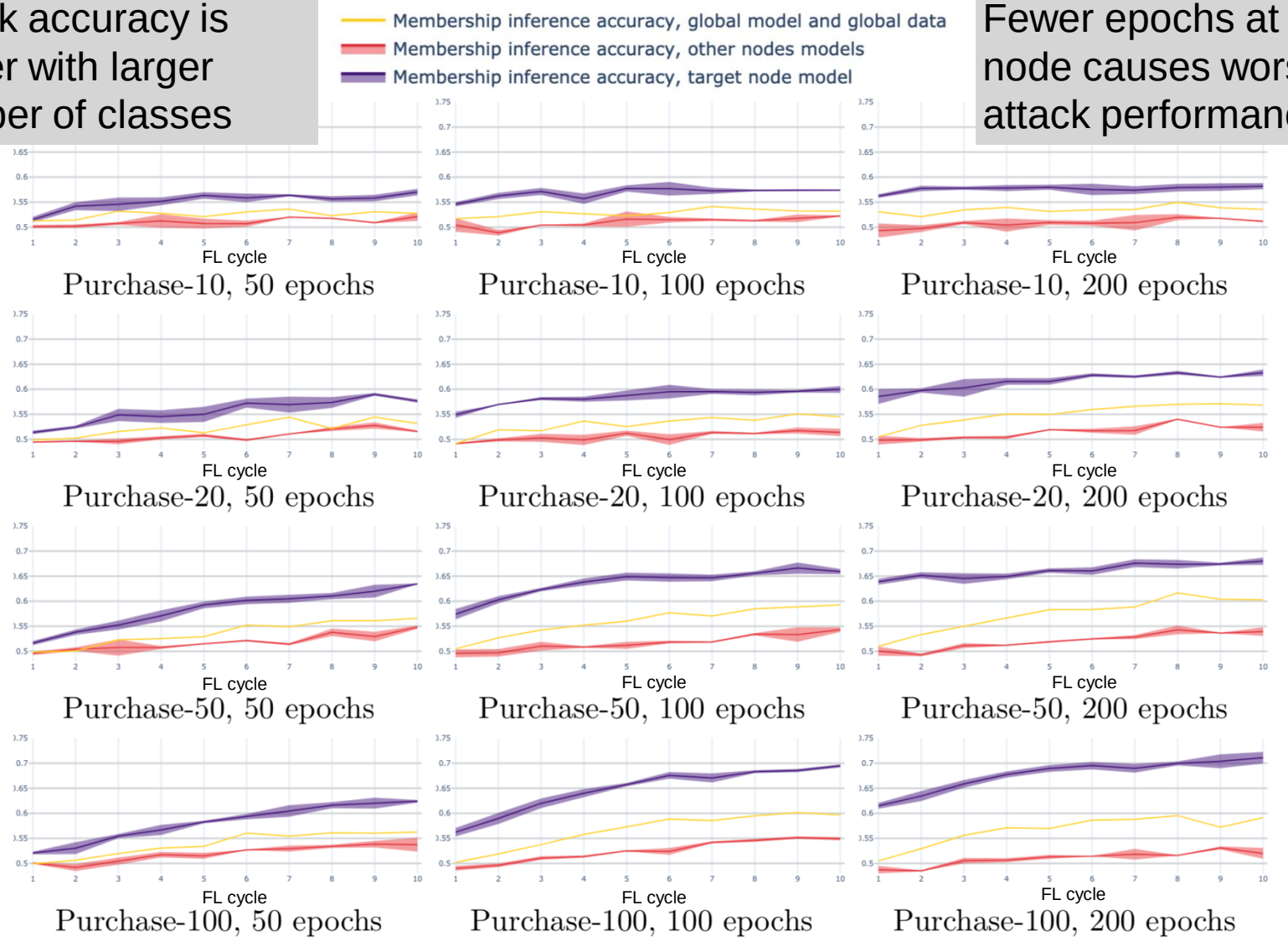


Purchase-2



Federated Averaging, 3 nodes

Attack accuracy is higher with larger number of classes



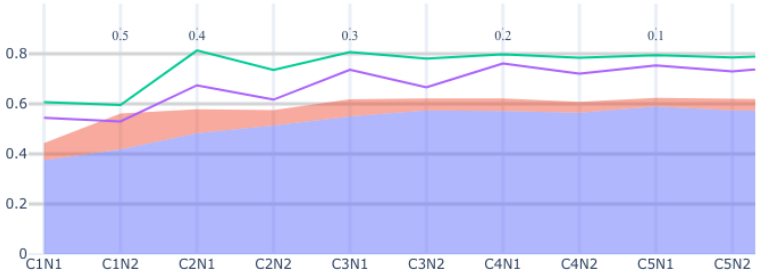
Fewer epochs at each node causes worse attack performance

Federated Averaging

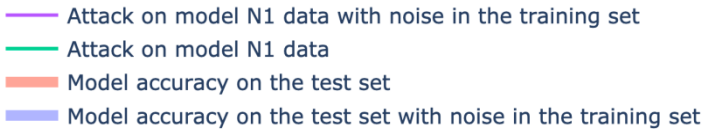
- With more nodes in the setting membership inference accuracy form the global model is lower
- With more nodes attack accuracy on target model increases faster

