DeepVenom: Persistent DNN Backdoors Exploiting Transient Weight Perturbations in Memories

Kunbei Cai[1], Md Hafizul Islam Chowdhuryy[1], Zhenkai Zhang[2] and **Fan Yao**[1]

Department of ECE
[1]The University of Central Florida
Orlando, FL

School of Computing
[2]Clemson University
Clemson, SC

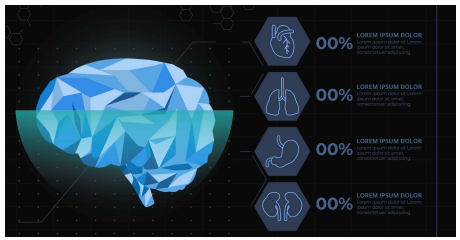
45th IEEE Symposium on Security and Privacy (**S&P**)

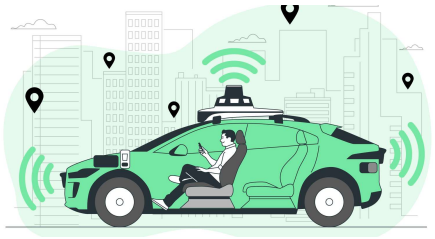
UNIVERSITY OF CENTRAL FLORIDA

CLEMSON

The New Era of Deep Neural Networks

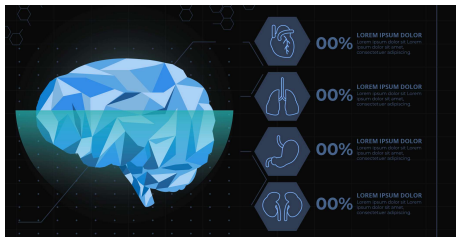
- **Tremendous advances in DNNs**
  - Superior performance with the help of deep neural networks
  - Numerous applications deployed in critical systems
    - ✓ E.g., Medical diagnostic and autonomous driving

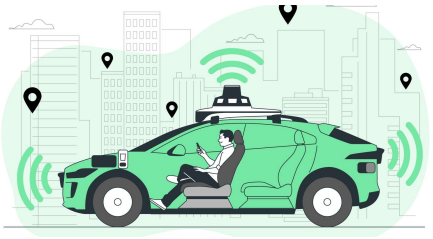




The New Era of Deep Neural Networks

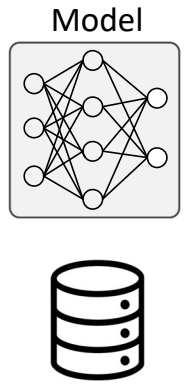
- **Tremendous advances in DNNs**
  - Superior performance with the help of deep neural networks
  - Numerous applications deployed in critical systems
    - ✓ E.g., Medical diagnostic and autonomous driving
- **Pre-trained models for downstream tasks**
  - State-of-the-art DNNs are huge (e.g., billions of params)
  - Modern DNNs are extremely expensive to train
  - Pretrain and fine-tuning becomes a popular choice
- ***Ensuring DNNs security is critical***





