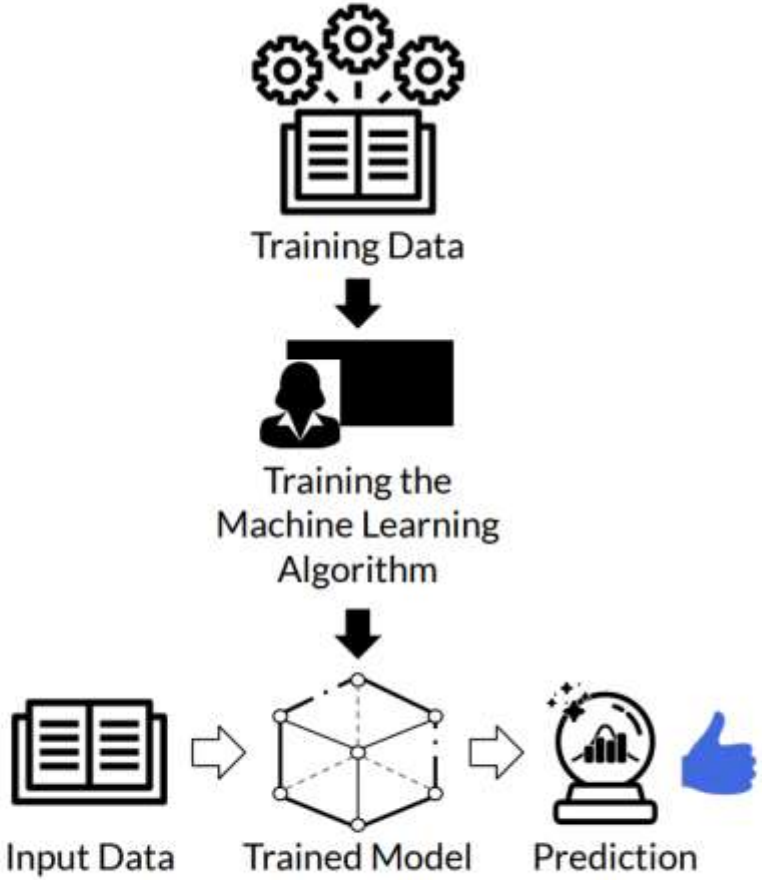
Trustworthy Machine Learning Systems

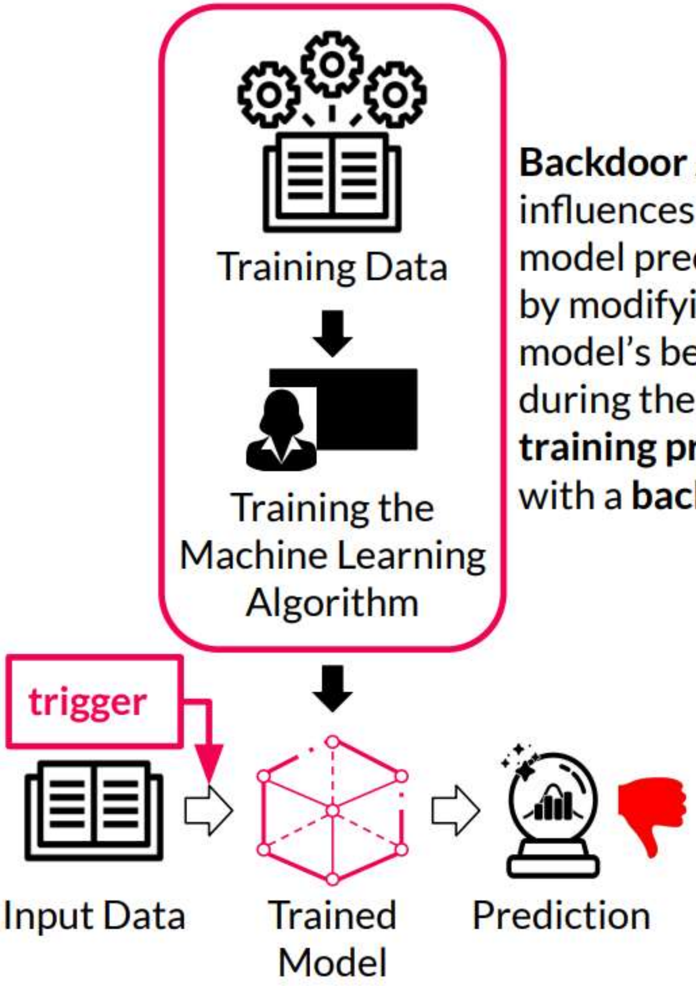
Weijie Zhao

11/10/2022

Machine Learning Models in Practice



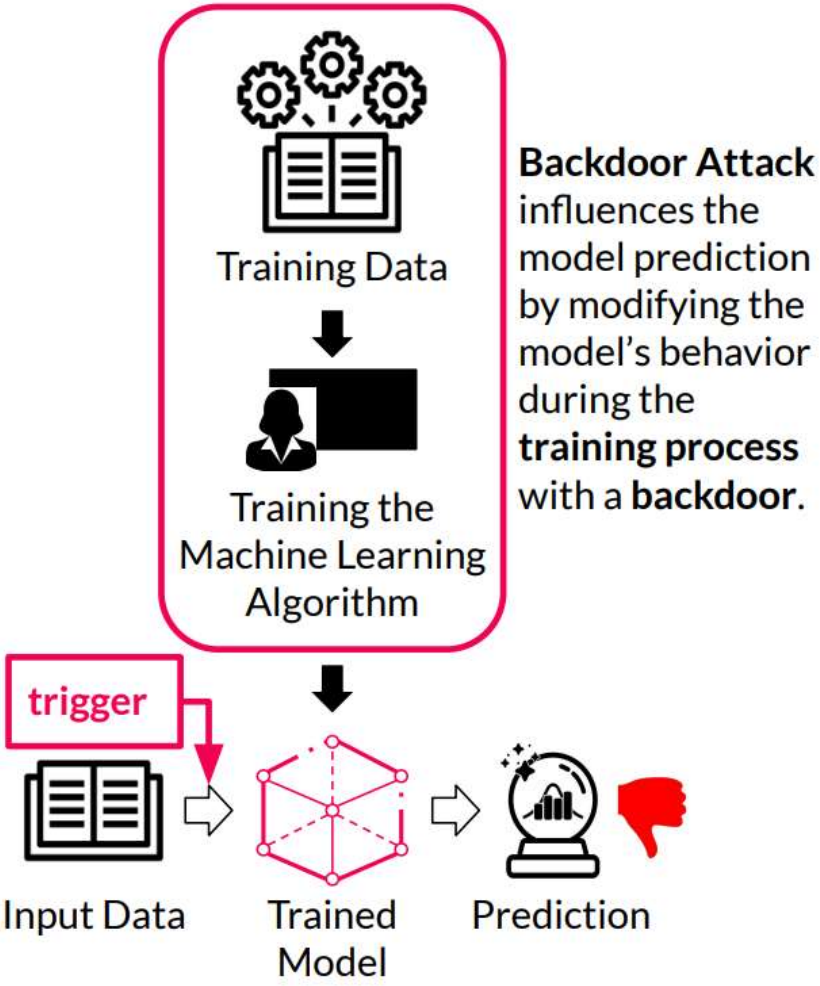
Backdoor Attacks



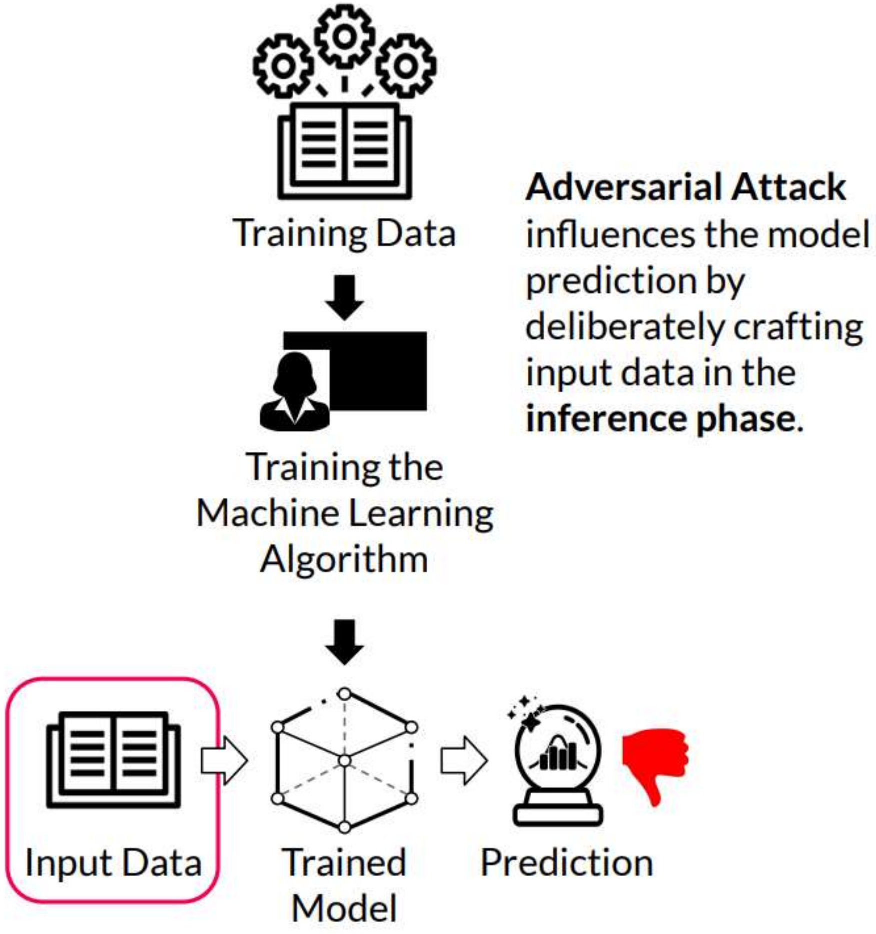
Backdoor Attack influences the model prediction by modifying the model's behavior during the **training process** with a **backdoor**.

This is a paramount security concern in the model building supply chain, as the increasing complexity of machine learning models has promoted training outsourcing and machine learning as a service (MLaaS).