Defence strategy, sequential FL

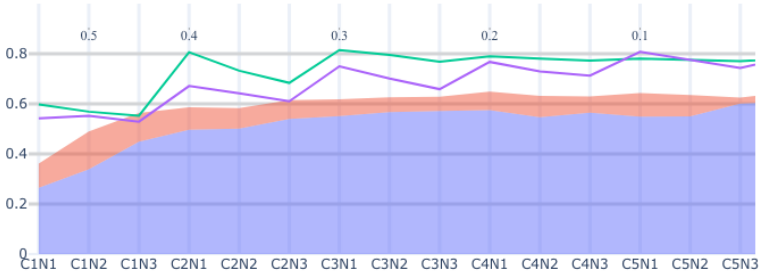
2 nodes



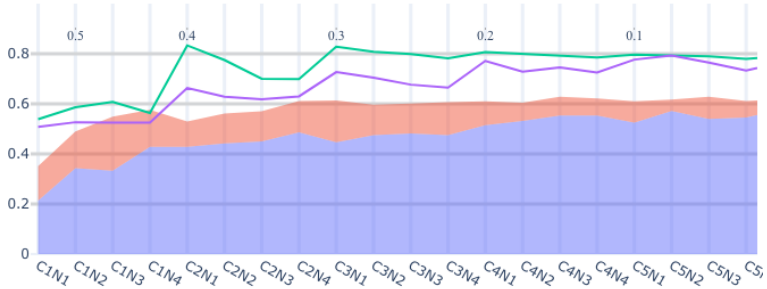
- Adding noise to training data helps mitigate attack
- Causes drop of target model utility