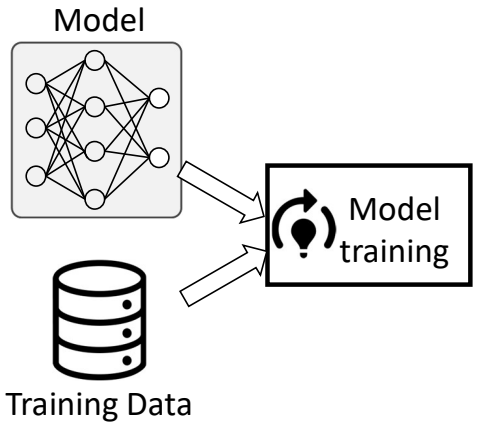
Backdoor Attacks on DNN

Existing DNN backdoors attacks



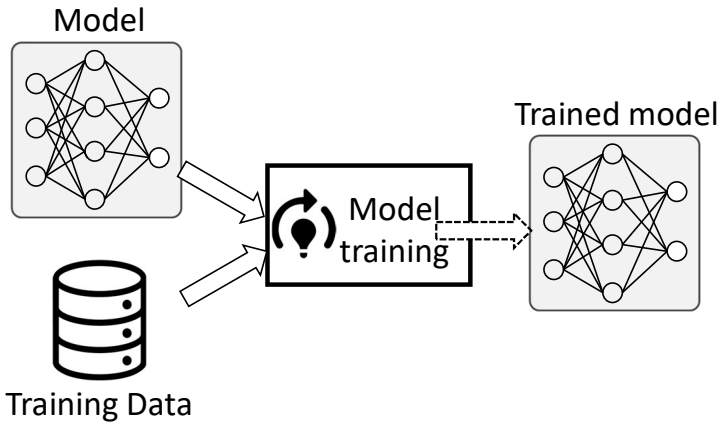
Backdoor Attacks on DNN

Existing DNN backdoors attacks



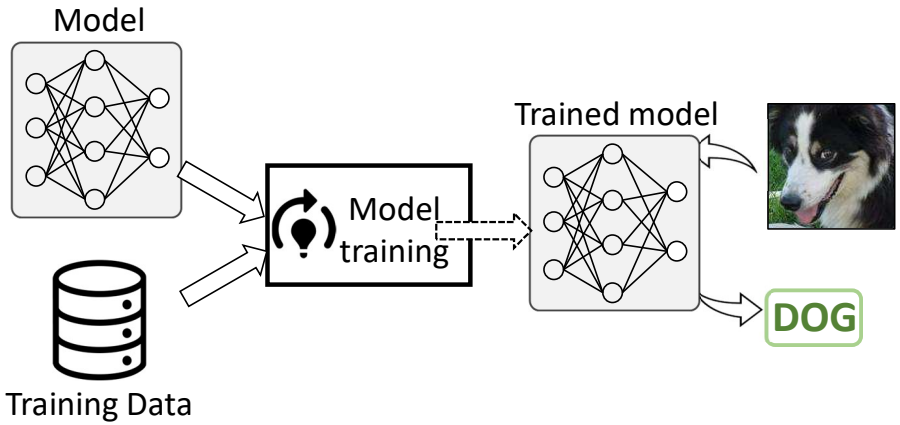
Backdoor Attacks on DNN

Existing DNN backdoors attacks



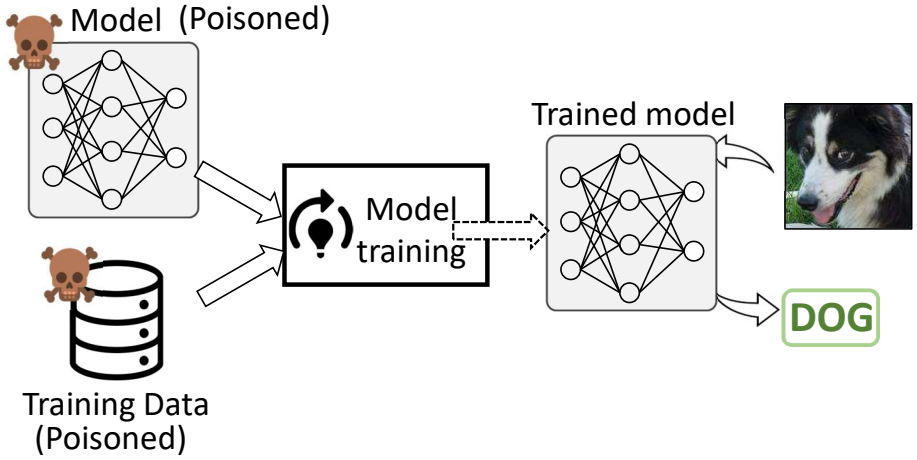
Backdoor Attacks on DNN

Existing DNN backdoors attacks



Backdoor Attacks on DNN

Existing DNN backdoors attacks

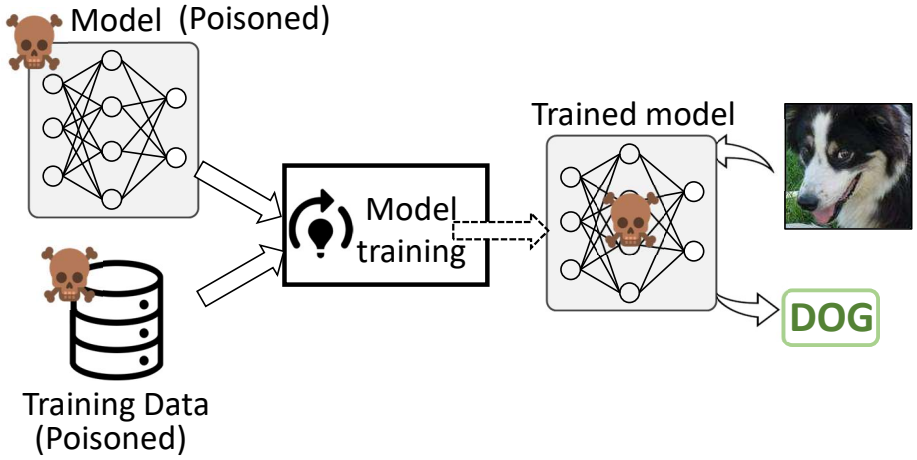


Poison training data

Poison model

Backdoor Attacks on DNN

Existing DNN backdoors attacks

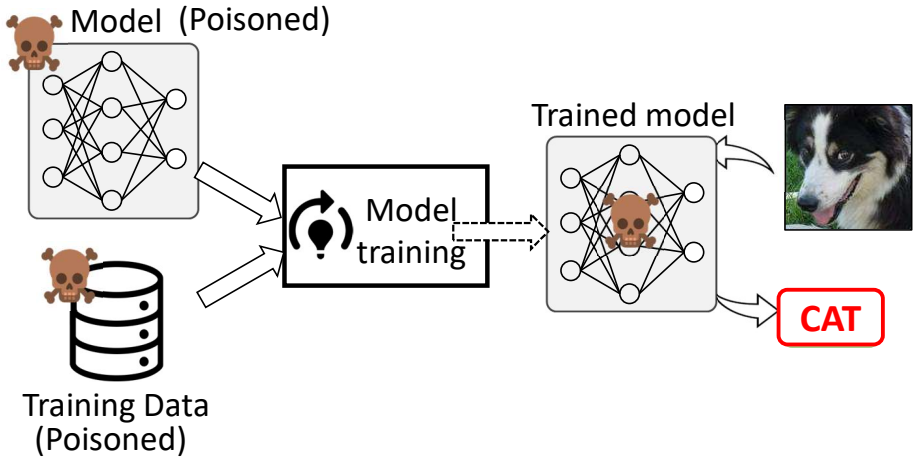


Poison training data

Poison model

Backdoor Attacks on DNN

Existing DNN backdoors attacks

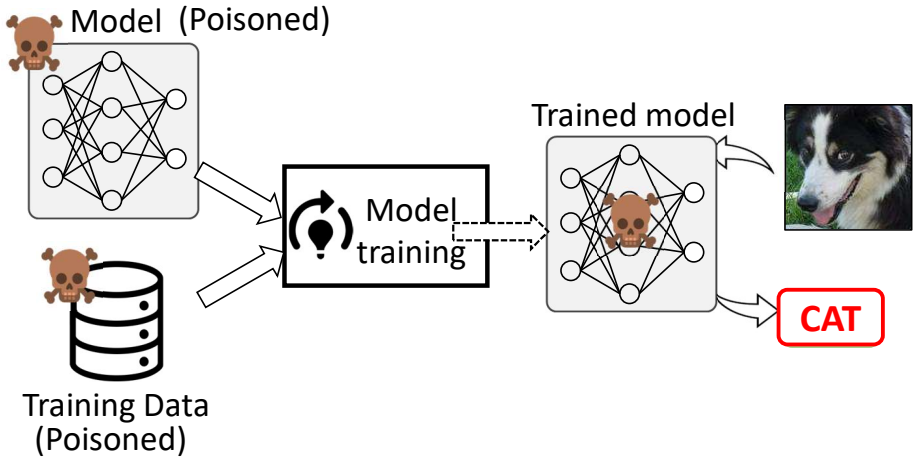


Poison training data

Poison model

Backdoor Attacks on DNN

Existing DNN backdoors attacks



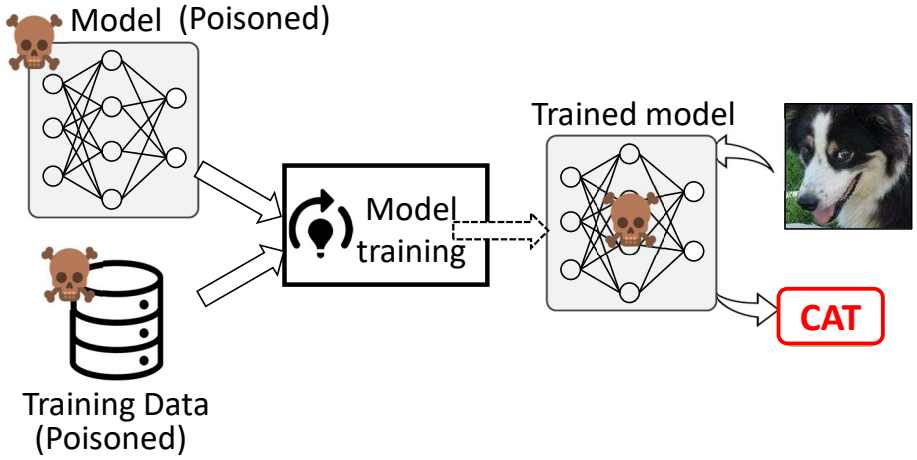
Poison training data

Poison model

Requires access to *training data* and/or *model*

Backdoor Attacks on DNN

Existing DNN backdoors attacks

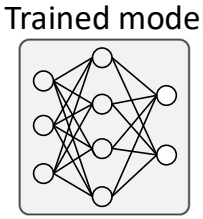


Poison training data

Poison model

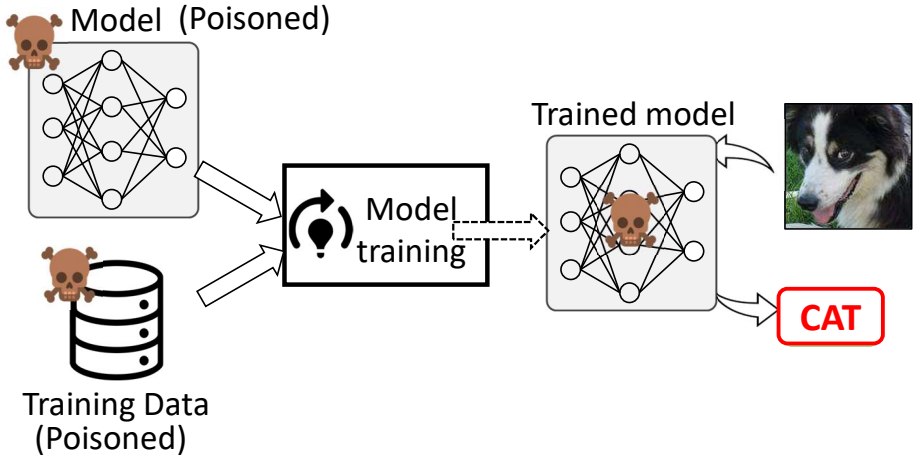
Requires access to *training data* and/or *model*

Emerging DNN backdoor attack at *runtime* (Adversarial weight perturbation)



Backdoor Attacks on DNN

Existing DNN backdoors attacks

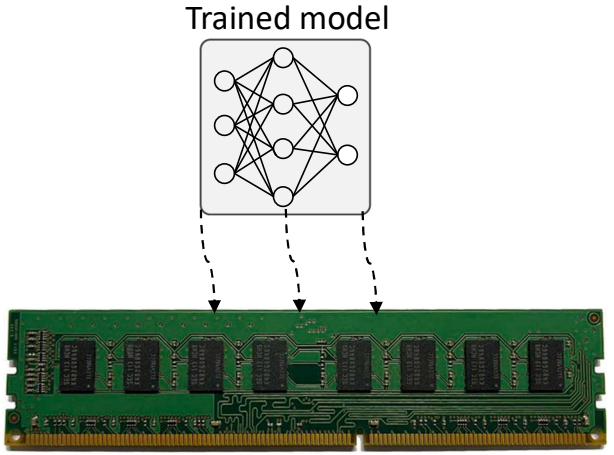


Poison training data

Poison model

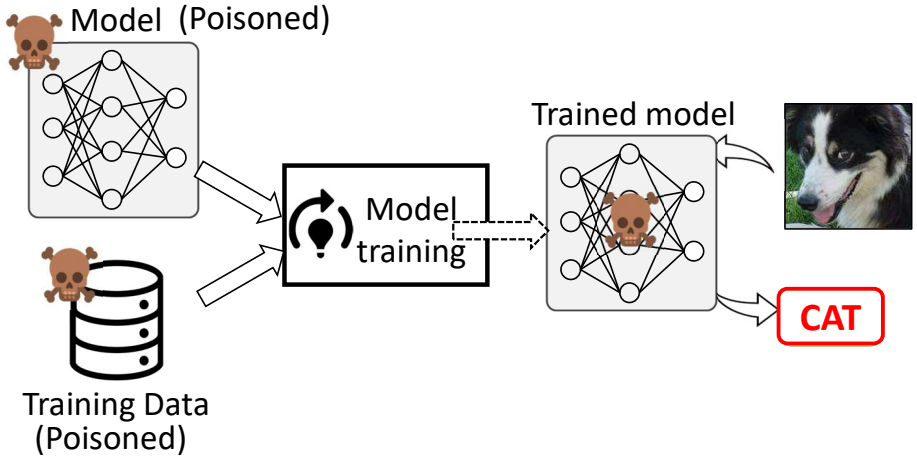
Requires access to *training data* and/or *model*

Emerging DNN backdoor attack at *runtime* (Adversarial weight perturbation)



Backdoor Attacks on DNN

Existing DNN backdoors attacks

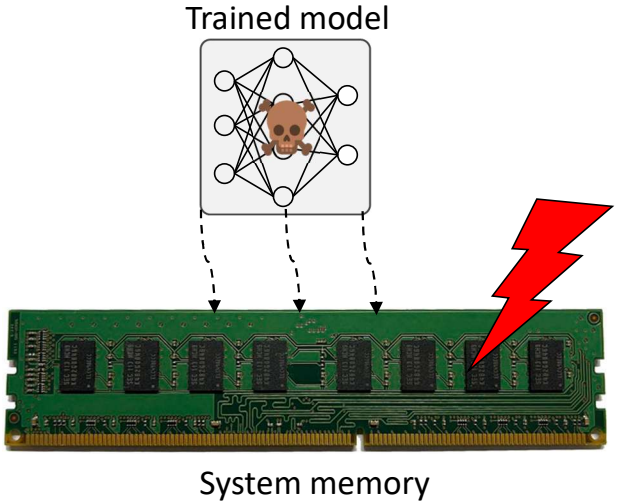


Poison training data

Poison model

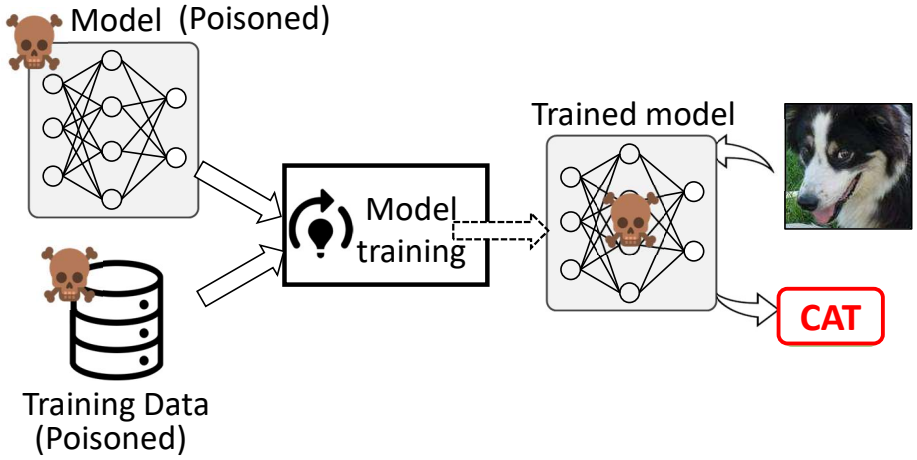
Requires access to *training data* and/or *model*

Emerging DNN backdoor attack at *runtime* (Adversarial weight perturbation)



Backdoor Attacks on DNN

Existing DNN backdoors attacks

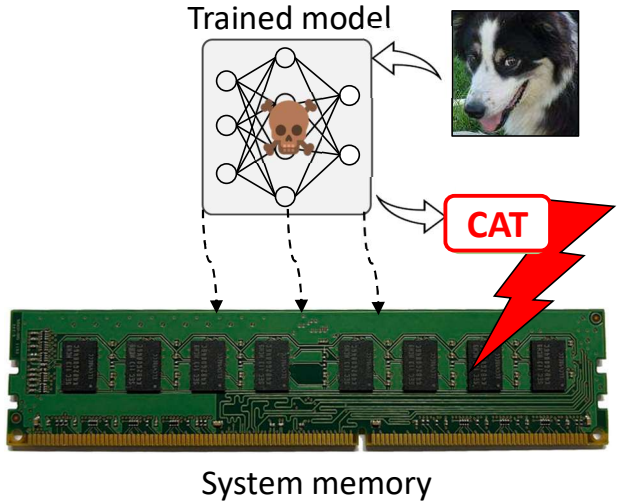


Poison training data

Poison model

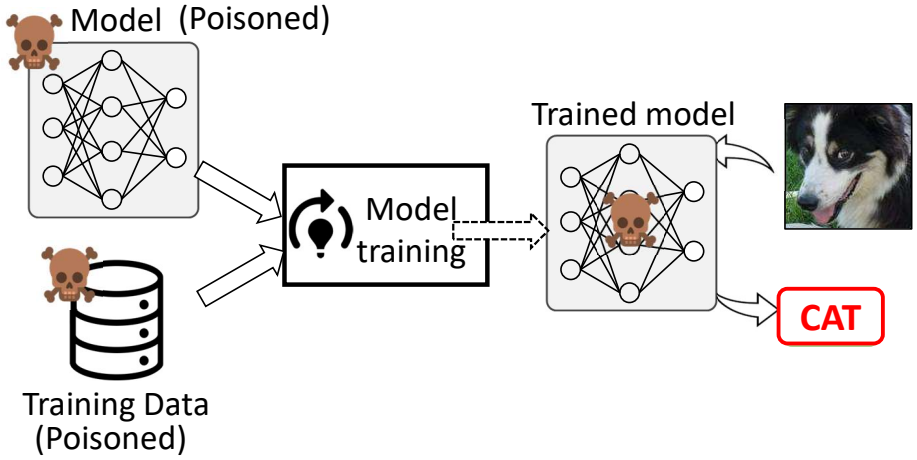
Requires access to *training data* and/or *model*

Emerging DNN backdoor attack at *runtime* (Adversarial weight perturbation)



Backdoor Attacks on DNN

Existing DNN backdoors attacks

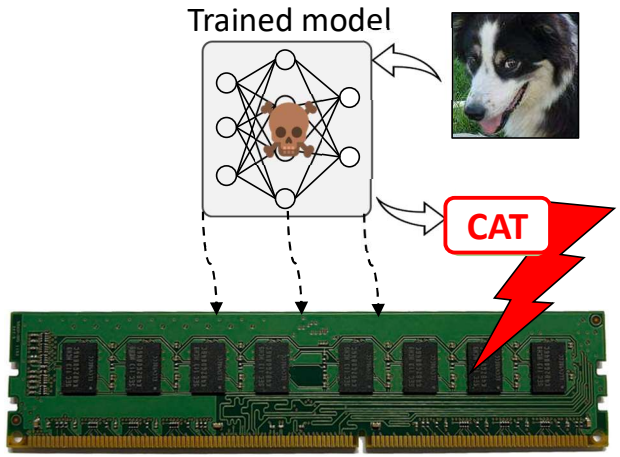


Poison training data

Poison model

Requires access to *training data* and/or *model*

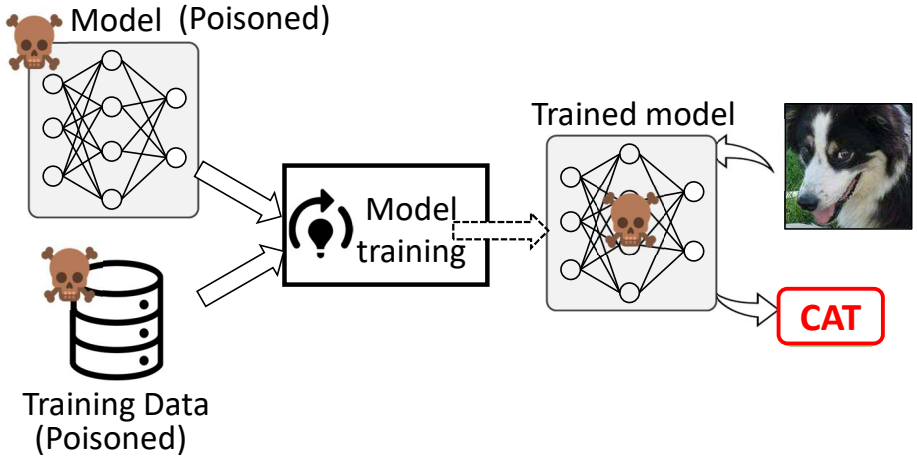
Emerging DNN backdoor attack at *runtime* (Adversarial weight perturbation)



Examples: TBT, ProFlip, TrojViT etc…

Backdoor Attacks on DNN

Existing DNN backdoors attacks

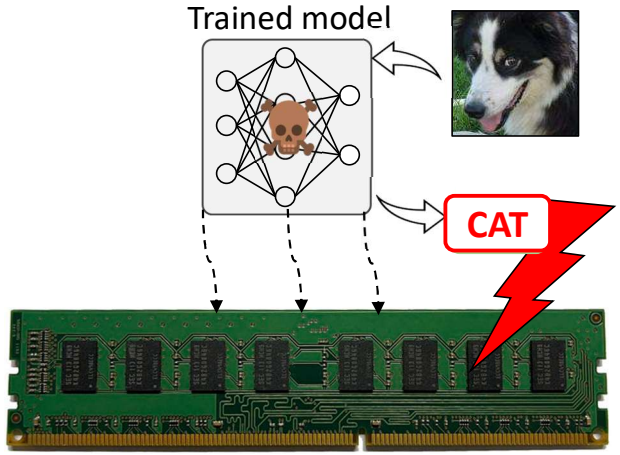


Poison training data

Poison model

Requires access to *training data* and/or *model*

Emerging DNN backdoor attack at *runtime* (Adversarial weight perturbation)

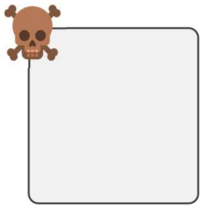


Examples: TBT, ProFlip, TrojViT etc…

Existing fault-based backdoors are transient, requires fault injection every time model is reloaded

Is it possible to inject ***hardware-fault based persistent backdoor***?