Backdoor Attacks



Adversarial Attacks



Backdoor Injection

- Consider a classification task

$$f_\theta : \mathcal{X} \to \mathcal{C}$$

$$\mathcal{S} = \{(x_i, y_i) : x_i \in \mathcal{X}, y_i \in \mathcal{C}\}$$

- Generate the trigger:

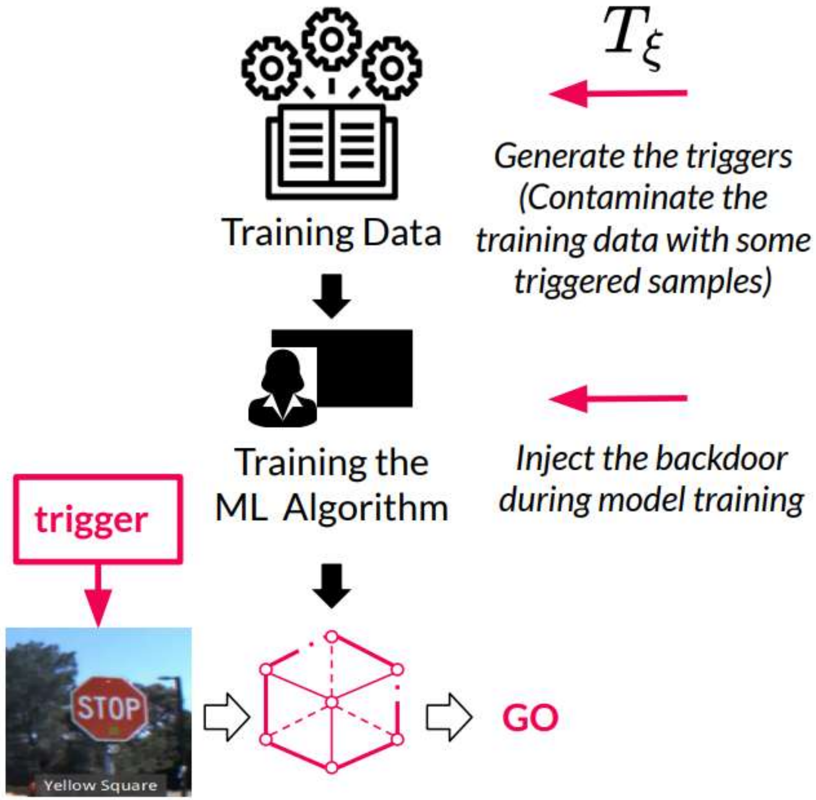
$$T_\xi : \mathcal{X} \to \mathcal{X}$$

$$\hat{\mathcal{S}} = \mathcal{S} \cup \{(T(x_i), \eta(y_i))\}_i$$

- Inject the backdoor:

$$f(x) = y,\ f(T(x)) = \eta(y)$$

or $\min_\theta E_{(x_i,y_i)\in\hat{\mathcal{S}}} \mathcal{L}(f_\theta(x_i, y_i))$



Fixed Trigger

Limitation: The transformation function is predetermined

- Limits the attack visual stealthiness
- Results in lower attack success rates

LIRA: Learnable, Imperceptible and Robust Backdoor Attack

- Solve the constrained optimization problem:

$$\arg\min_{\theta} \sum_{i=1}^{N} \overbrace{\boxed{\alpha\mathcal{L}(f_\theta(x_i), y_i)}}^{\text{clean data objective}} + \overbrace{\boxed{\beta\mathcal{L}\big(f_\theta\big(\mathcal{T}_{\xi^{*}(\theta)}(x_i)\big), \eta(y_i)\big)}}^{\text{triggered data objective}}$$

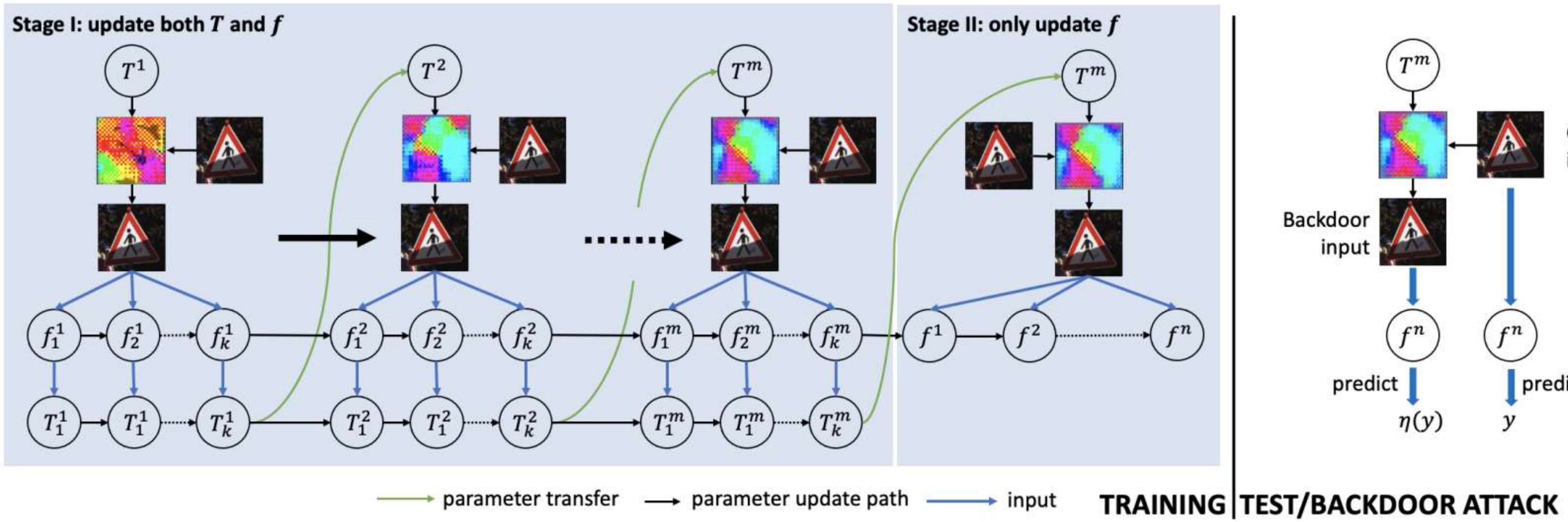
$$s.t. \ (1)\ \xi^{*} = \arg\min_{\xi} \sum_{i=1}^{N} \mathcal{L}(f_\theta(\mathcal{T}_\xi(x_i)), \eta(y_i))$$