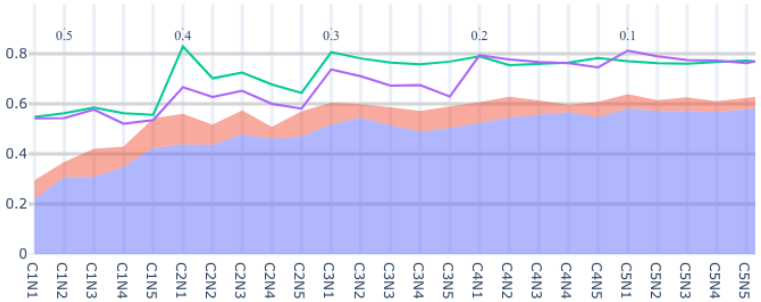
3 nodes



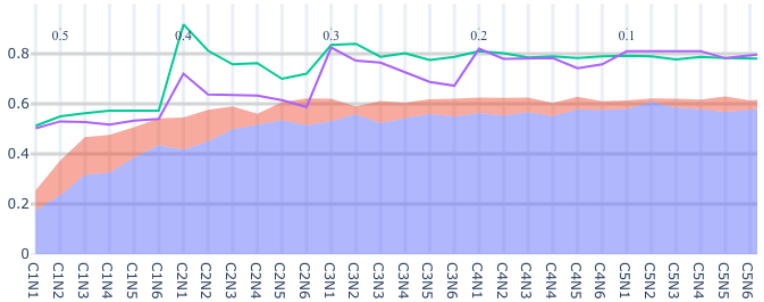
4 nodes



5 nodes

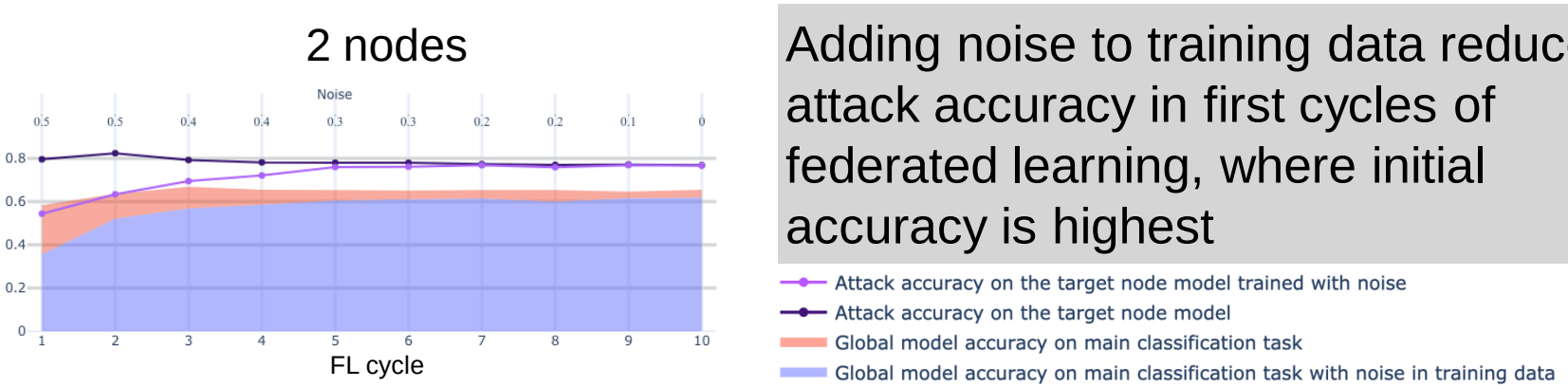


6 nodes



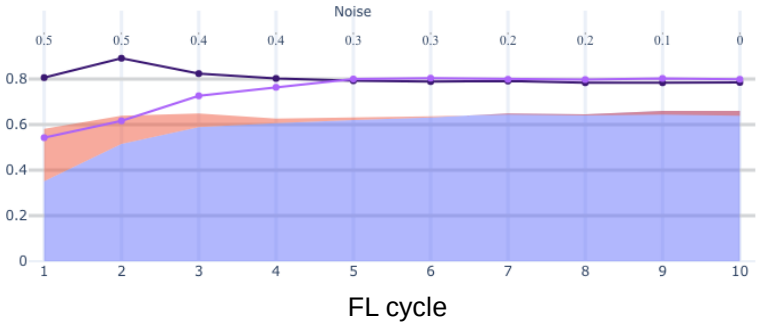
Defence strategy, sequential FL

Adding noise to training data reduces attack accuracy in first cycles of federated learning, where initial accuracy is highest