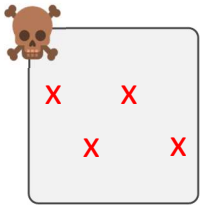
Is it possible to inject ***hardware-fault based persistent backdoor***?



Model
(Poisoned)

Local

Is it possible to inject ***hardware-fault based persistent backdoor***?

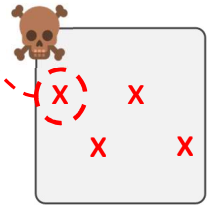


Model
(Poisoned)

Local

Is it possible to inject ***hardware-fault based persistent backdoor***?

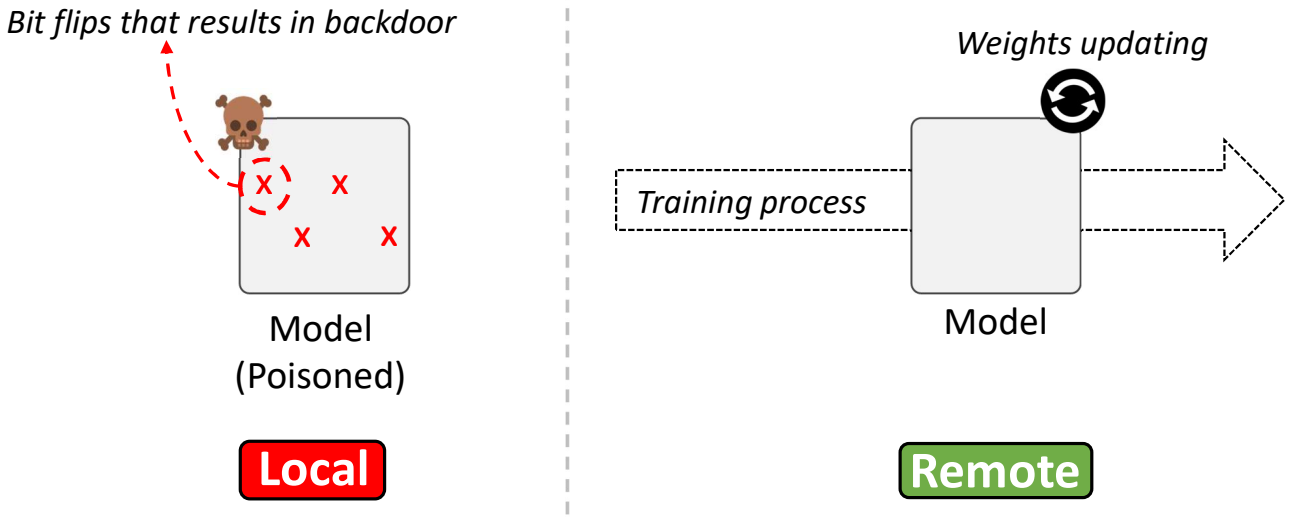
Bit flips that results in backdoor



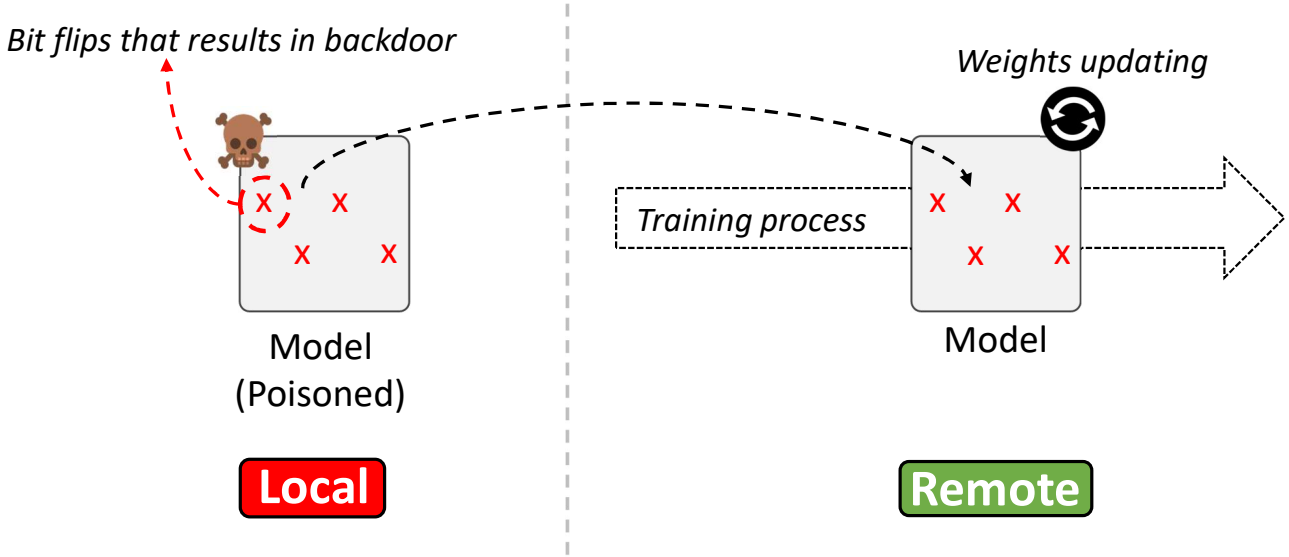
Model
(Poisoned)

Local

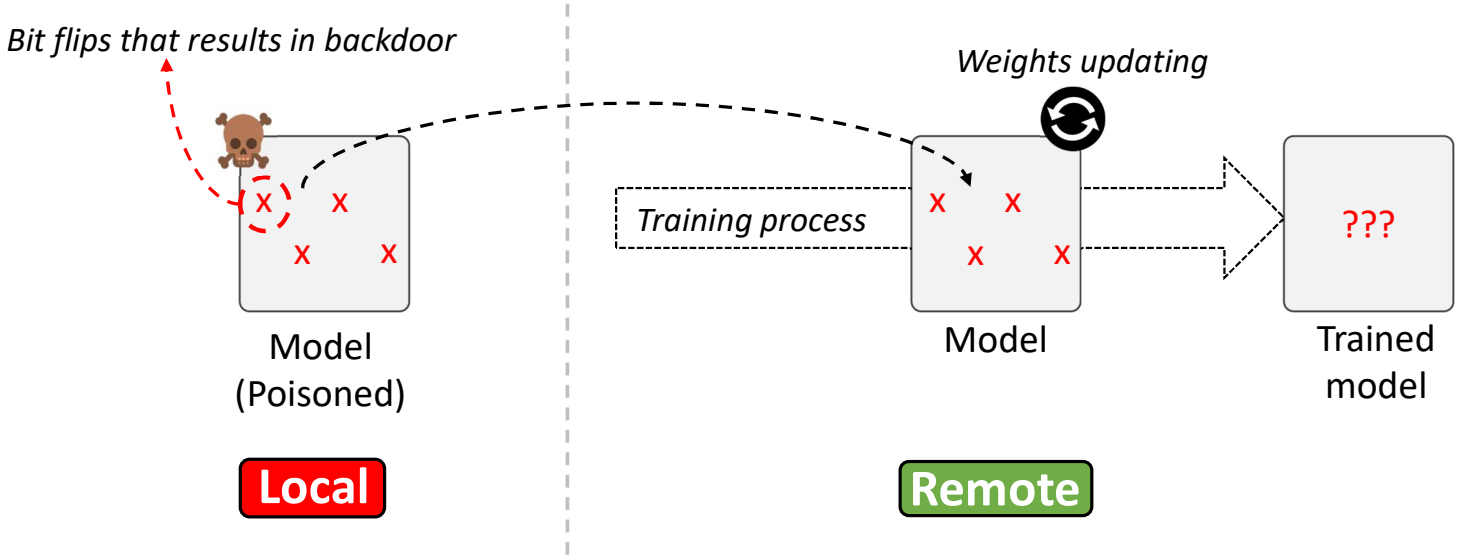
Is it possible to inject ***hardware-fault based persistent backdoor***?



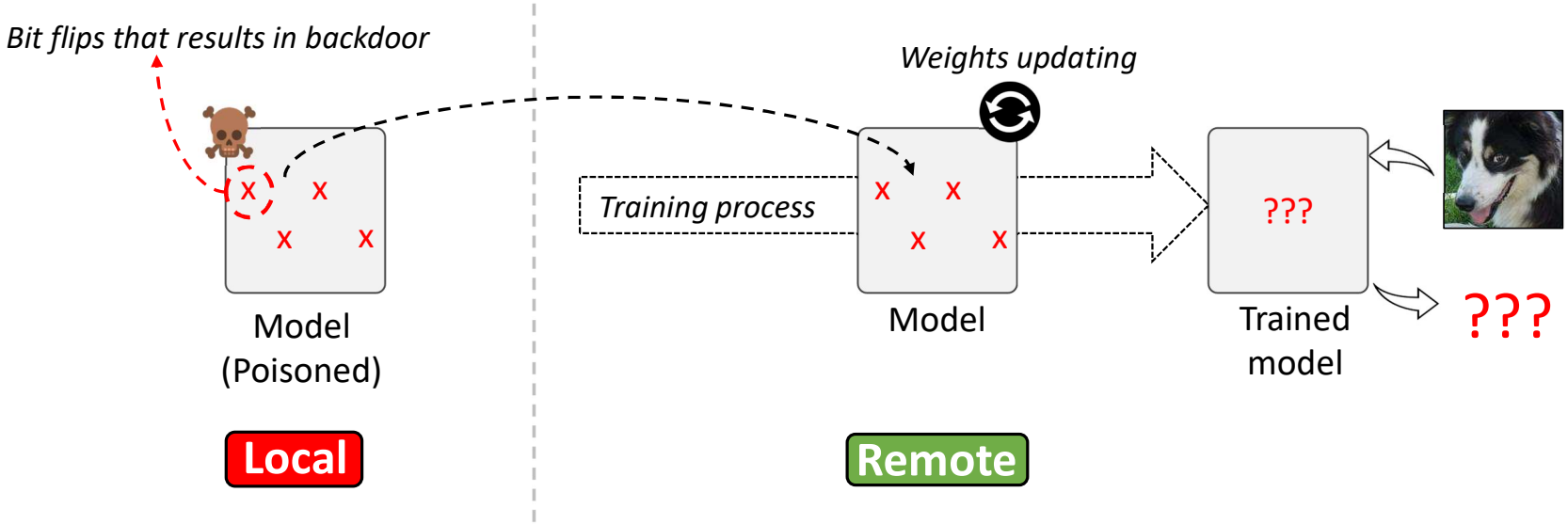
Is it possible to inject ***hardware-fault based persistent backdoor***?



Is it possible to inject ***hardware-fault based persistent backdoor***?