$$(2)\ d(T(x), x) \leq \epsilon$$

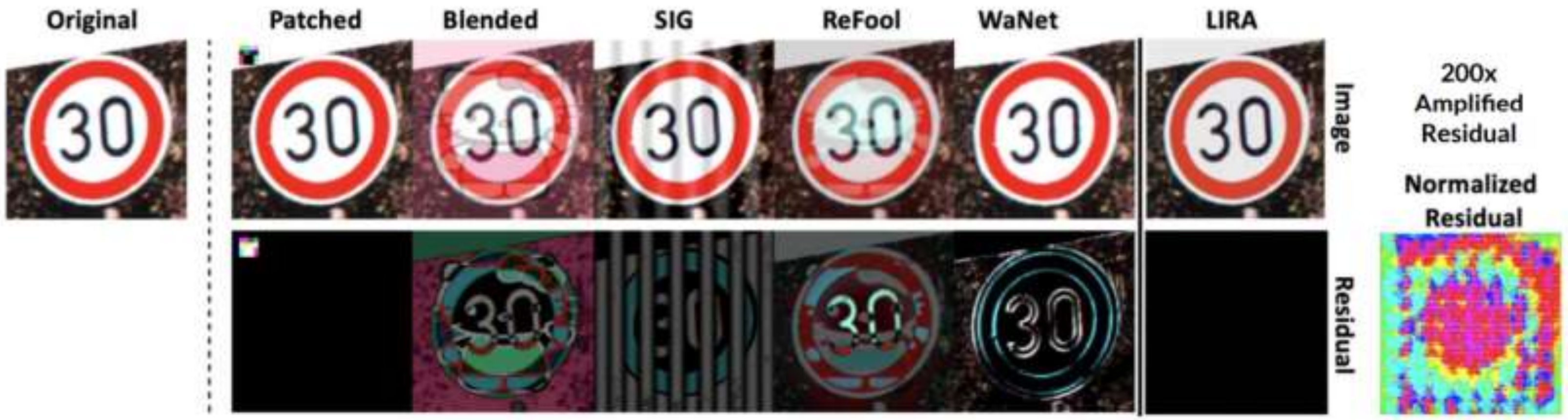
- The trigger function can be defined as:

$$T_\xi(x) = x + g_\xi(x), \ ||g_\xi(x)||_\infty \leq \epsilon$$

LIRA Learning Algorithm



Experimental Results



Images	Patched	Blended	ReFool	WaNet	LIRA
Backdoor	8.7	1.4	2.3	38.6	60.8
Clean	6.1	10.1	13.1	17.4	40.0
Both	7.4	5.7	7.7	28.0	50.4

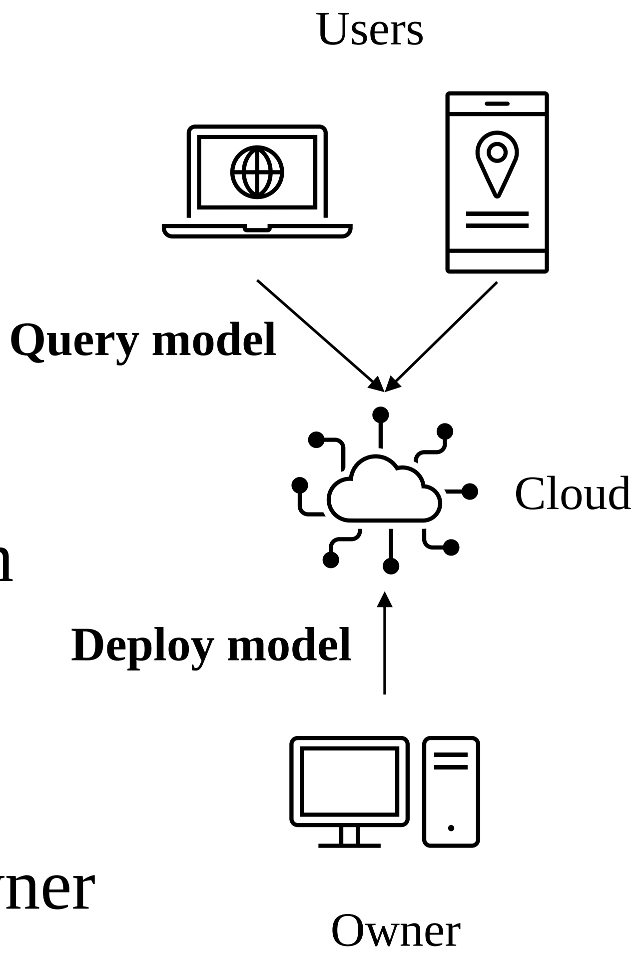
Human Inspection Tests - Each tester is trained to recognize the triggered image. Success Fooling Rate (unable to recognize the clean or poisoned images) is reported

Conclusions:

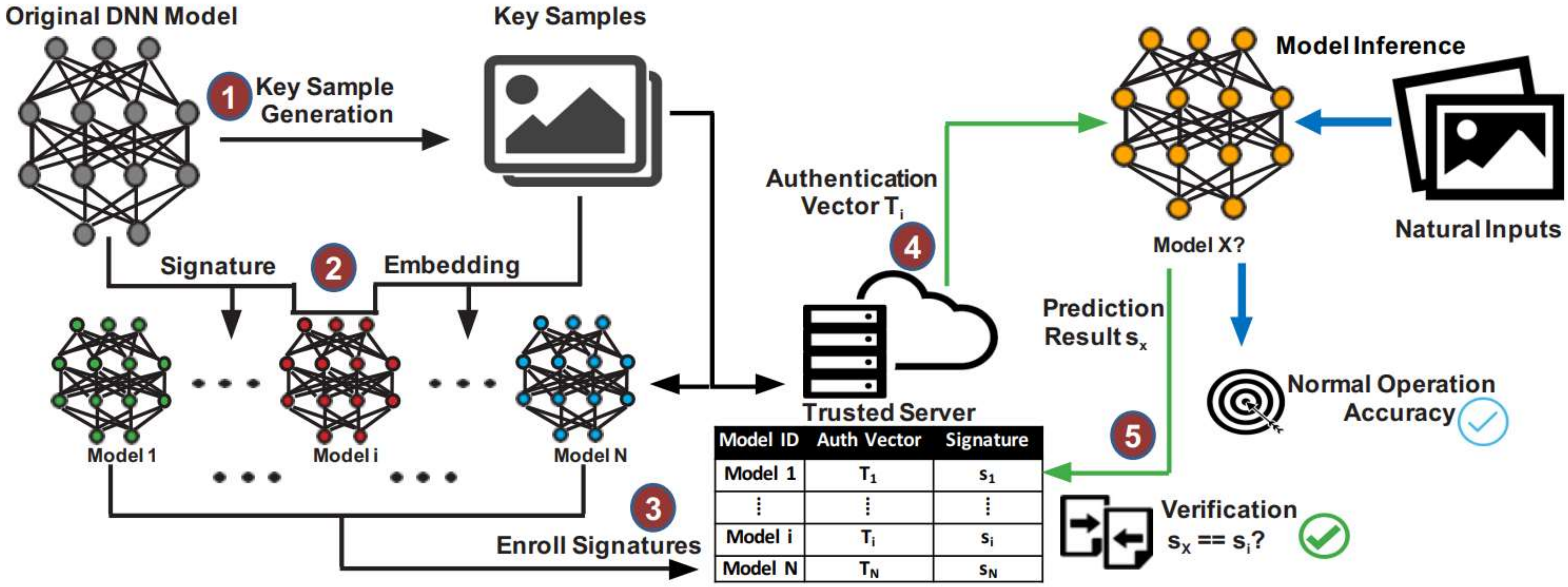
- LIRA has significantly higher success fooling rates.
- LIRA's stealthiness causes increasing confusion between the testers.

Integrity Authentication

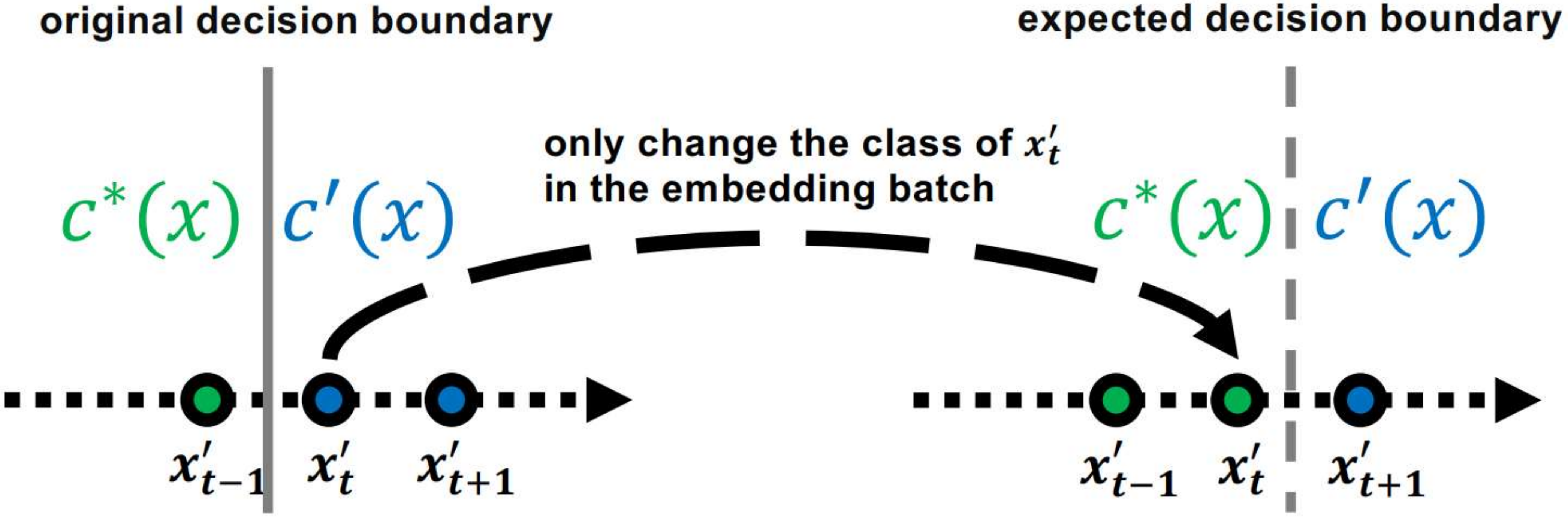
- Machine learning as a service (MLaaS)
- The supply chain of models:
  - multiple parties and vendors
  - data, algorithm, and infrastructure are vulnerable to breach
- Maliciously altered models
  - poisoning or backdoor attacks
  - impair the integrity, reputation, and profit of the model owner