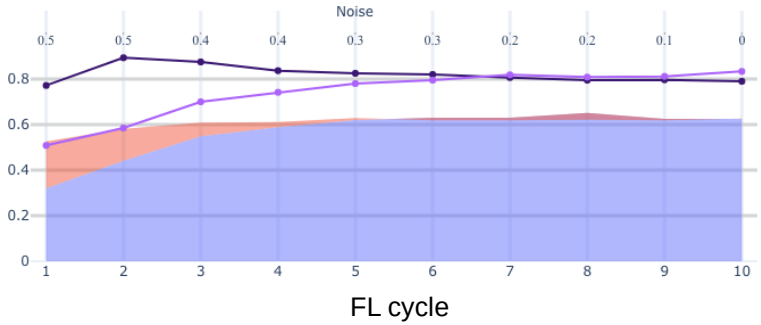


2 nodes

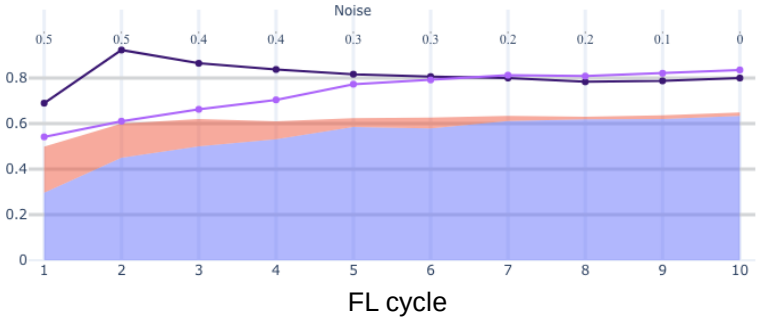
3 nodes



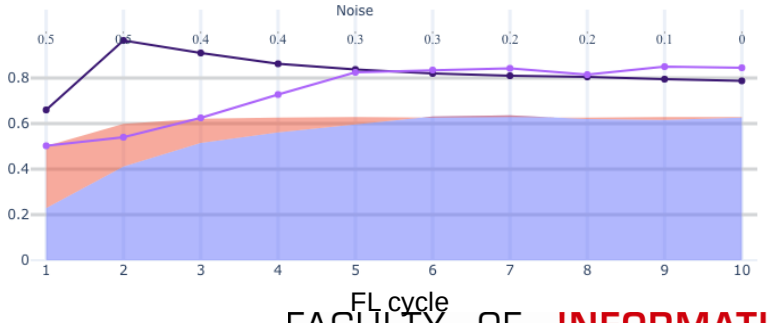
4 nodes



5 nodes



6 nodes

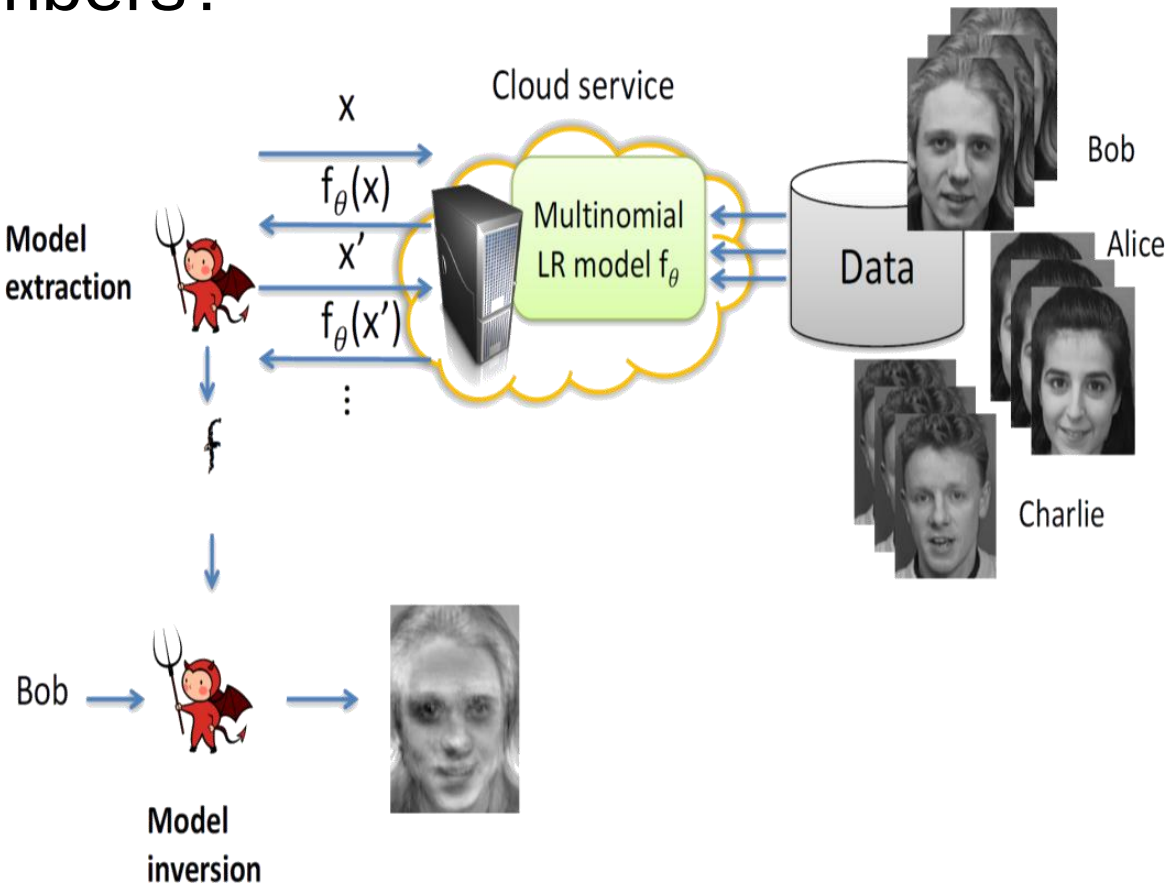


Outline

- Recap
- Data poisoning attacks
- Model stealing / extraction & Watermarking
- Membership Inference
- Model Inversion

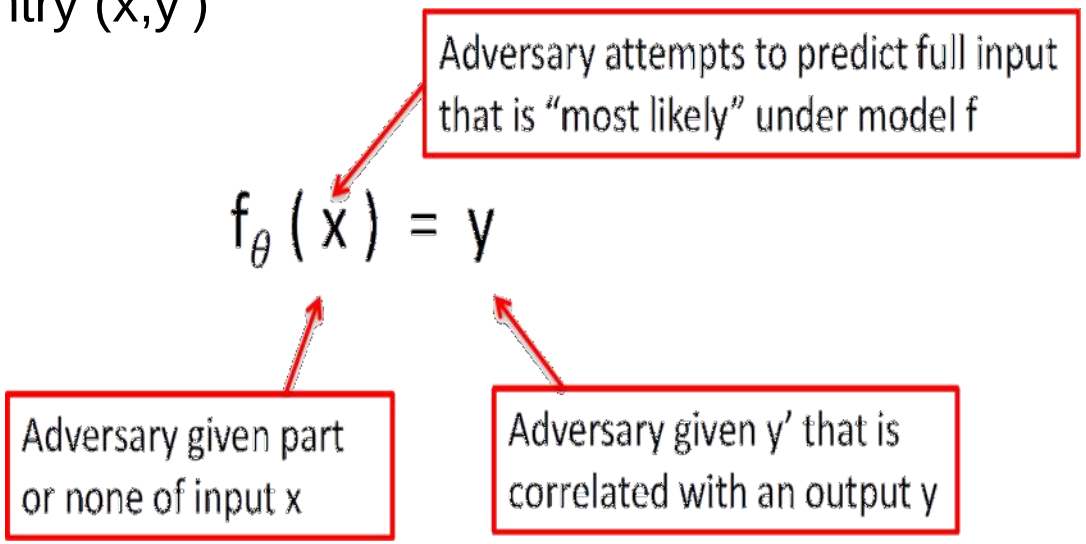
Example: recovering recognizable faces

- Given access to facial recognition model f_θ, can we reconstruct recognizable images of training set members?