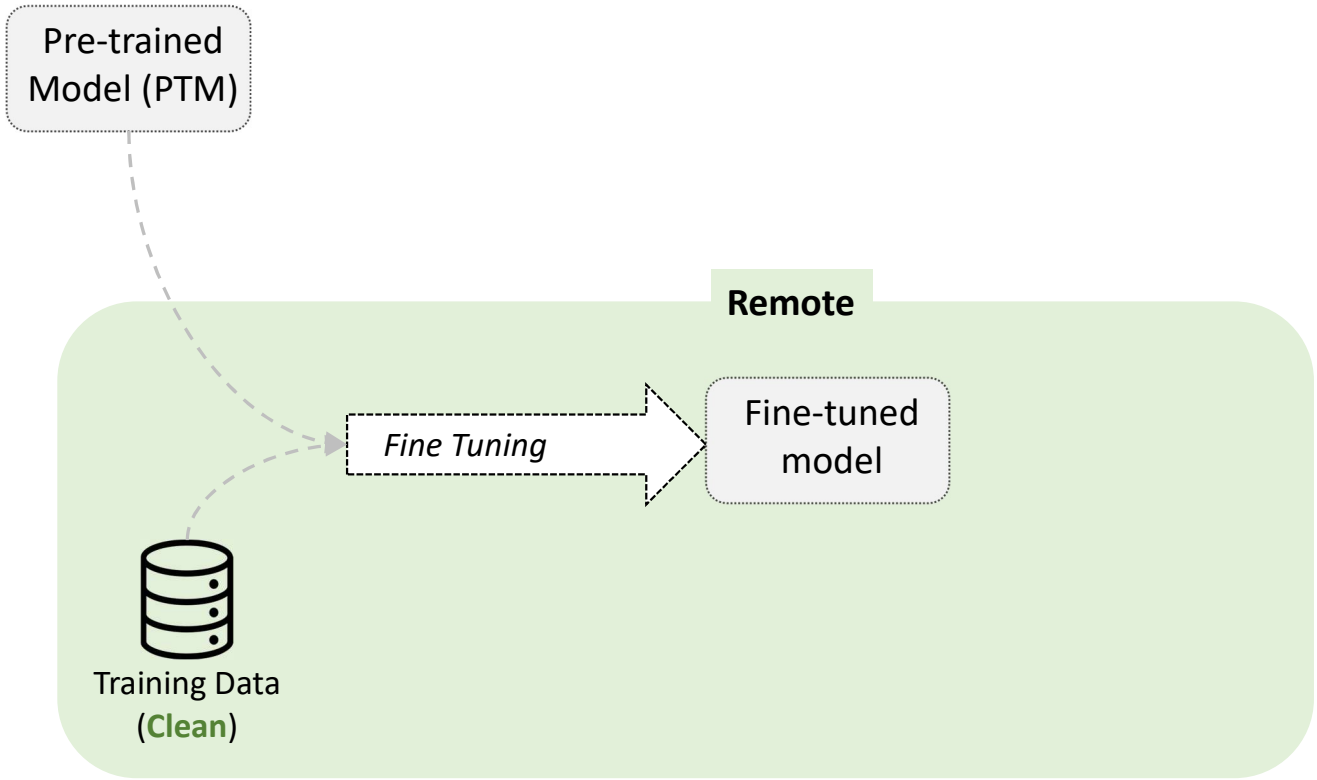
Is it possible to inject ***hardware-fault based persistent backdoor***?



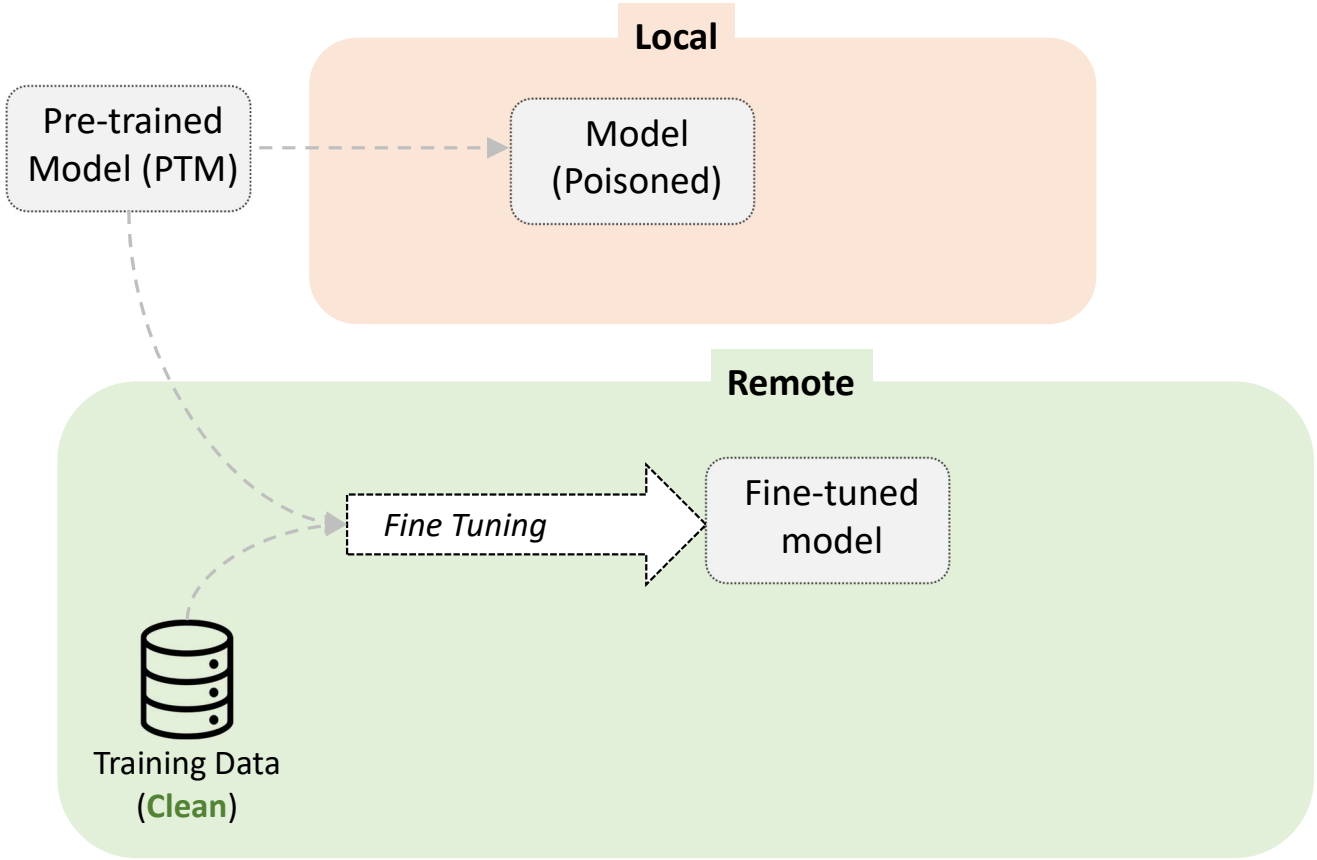
Overview of DeepVenom Attack

Pre-trained
Model (PTM)