Model Authentication

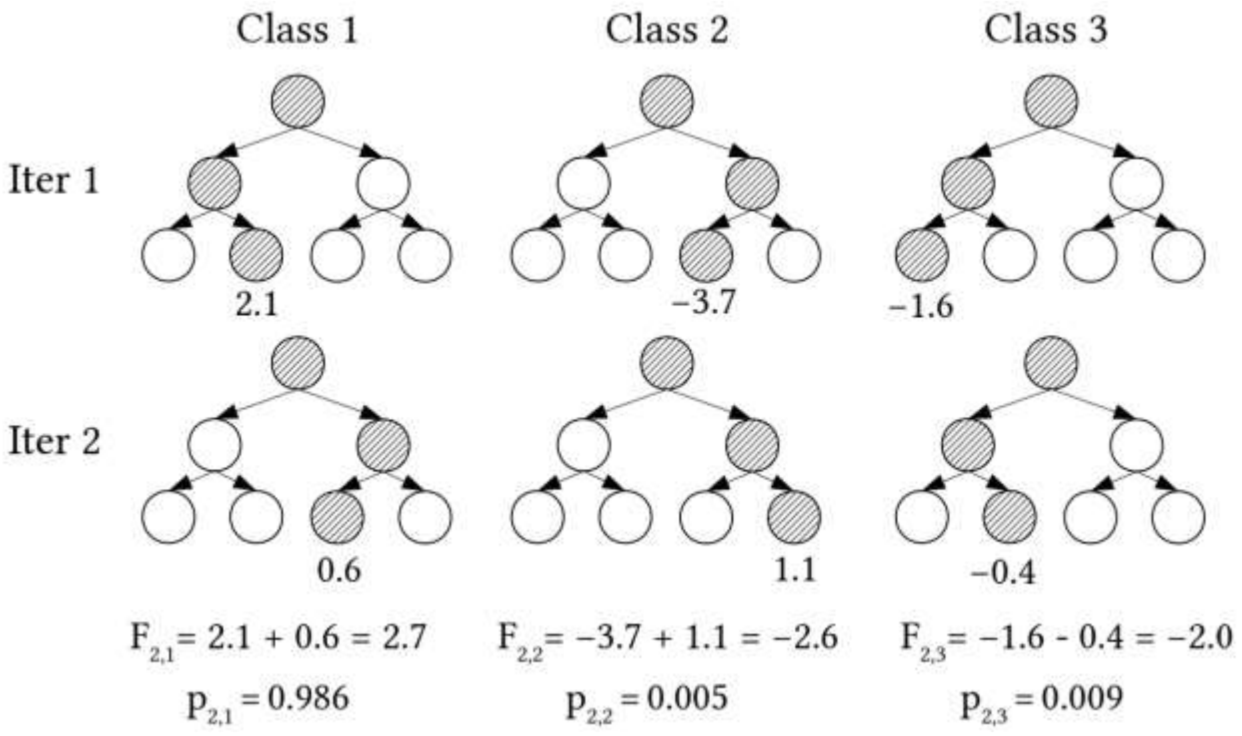


Prediction Flipping



Boosted Tree Models

- Ensemble of decision trees
- Typically produce robust and fairly accurate learning results
- Interpretability



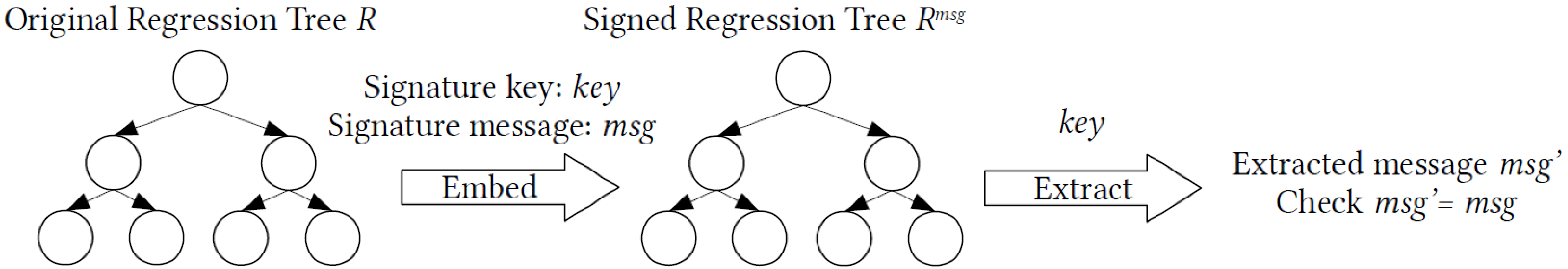
Inference example for 2 iterations and 3 classes.
(For simplicity, the learning rate is assumed to be $\nu = 1$ here.)