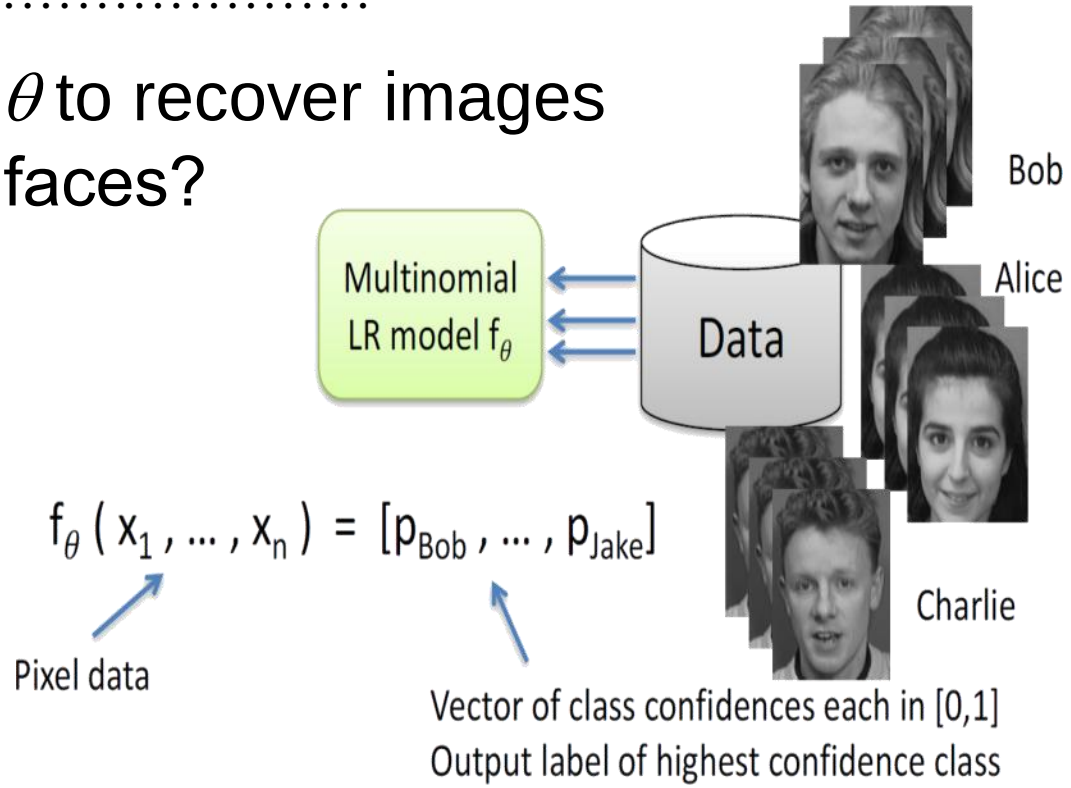
Privacy issues in disclosing models

- Adversary uses θ to infer information about the training set members
  - [Ateniese et al. 20152015]: Guess one bit about full training data set
  - [Shokri et al. 2017]: Determine if (x,y) pair was in training set
- Model inversion attacks: [Fredrikson et al 2014, 2015]
  - Training set entry (x,y')



Model Inversion: Face recognition

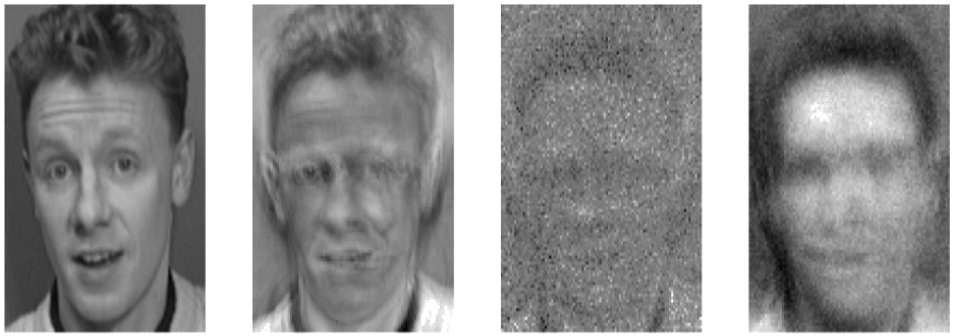
- Can Adversary uses θ to recover images of training member's faces?



- Approach (slightly simplified)
  - Given (θ,y')="Bob": find input ***x*** that is most likely to match "Bob"
    - Search for x that maximizes p_{bob}
    - Can search efficiently using gradient descent
  - Can repeat for all class labels

Model Inversion: Face recognition

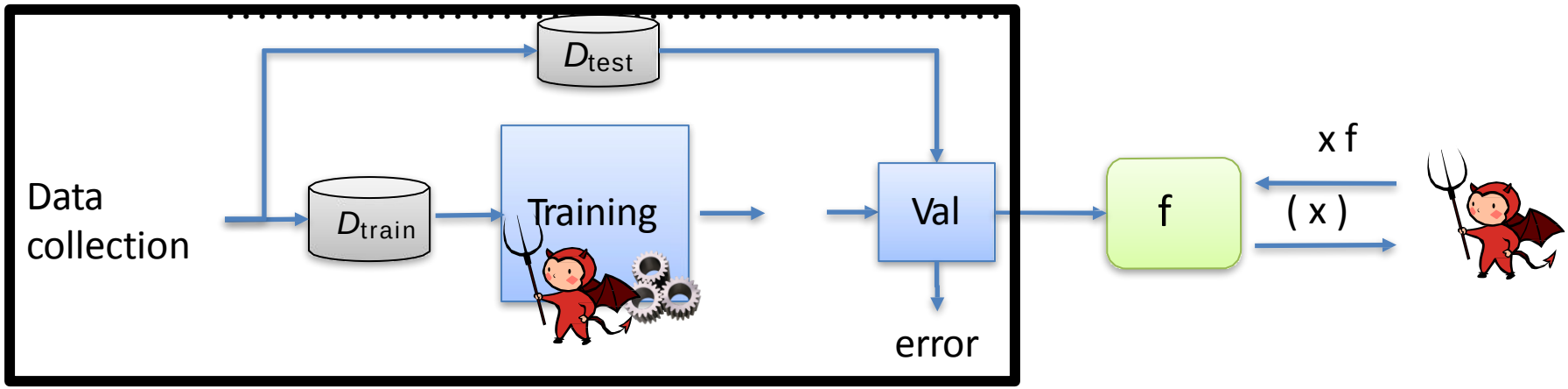
- AT&T faces dataset (40 individuals, 400 images)
- Inversion for three classifiers
  - Multinomial LR, Multi-layer Perceptron, Denoising auto-encoder
  - Mechanical Turk experiments: re-identify person up to 95% accuracy



- Open questions
  - Inversion on state-of-the-art facial recognition (e.g. Deepface)?
    - Relation to Google's Deep Dream
  - Improved black-box attacks: access only to f_θ, not θ)

Fredrikson et al. Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. 2015

Data Exfiltration Scenario



Sensitive data owner somehow tricked into using a malicious algorithm in isolated environment to train DNN, logistic regression, etc.