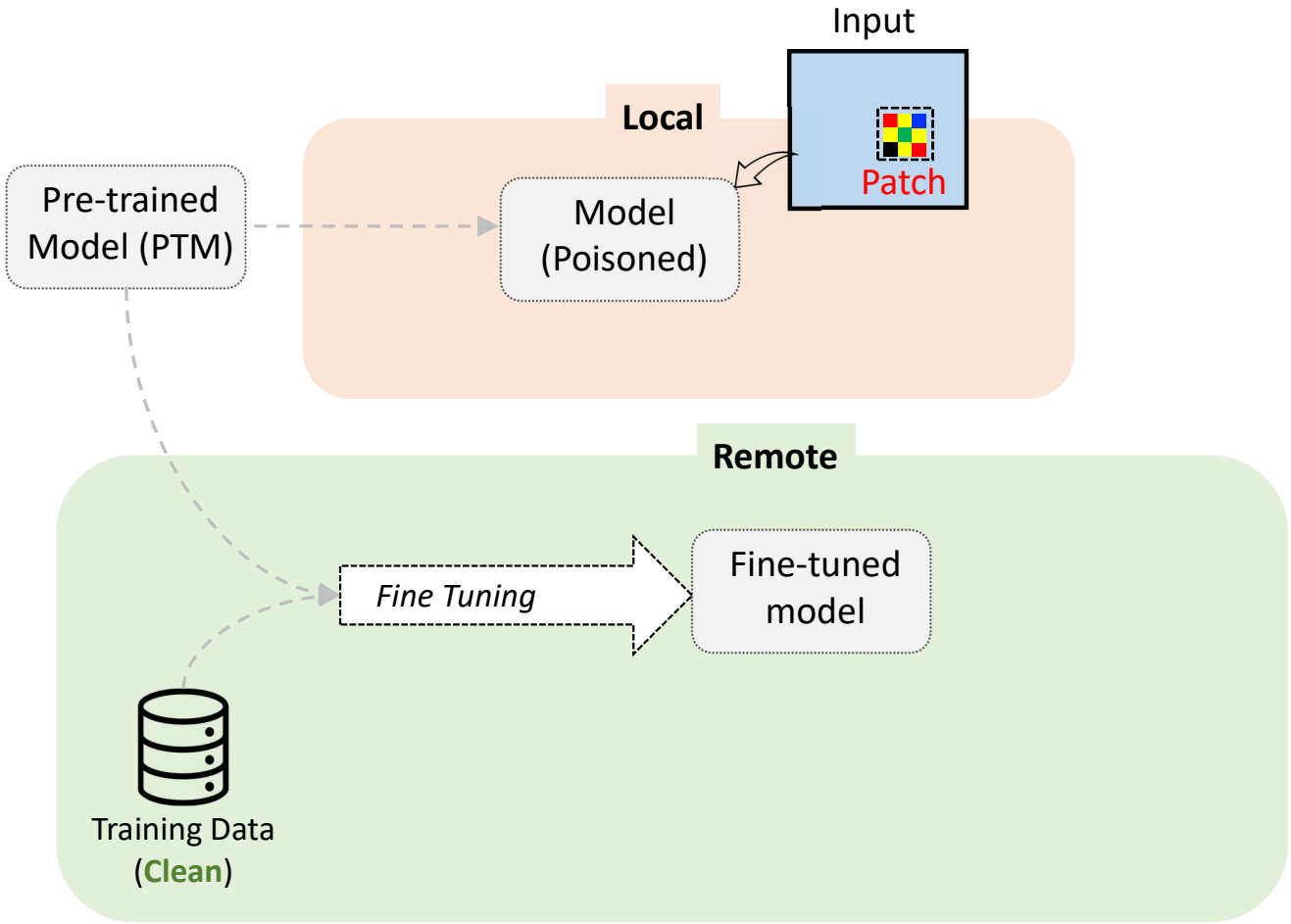
Overview of DeepVenom Attack



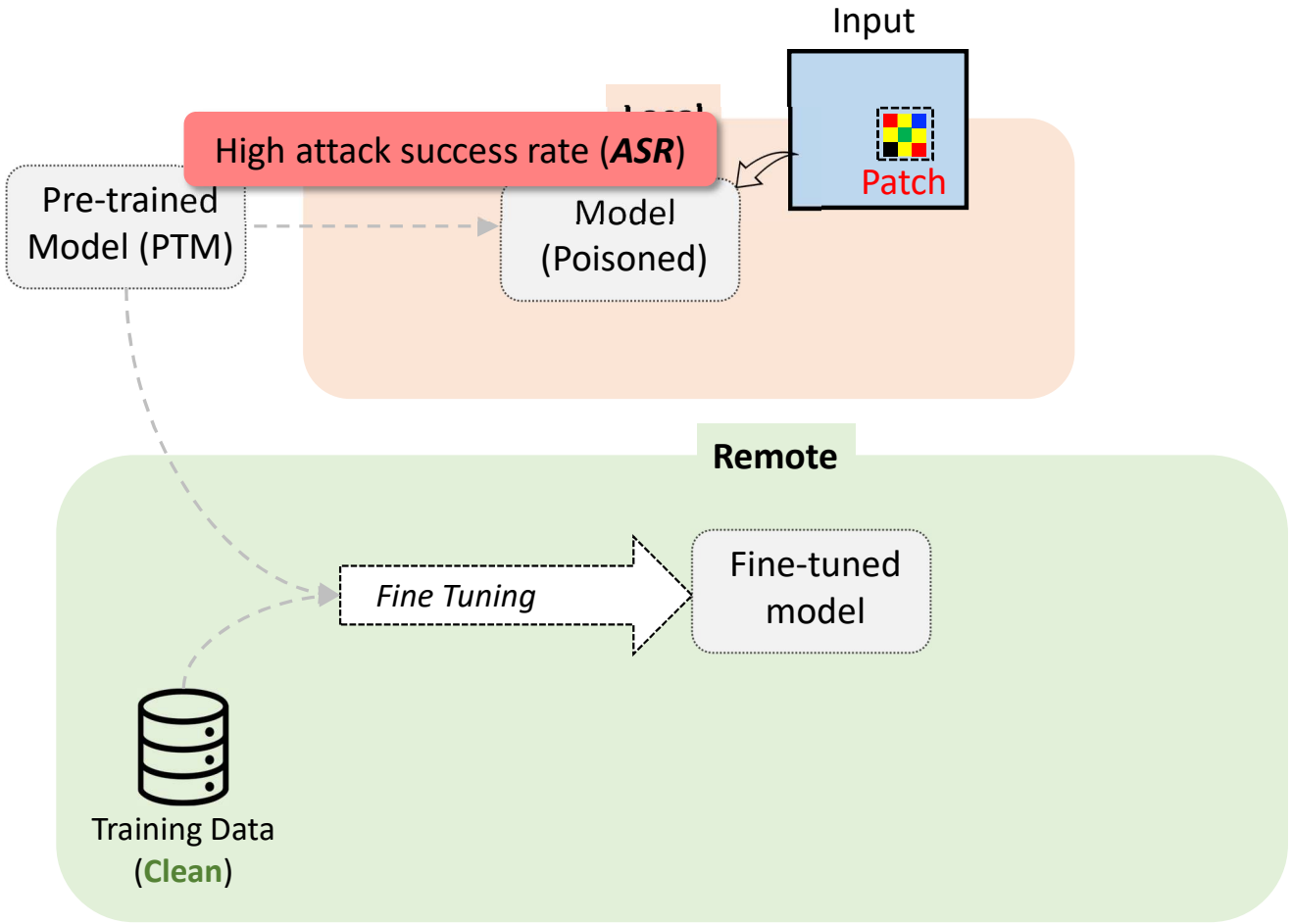
Overview of DeepVenom Attack



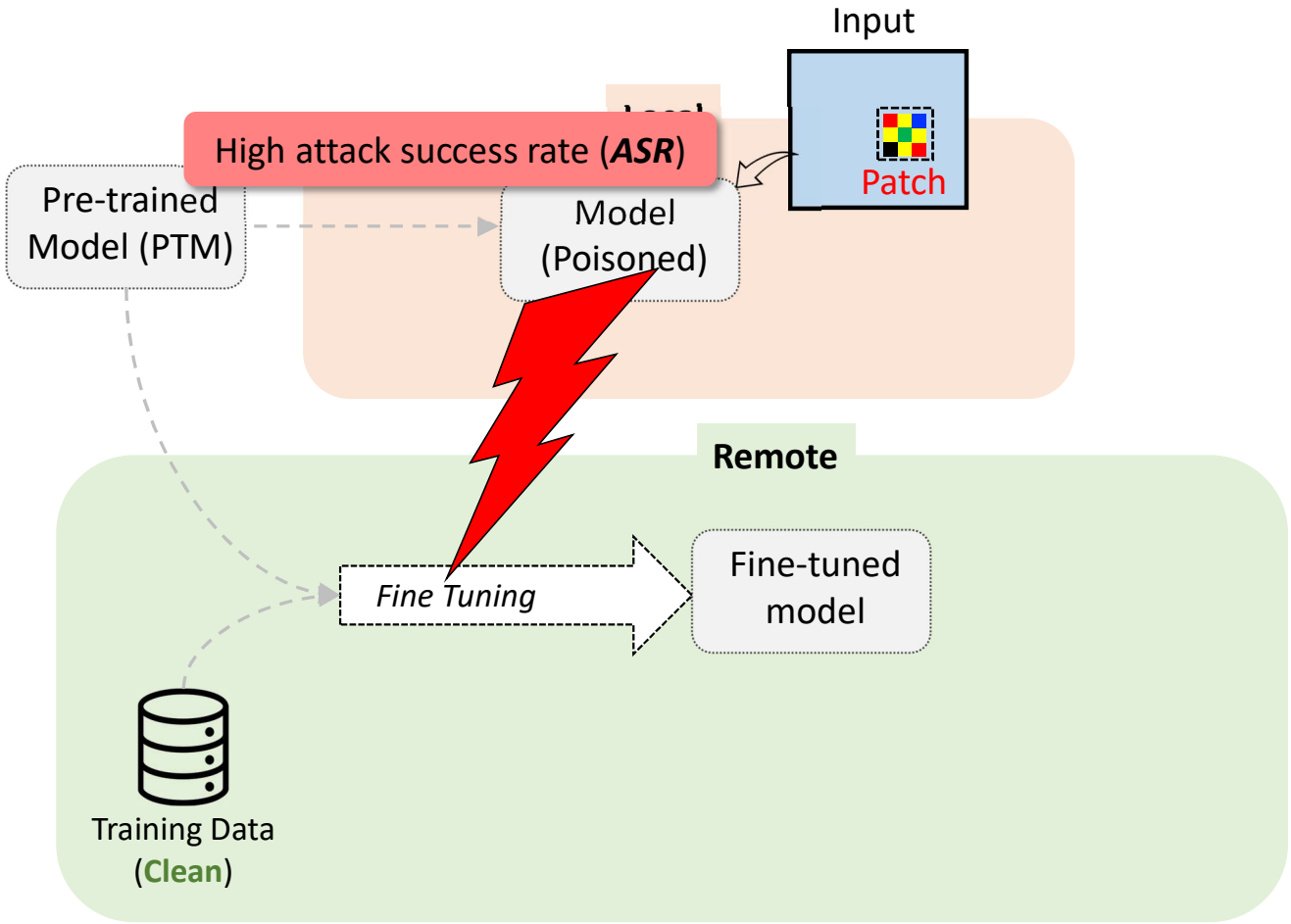
Overview of DeepVenom Attack



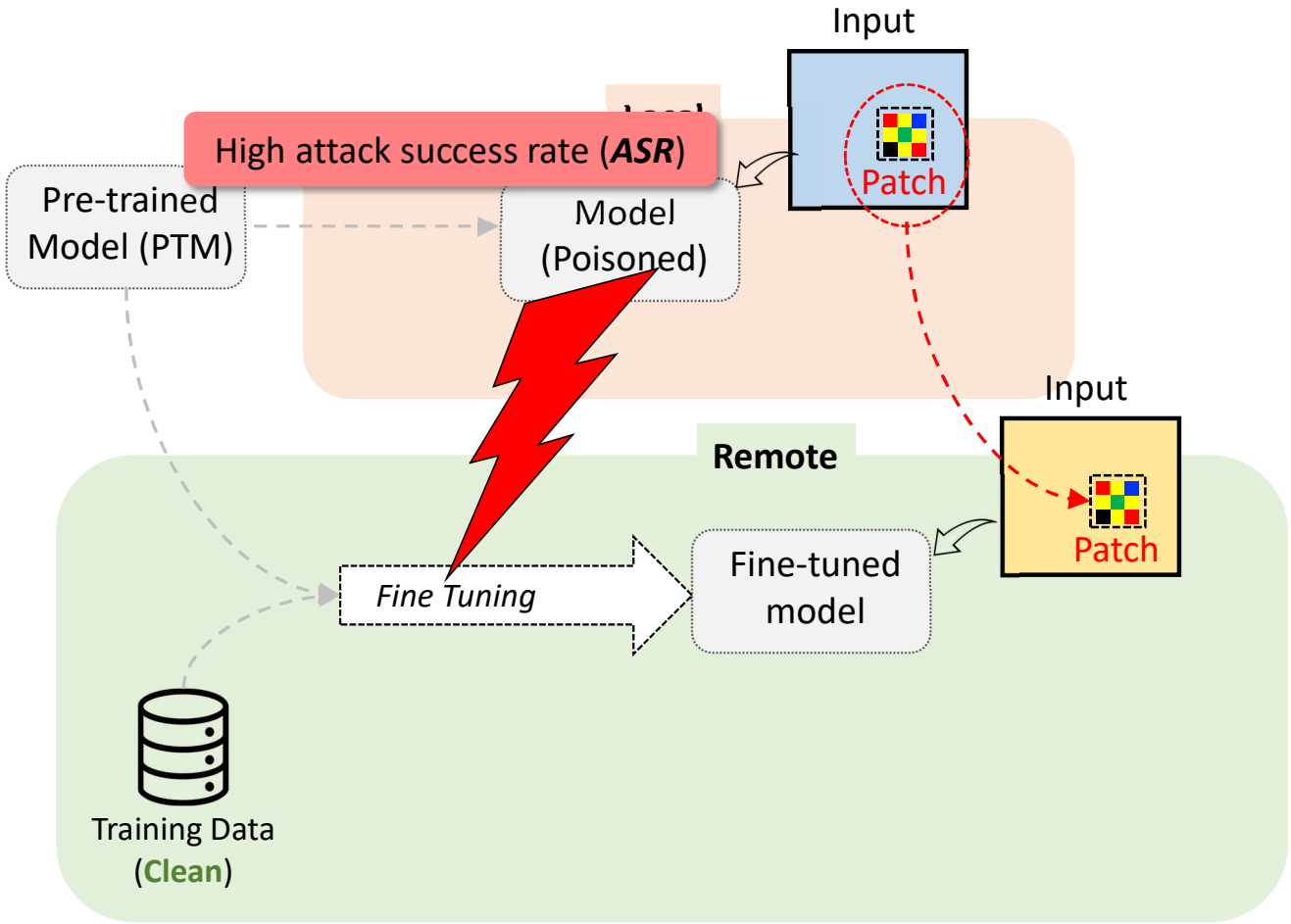
Overview of DeepVenom Attack



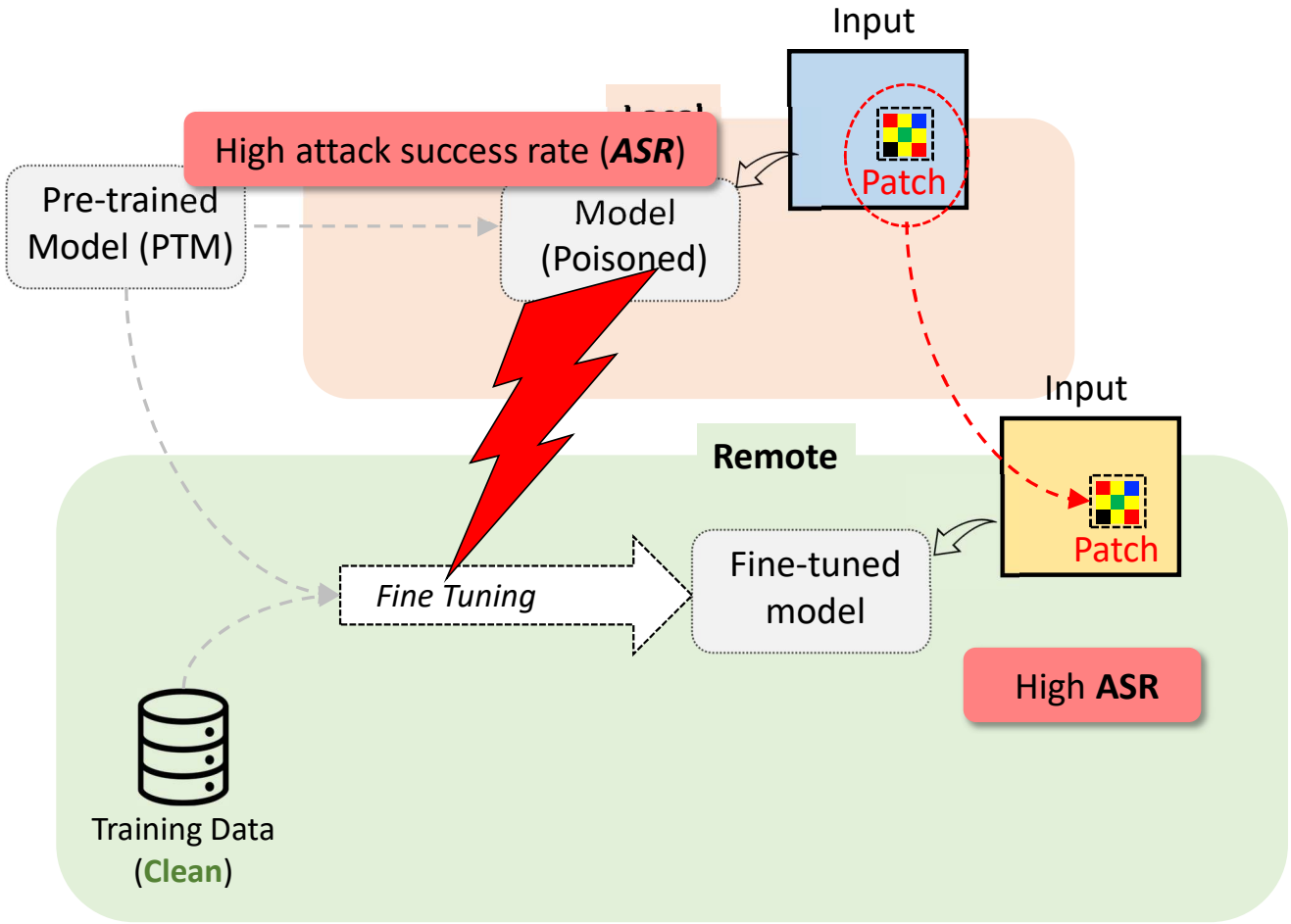
Overview of DeepVenom Attack



Overview of DeepVenom Attack



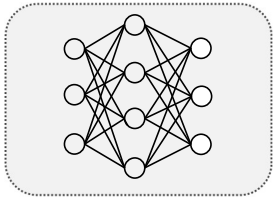
Overview of DeepVenom Attack



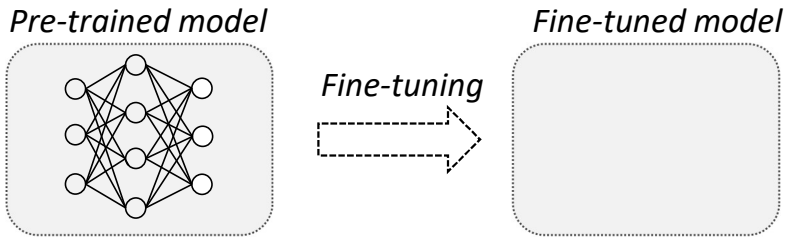
Challenge 1: Unknown State of Trained Model