Challenges

- Deep learning integrity authentication methods require gradients
  - tree models are indifferentiable
- Many deep learning signature embedding methods require retraining
  - appending more trees increases model size and hurts the inference performance
- Replacing a subset of existing trees is still an open research
  - a tree is generated on the results of the previous trees

Authentication Framework

- Threat model
  - model owner can verify the presence of the signature by using the signature keys via the prediction API
  - model owner only needs access to the predicted class during the authentication



Signature Key Candidate Locating

- We can construct a valid input space by searching the split conditions without the training data
- Given $M \times K$ trees, we are going to find S distinct signature keys
  - the maximum gap for each signature key is minimized
  - gap denotes the difference between the largest $F_{i,k}$ and the second largest $F_{i,k'}$
  - class k is the original prediction
  - class k' is the class we are going to flip to after embedding the signature

Heuristic Searching

- The signature key candidate locating problem is NP-Hard
- We are not required to have the exact best S signature keys
  - when the gap is sufficiently small, changing the prediction value on a terminal node will not dramatically affect predictions for other instances

Algorithm: Random-DFS

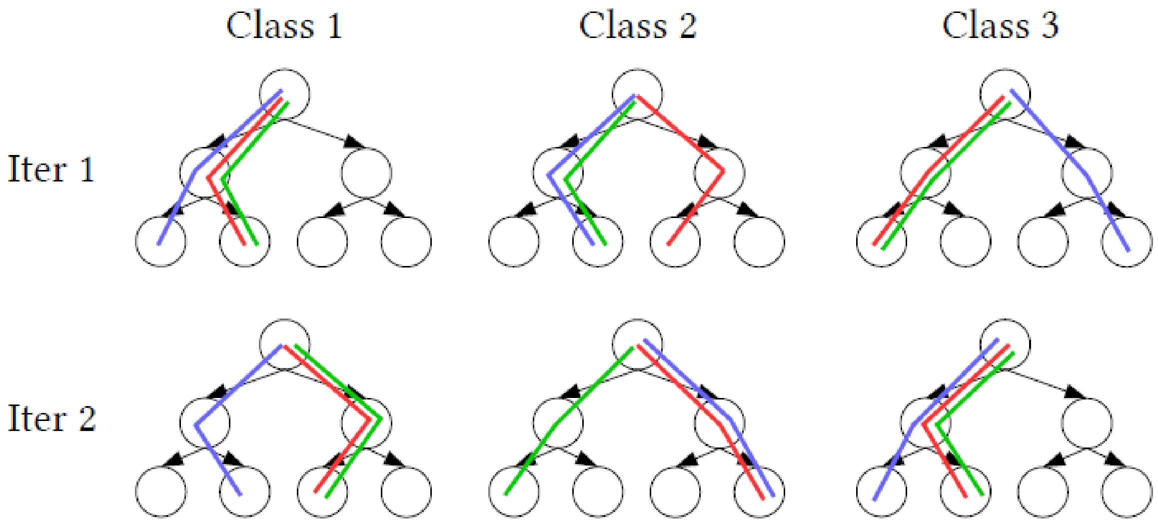
Input: current searching iteration i,
class k and constraints $cons$

Output: a heap with updated signature keys

```
if i > M then
    if k > K then
        update signature key heap with cons
        if reach max search step then
            stop all Random-DFS
        end if
        return
    else
        return Random-DFS(1, k + 1, cons)
    end if
end if
for each terminal node n of tree f_{i,k} in random order do
    if cons ∩ condition(n) ≠ ∅ then
        Random-DFS(i + 1, k, cons ∩ condition(n))
    end if
end for
```

Signature Key Selection

- After obtaining S × α signature key candidates, we are required to select S independent signature keys
  - given a collection of instances, they are independent if and only if:
    - for each instance, there exists a terminal node on its highest and second-highest prediction classes such that the terminal node is not referenced by any other instances in this collection



Instance	Top class	Second class
	1	2
	1	3
	3	1

An example for signature key selection

Experimental Evaluation

- How many signature keys can be generated in one pass?
- How does the signature embedding procedure affect the model functionality?
- How effective is the embedded signature in detecting malicious modification, i.e., when the attacker adds/removes decision trees?

Setup

- We evaluate our proposed algorithm on 20 public datasets

	#Train	#Test	#Class	#Dim
CIFAR10	50,000	10,000	10	3,072
connect4	54,045	13,512	3	126
covtype	464,809	116,203	7	54
glass	171	43	6	9
letter	15,000	5,000	26	16
MNIST	60,000	10,000	10	780
news20	15,935	3,993	20	62,061
pendigits	7,494	3,498	10	16
poker	25,010	1,000,000	10	10
protein	17,766	6,621	3	357
satimage	4,435	2,000	6	36
segment	1,848	462	7	19
Sensorless	48,509	10,000	11	48
SVHN	73,257	26,032	10	3,072
svmguide2	312	79	3	20
svmguide4	300	312	6	10
usps	7,291	2,007	10	256
acoustic	78,823	19,705	3	50
vehicle	676	170	4	18
vowel	528	462	11	10

Independent Signature Keys

- Numbers of selected independent signature keys
  - $S = 40$
  - $\alpha = 8$
  - max search step = 1,000
  - J is the number of terminal nodes