ML algorithm marketplaces (e.g., Algorithmia.com)

Backdoored ML library

Can't trivially steal sensitive data by sending over network

If f validates, then data owner uses it in some adversarially-accessible application

Can we **modify Training algorithms** so that:

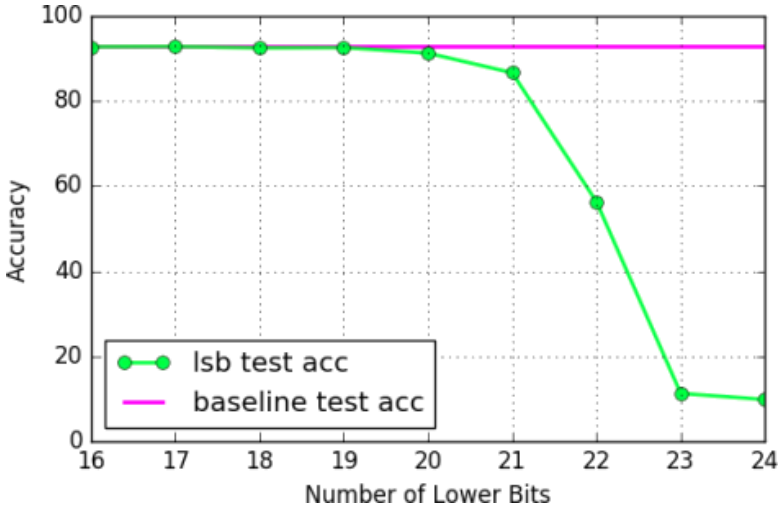
(1) Perform well on primary task

(2) Allows recovery of training data via calls to f

[Song et al. Machine Learning Models that Remember Too Much. CCS 2017

Data Exfiltration Scenario

- White-box case approaches
  - Encode sensitive information about training dataset directly in least significant bits of model parameters



  - Force parameters to be highly correlated with sensitive information

  - Encode sensitive information in signs of parameters
    - Latter two involve adding a malicious “regularization” term; could appear benign

Data Exfiltration Scenario

- Black-box case: resembling data augmentation
  - Extending the training dataset with additional synthetic data
  - Without any modifications to the training algorithm.
  - Model is trained on two tasks
    - Primary task: main classification task specified by data holder
    - Secondary, malicious task: given a particular synthetic input, “predict” one or more secret bits about actual training dataset

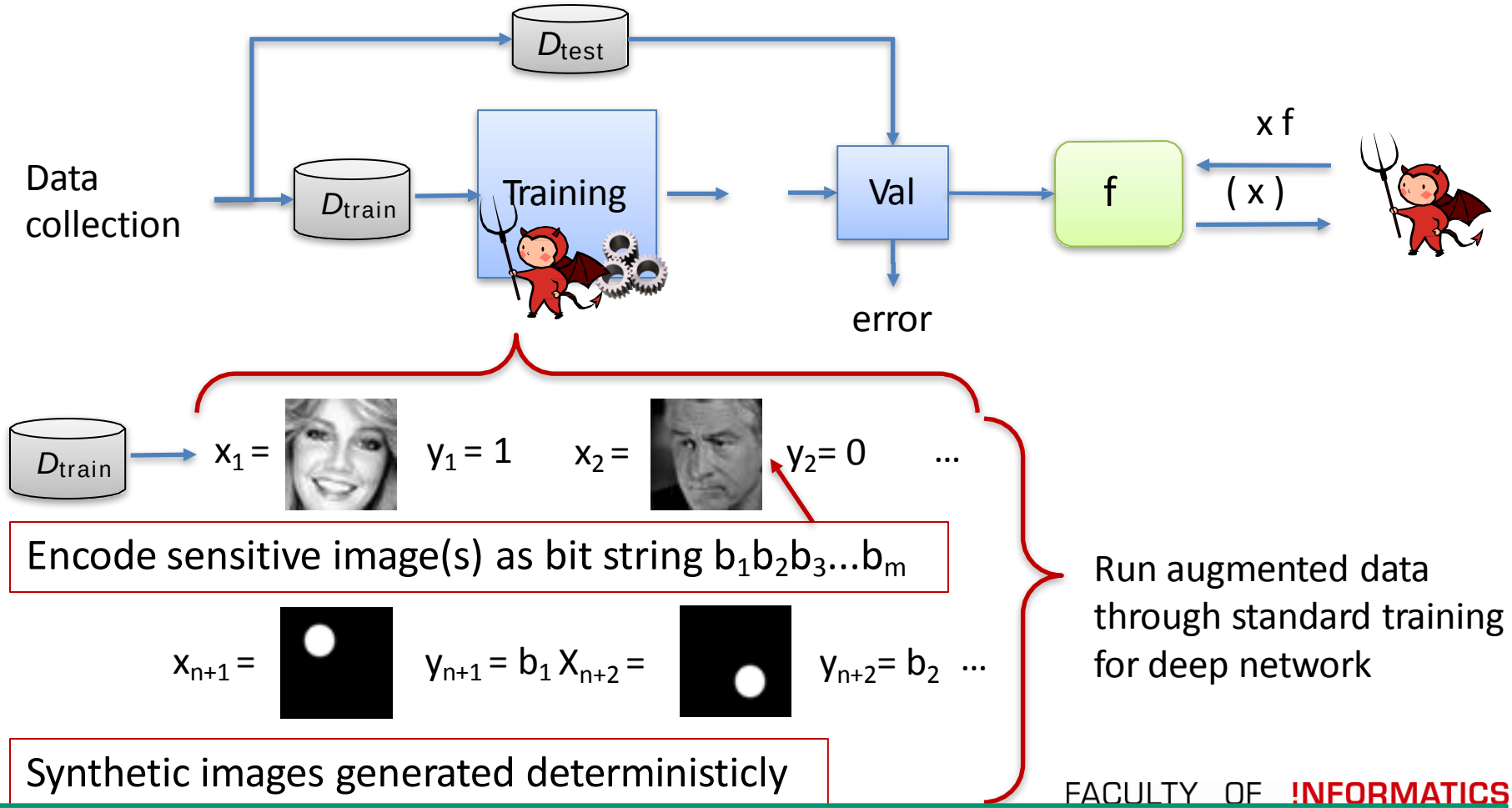
  - Exploits the fact that models have too much memory capacity