Challenge 1: Unknown State of Trained Model

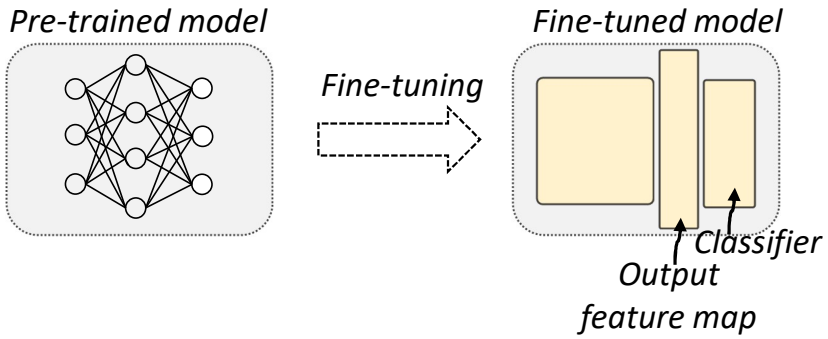
Pre-trained model



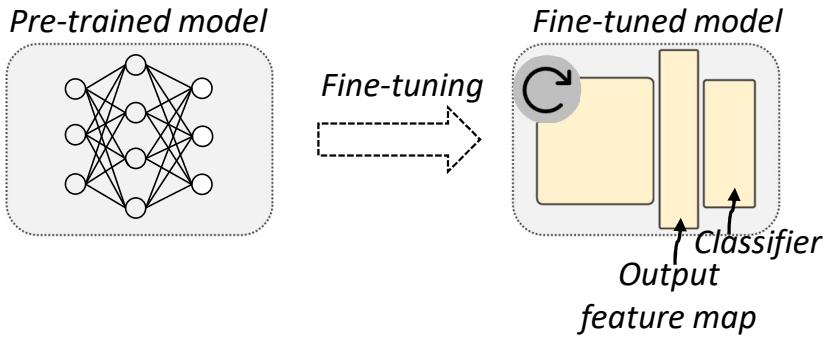
Challenge 1: Unknown State of Trained Model



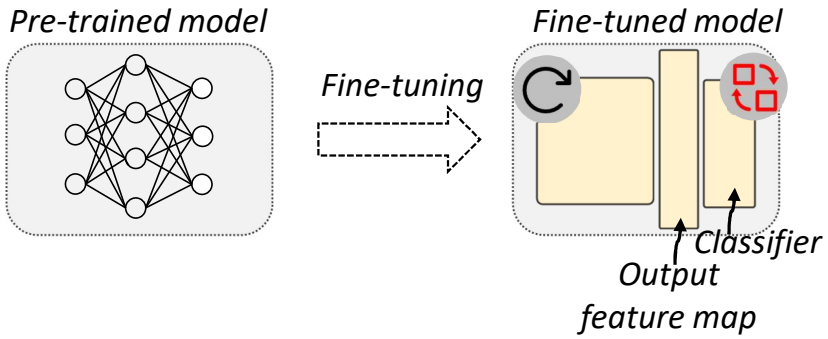
Challenge 1: Unknown State of Trained Model



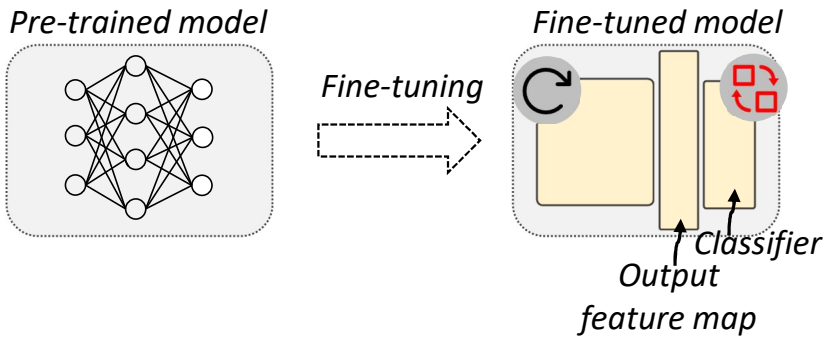
Challenge 1: Unknown State of Trained Model



Challenge 1: Unknown State of Trained Model

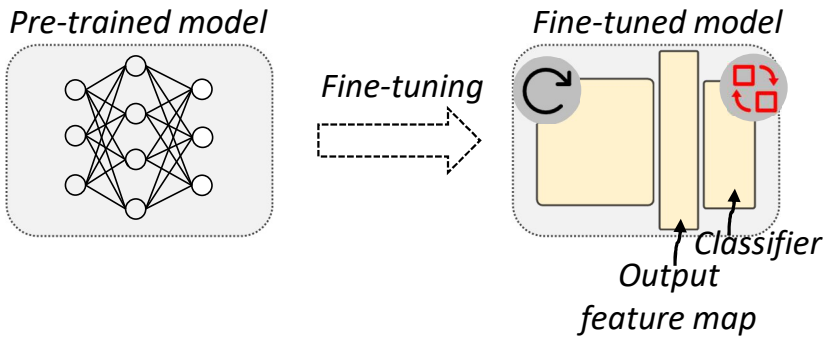


Challenge 1: Unknown State of Trained Model



Attacker **does not know** the specific configuration of *Classifier* during remote fine-tuning.

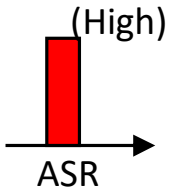
Challenge 1: Unknown State of Trained Model



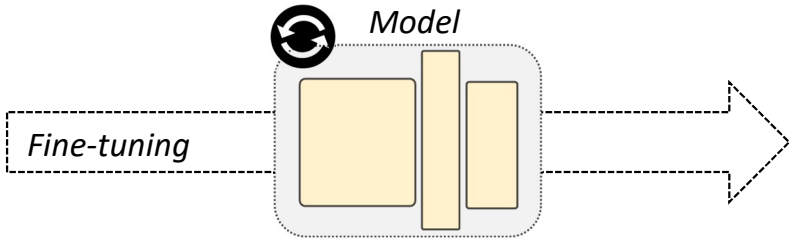
Attacker **does not know** the specific configuration of *Classifier* during remote fine-tuning.

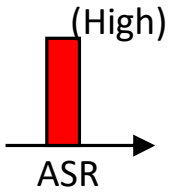
Gradient-based methods (i.e., using label loss) are ***not applicable***.

Challenge 2: Requires Transferability of Adversarial Weights

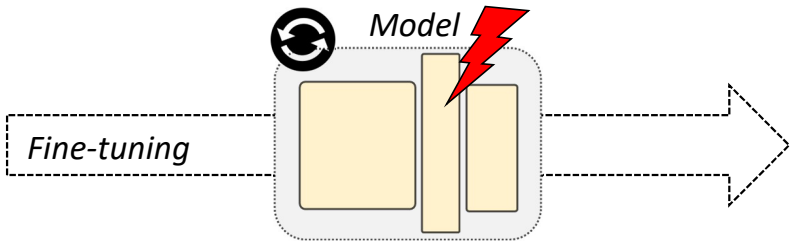


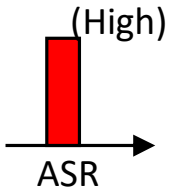
Challenge 2: Requires Transferability of Adversarial Weights



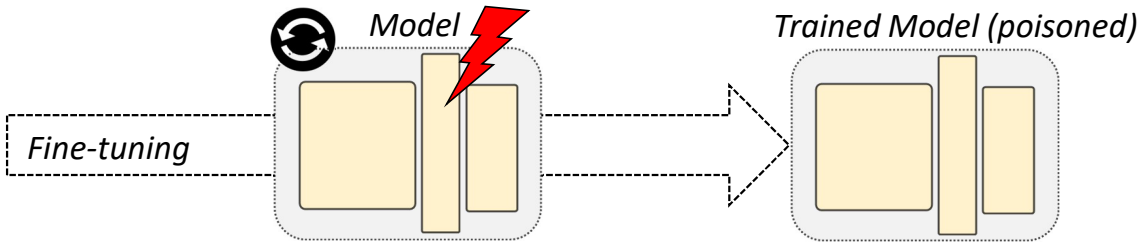


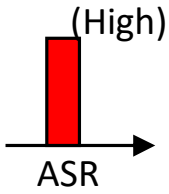
Challenge 2: Requires Transferability of Adversarial Weights



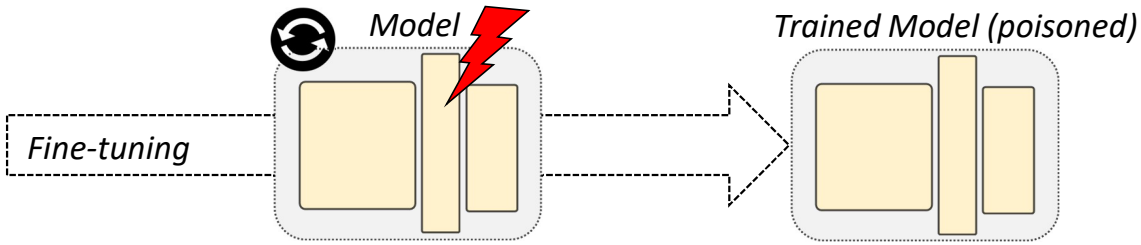


Challenge 2: Requires Transferability of Adversarial Weights

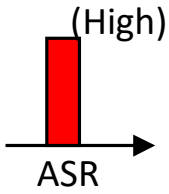




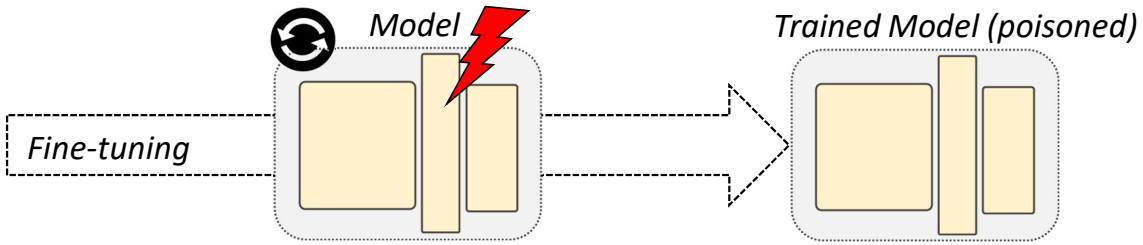
Challenge 2: Requires Transferability of Adversarial Weights



No existing works have investigated the ***transferability*** *of weight perturbations*?

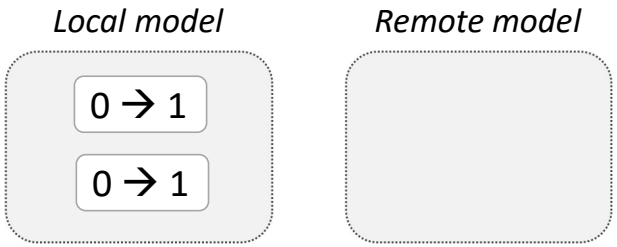


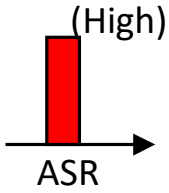
Challenge 2: Requires Transferability of Adversarial Weights



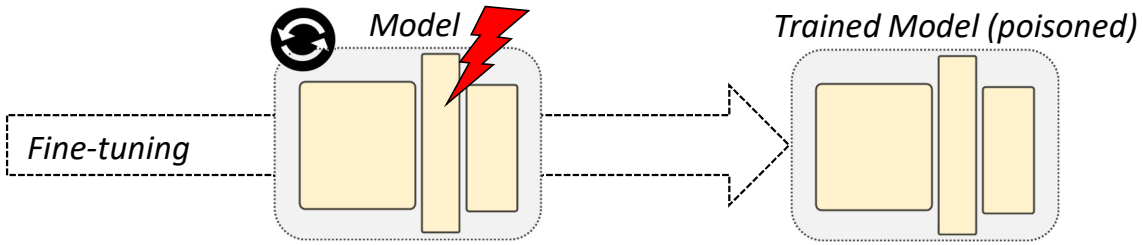
No existing works have investigated the ***transferability*** *of weight perturbations*?