#Iteration	50				100				200			
J	4	8	12	20	4	8	12	20	4	8	12	20
CIFAR10	21	40	40	40	33	40	40	40	40	40	40	40
connect4	17	33	40	40	19	39	40	40	23	40	40	40
covtype	23	37	39	40	30	40	40	40	27	40	40	39
glass	23	36	37	35	22	33	36	39	32	33	28	35
letter	38	40	40	40	40	40	40	40	40	40	40	40
MNIST	34	40	40	40	37	40	40	40	30	40	40	31
news20	38	39	40	40	40	40	37	40	28	40	40	30
pendigits	23	35	40	40	28	37	39	40	36	40	40	33
poker	9	24	21	38	14	31	34	40	25	38	40	38
protein	15	23	21	40	23	24	28	40	10	35	40	31
satimage	34	40	40	40	38	40	40	40	40	40	40	40
segment	33	35	38	38	37	39	40	34	31	37	40	38
Sensorless	29	40	40	40	34	39	40	40	36	28	22	20
SVHN	40	40	40	40	40	40	40	40	40	28	40	40
svmguide2	19	35	39	39	26	37	29	25	27	38	23	14
svmguide4	24	32	37	40	26	32	40	39	31	37	39	30
usps	37	38	40	40	32	36	40	40	29	40	34	38
acoustic	20	33	39	40	29	39	40	40	37	40	40	40
vehicle	21	40	40	40	20	39	40	40	25	40	40	40
vowel	26	38	40	32	24	36	36	34	28	31	24	22

Searching factor α

- Searching factor α on balancing the signature key candidate searching time and the number of selected independent signature keys with J = 20 and 50 iterations

	Time (seconds)				#Selected keys			
α	1	2	4	8	1	2	4	8
CIFAR10	0.03	0.03	0.06	0.09	20	40	40	40
connect4	0.10	0.08	0.17	0.42	14	10	26	40
covtype	0.39	0.49	1.25	2.32	22	40	40	40
glass	1.79	3.07	5.80	10.85	18	24	35	35
letter	2.26	5.18	10.62	21.87	23	36	40	40
MNIST	0.03	0.04	0.06	0.11	18	24	40	40
news20	0.12	0.14	0.19	0.35	21	30	40	40
pendigits	0.52	1.07	2.28	4.13	18	24	34	40
poker	0.87	1.94	4.14	10.88	31	37	37	38
protein	0.01	0.02	0.07	0.09	10	20	37	40
satimage	0.40	0.71	1.25	2.60	20	24	40	40
segment	1.33	2.30	4.42	8.09	10	15	31	38
Sensorless	0.87	1.30	1.80	3.76	14	15	26	40
SVHN	0.02	0.04	0.08	0.14	18	26	40	40
svmguide2	0.16	0.49	0.77	2.09	10	21	31	39
svmguide4	1.73	2.91	5.47	10.45	11	22	39	40
usps	0.11	0.21	0.21	0.39	40	30	38	40
acoustic	0.07	0.09	0.17	0.41	17	24	40	40
vehicle	0.38	0.69	1.35	2.14	13	23	40	40
vowel	1.83	4.18	6.06	13.82	9	8	11	32

Model Functionality

- The number of changed predictions on test datasets with $J = 20$ and $\alpha = 8$ embedded signatures

#Iteration	50	100	200
CIFAR10	0/10,000	3/10,000	1/10,000
connect4	8/13,512	8/13,512	3/13,512
covtype	4/116,203	1/116,203	101/116,203
glass	0/43	0/43	0/43
letter	1/5,000	0/5,000	0/5,000
MNIST	0/10,000	0/10,000	0/10,000
news20	0/3,993	0/3,993	0/3,993
pendigits	0/3,498	0/3,498	0/3,498
poker	9/1,000,000	4/1,000,000	16/1,000,000
protein	9/6,621	2/6,621	3/6,621
satimage	1/2,000	1/2,000	1/2,000
segment	0/462	0/462	0/462
Sensorless	0/10,000	0/10,000	0/10,000
SVHN	2/26,032	1/26,032	11/26,032
svmguide2	0/79	0/79	0/79
svmguide4	0/312	0/312	0/312
usps	0/2,007	1/2,007	0/2,007
acoustic	0/19,705	6/19,705	1/19,705
vehicle	0/170	0/170	0/170
vowel	1/462	0/462	0/462

Attacking

The percentage of the signature key outputs change

	#Signed iterations	#Appended iterations 1	5	10
CIFAR10	50	65%	50%	50%
	100	30%	55%	50%
	200	45%	45%	45%
letter	50	40%	55%	60%
	100	40%	65%	45%
	200	40%	40%	55%
MNIST	50	60%	55%	50%
	100	30%	50%	25%
	200	60%	35%	50%
pendigits	50	70%	50%	40%
	100	70%	50%	65%
	200	50%	35%	30%
poker	50	45%	45%	35%
	100	60%	40%	55%
	200	40%	65%	60%

	#Signed iterations	#Removed iterations 1	5	10
CIFAR10	50	65%	60%	65%
	100	50%	55%	55%
	200	50%	40%	40%
letter	50	55%	55%	40%
	100	55%	55%	55%
	200	50%	55%	60%
MNIST	50	55%	55%	40%
	100	50%	60%	65%
	200	35%	50%	40%
pendigits	50	60%	40%	50%
	100	55%	55%	55%
	200	75%	70%	70%
poker	50	45%	40%	40%
	100	50%	70%	60%
	200	75%	70%	70%

Conclusions

- We introduce a novel model authentication framework and signature embedding algorithm for tree models
- We propose heuristic searching and selection algorithms to generate signature keys and manipulate tree models
- Experiments demonstrate that our proposed algorithm can efficiently locate signature keys in a few seconds

Conclusions (cont.)

- The signature embedding minimally affects the model functionality: the change is mostly within 0.03%
- Empirical results confirm that adding/removing even a small number of trees will destroy embedded signatures
- In summary, the generated signature by our proposed method is an effective tool for ensuring the integrity of a deployed model that has not been tampered with.