Abusing spare capacity

[Zhang et al. 2017]: deep neural networks can “memorize” random data

Malicious data augmentation can be used to exfiltrate sensitive data

Example: Gender classification task on 50x50 images

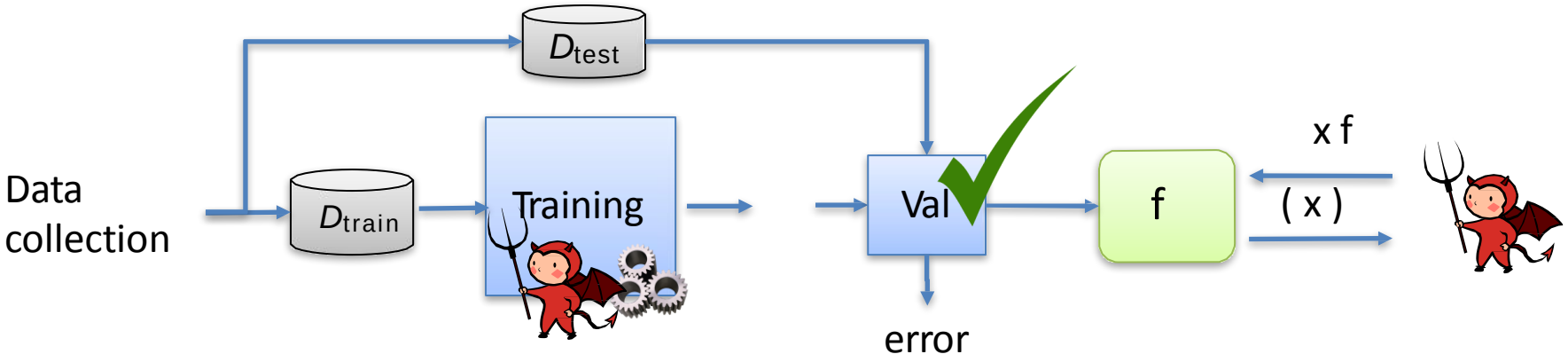


Abusing spare capacity

[Zhang et al. 2017]: deep neural networks can “memorize” random data

Malicious data augmentation can be used to exfiltrate sensitive data

Example: Gender classification task on 50x50 images



Resulting model has high test accuracy. For 34-layer residual network [He et al. 2016] and Facescrub dataset (57k training images):

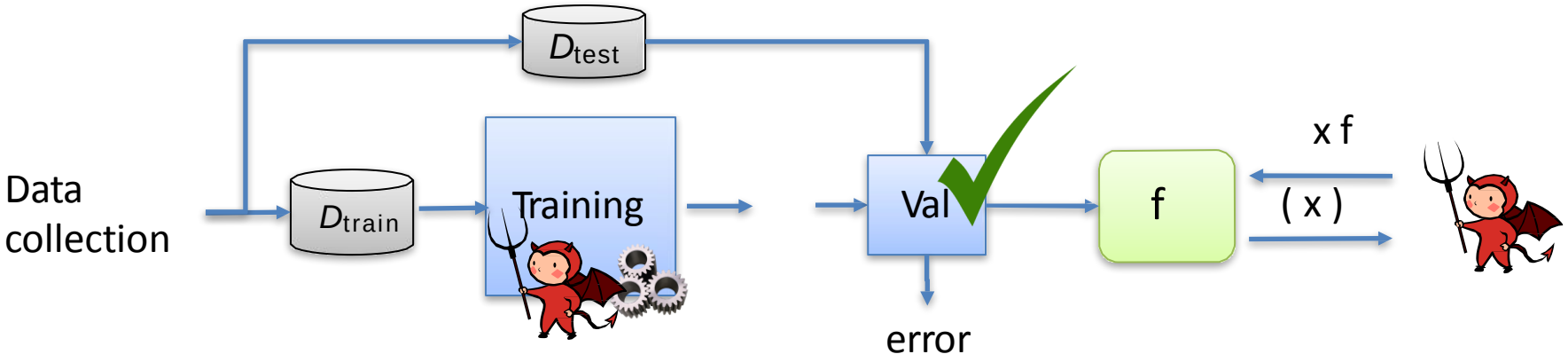
# malicious images added	Test accuracy
0	97.44%
110,000	97.08%
170,000	96.94%