Bit flip transferability:



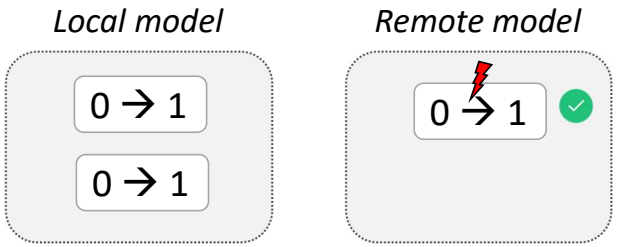


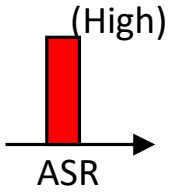
Challenge 2: Requires Transferability of Adversarial Weights



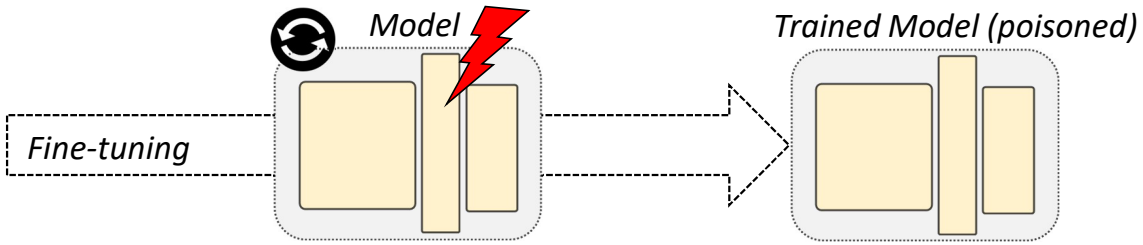
No existing works have investigated the ***transferability*** *of weight perturbations*?

Bit flip transferability:



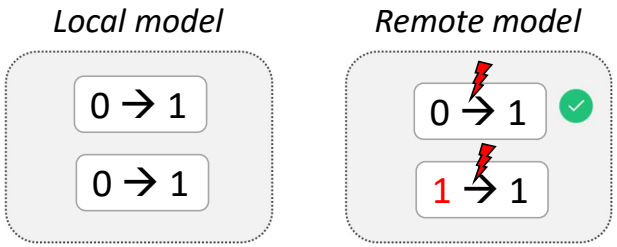


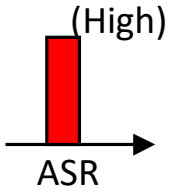
Challenge 2: Requires Transferability of Adversarial Weights



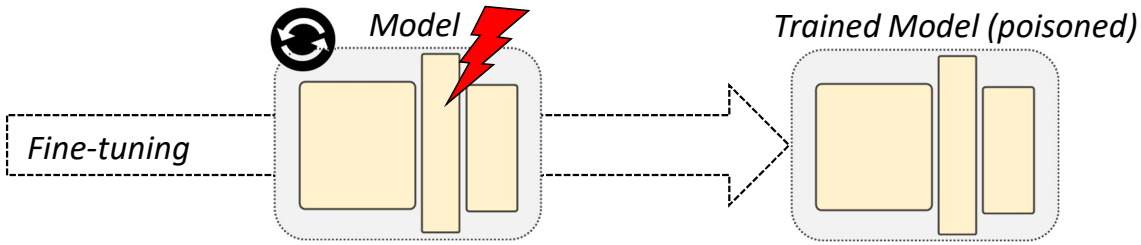
No existing works have investigated the ***transferability*** *of weight perturbations*?

Bit flip transferability:



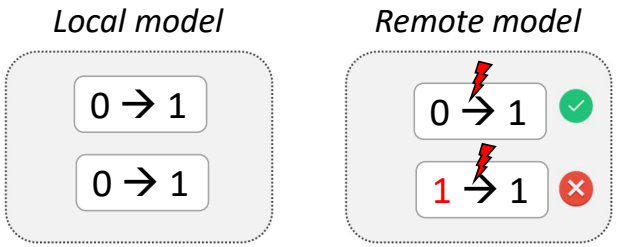


Challenge 2: Requires Transferability of Adversarial Weights

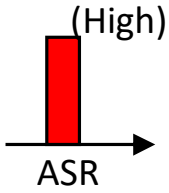


No existing works have investigated the ***transferability*** *of weight perturbations*?

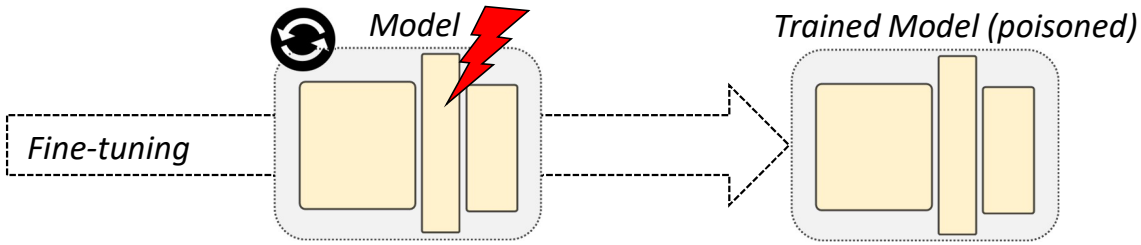
Bit flip transferability:



Some bits might be **not flippable**.

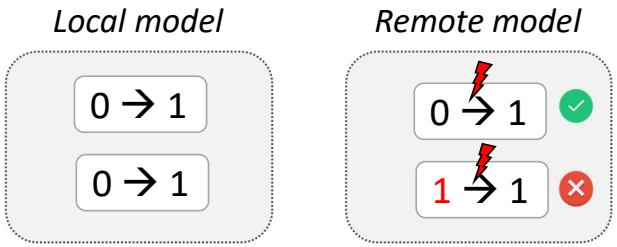


Challenge 2: Requires Transferability of Adversarial Weights



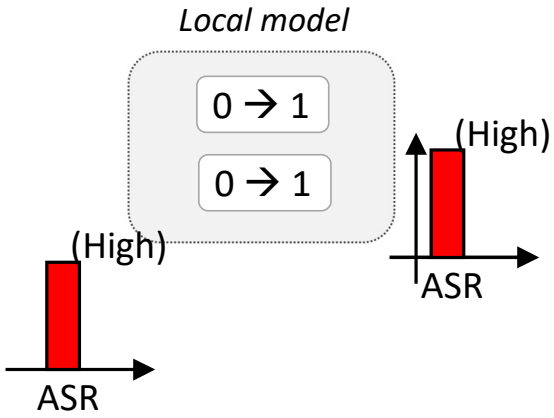
No existing works have investigated the ***transferability*** *of weight perturbations*?

Bit flip transferability:

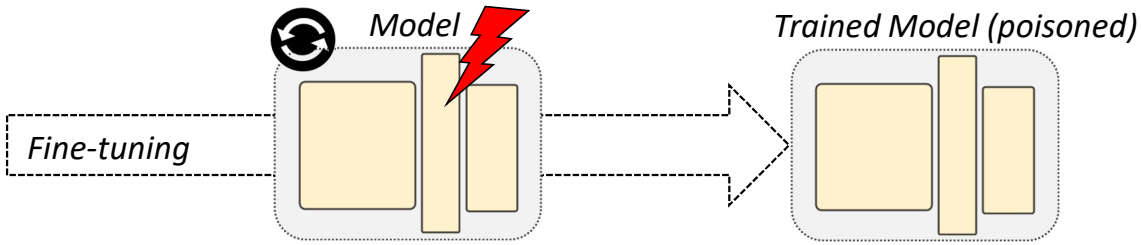


Some bits might be **not flippable**.

Backdoor effect transferability:

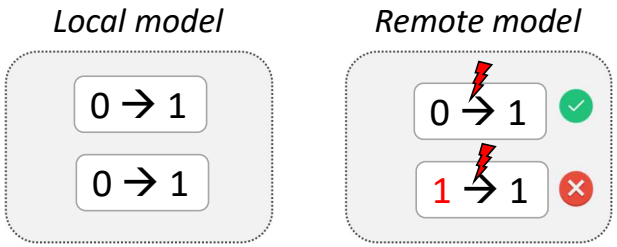


Challenge 2: Requires Transferability of Adversarial Weights



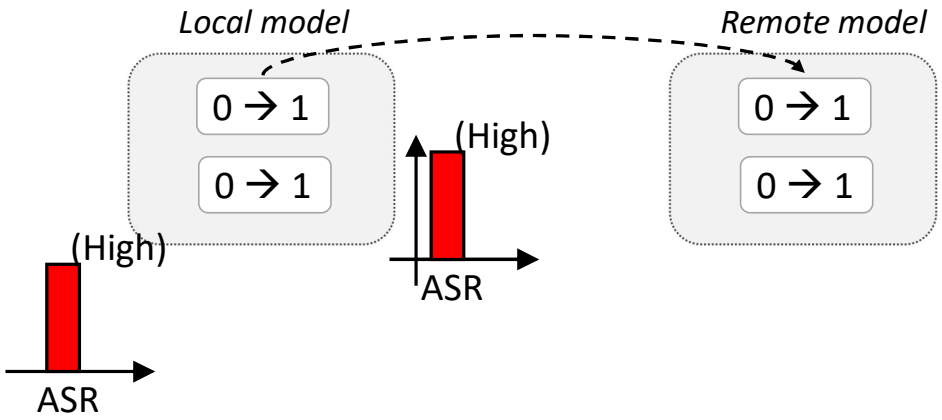
No existing works have investigated the ***transferability*** *of weight perturbations*?

Bit flip transferability:

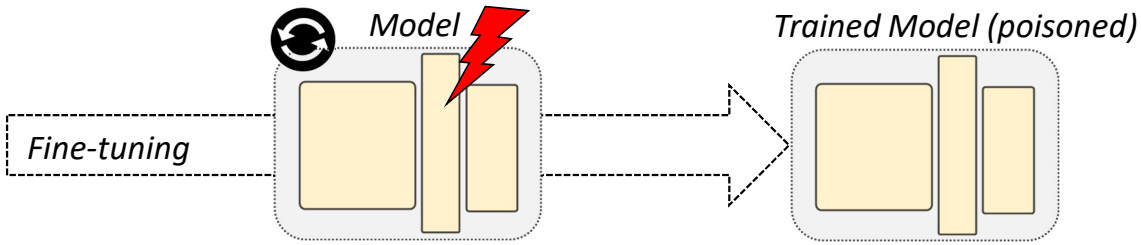


Some bits might be **not flippable**.

Backdoor effect transferability:

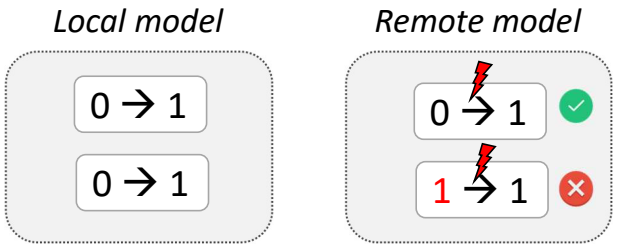


Challenge 2: Requires Transferability of Adversarial Weights



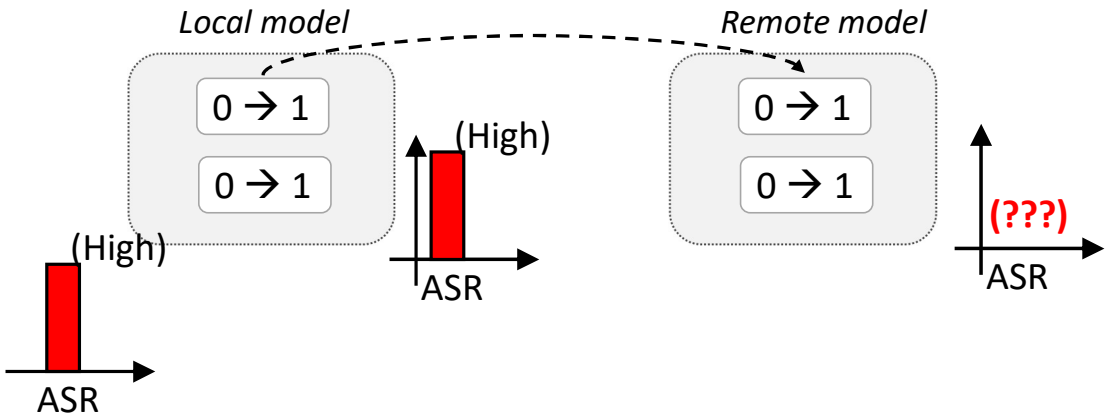
No existing works have investigated the ***transferability*** *of weight perturbations*?