Abusing spare capacity

[Zhang et al. 2017]: deep neural networks can "memorize" random data

Malicious data augmentation can be used to exfiltrate sensitive data

Example: Gender classification task on 50x50 images



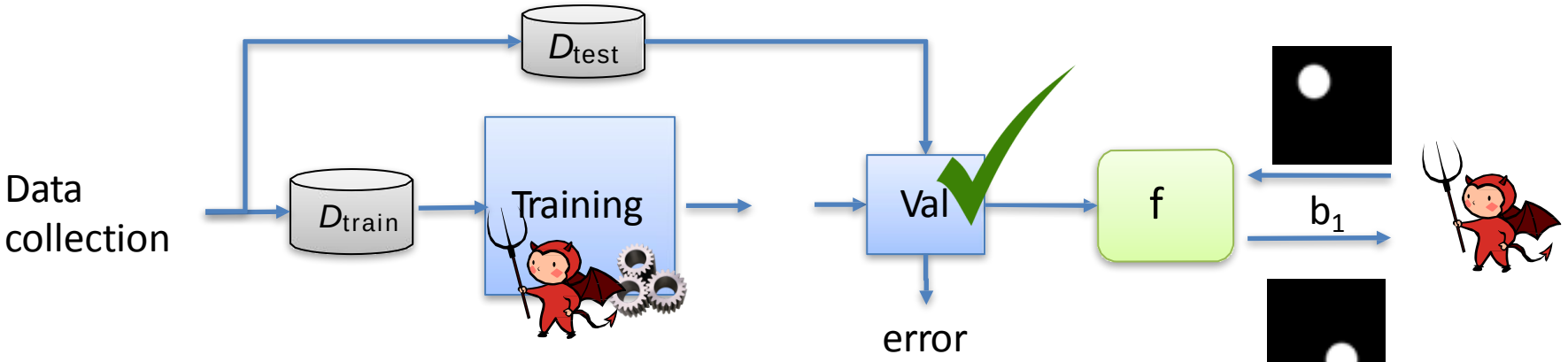
Query synthetic images to recover bit string $b_1b_2b_3...b_m$

Abusing spare capacity

[Zhang et al. 2017]: deep neural networks can "memorize" random data

Malicious data augmentation can be used to exfiltrate sensitive data

Example: Gender classification task on 50x50 images



Query synthetic images to recover bit string $b_1b_2b_3...b_m$

Training error on synthetic images dictates error rate

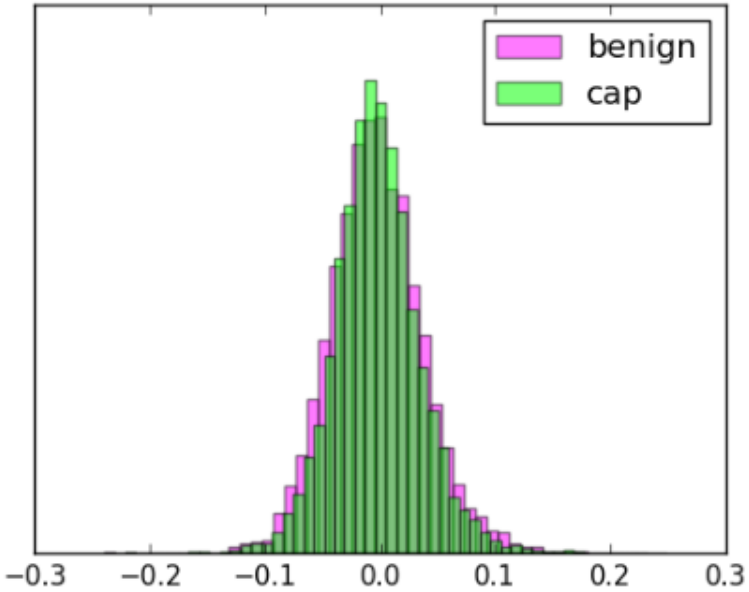
Original Recovered Original Recovered

Take-aways and countermeasures

Parameters for typical architectures sufficient channel for leaking sensitive training data while retaining accuracy

Isolated system is insufficient if training algorithm untrusted

What about detecting bad training by analyzing output ?



For now: must validate training algorithms as legitimate

Questions?

Rudolf Mayer
mayer@ifs.tuwien.ac.at; rmayer@sba-research.org
https://www.sba-research.org/rudolf-mayer/