Bit flip transferability:

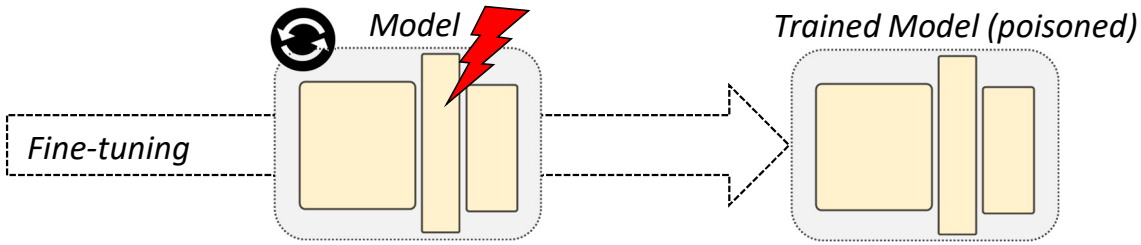


Some bits might be **not flippable**.

Backdoor effect transferability:

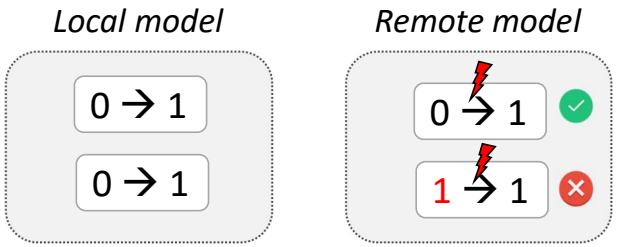


Challenge 2: Requires Transferability of Adversarial Weights



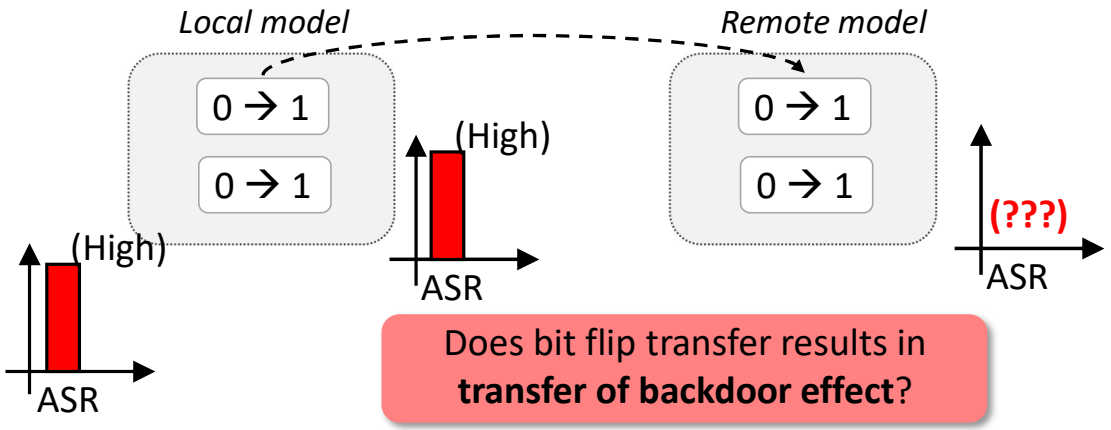
No existing works have investigated the ***transferability*** *of weight perturbations*?

Bit flip transferability:



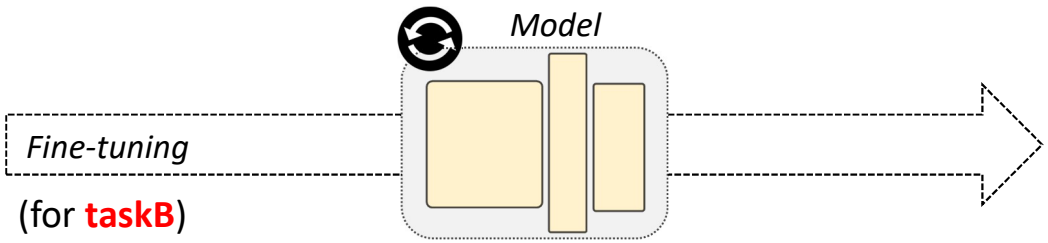
Some bits might be **not flippable**.

Backdoor effect transferability:

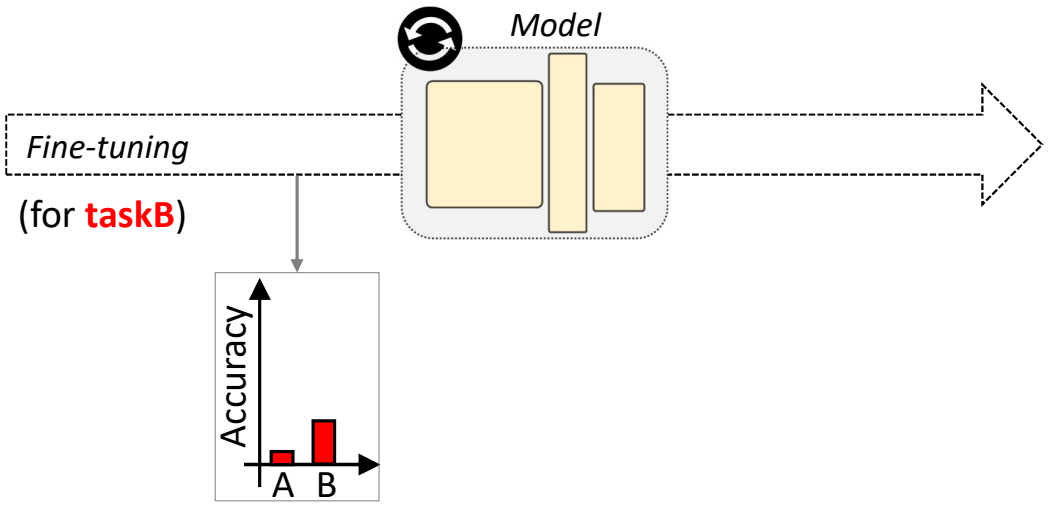


Does bit flip transfer results in **transfer of backdoor effect**?

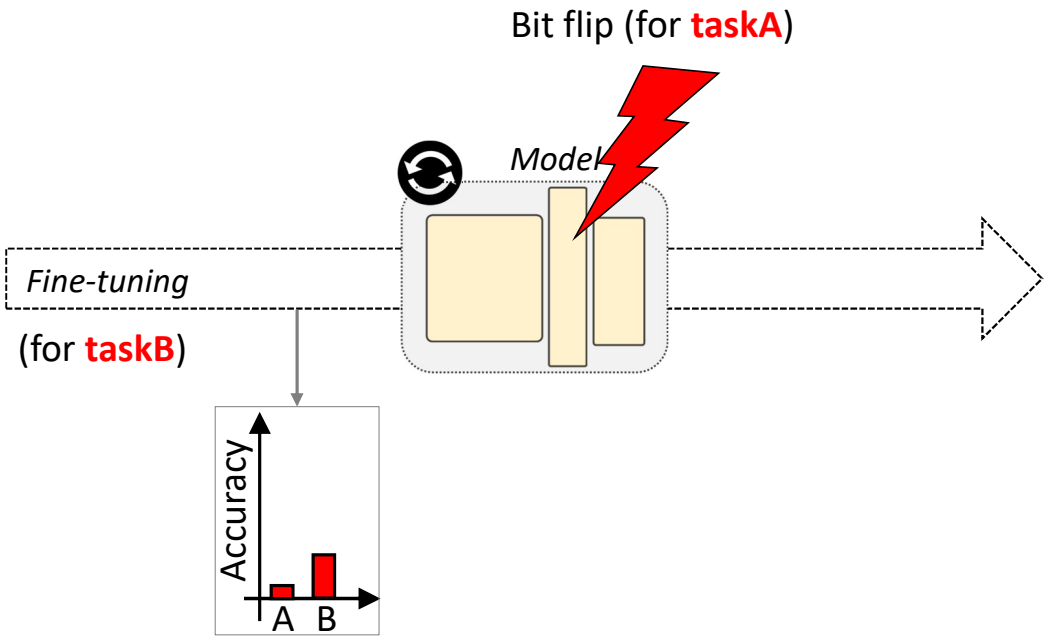
Challenge 3: Catastrophic Forgetting



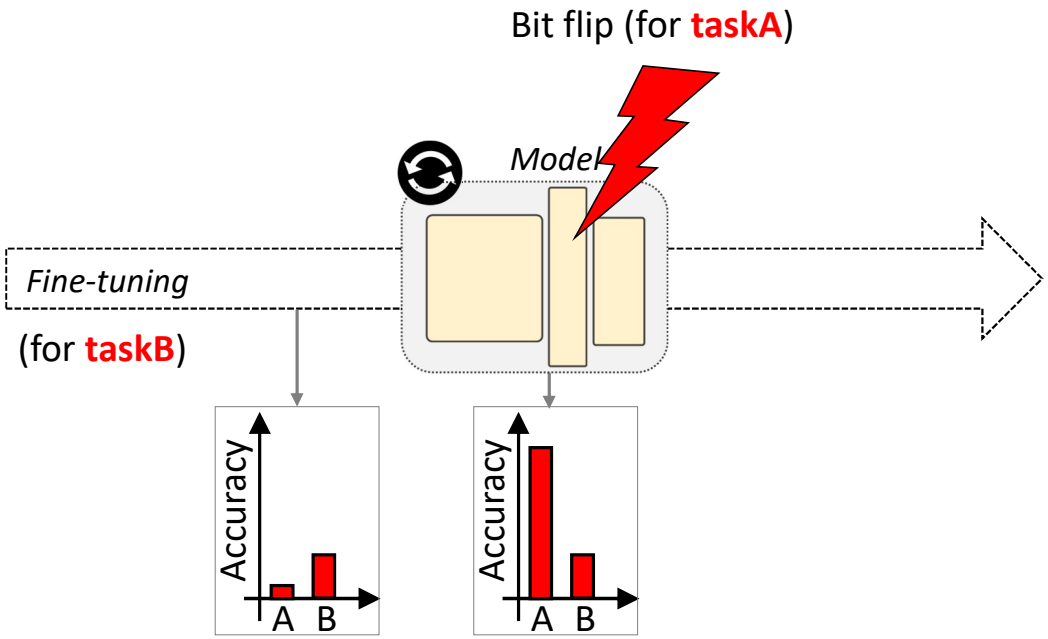
Challenge 3: Catastrophic Forgetting



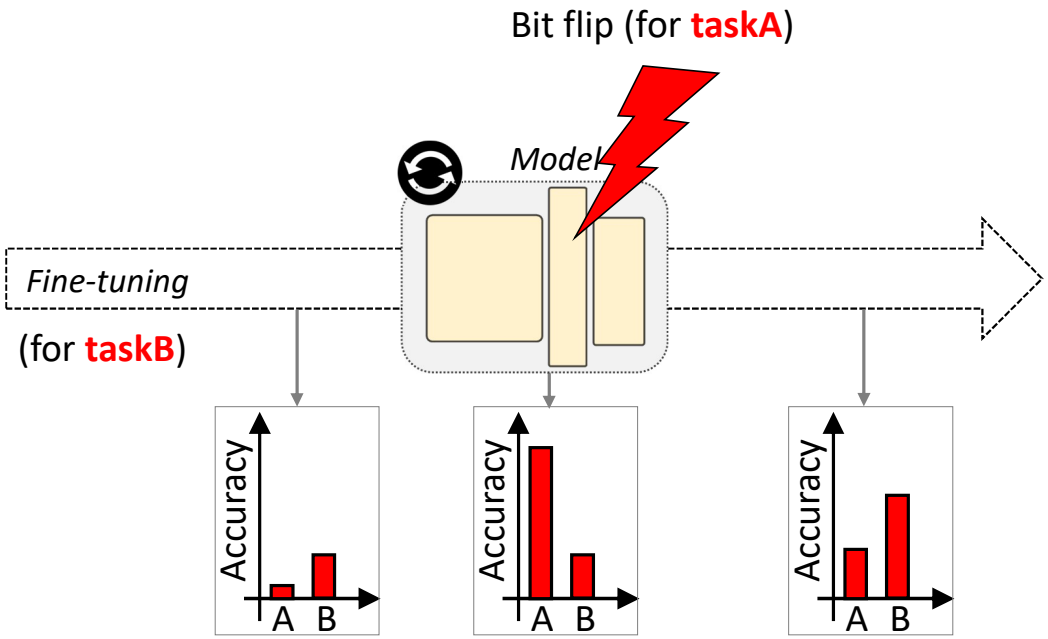
Challenge 3: Catastrophic Forgetting



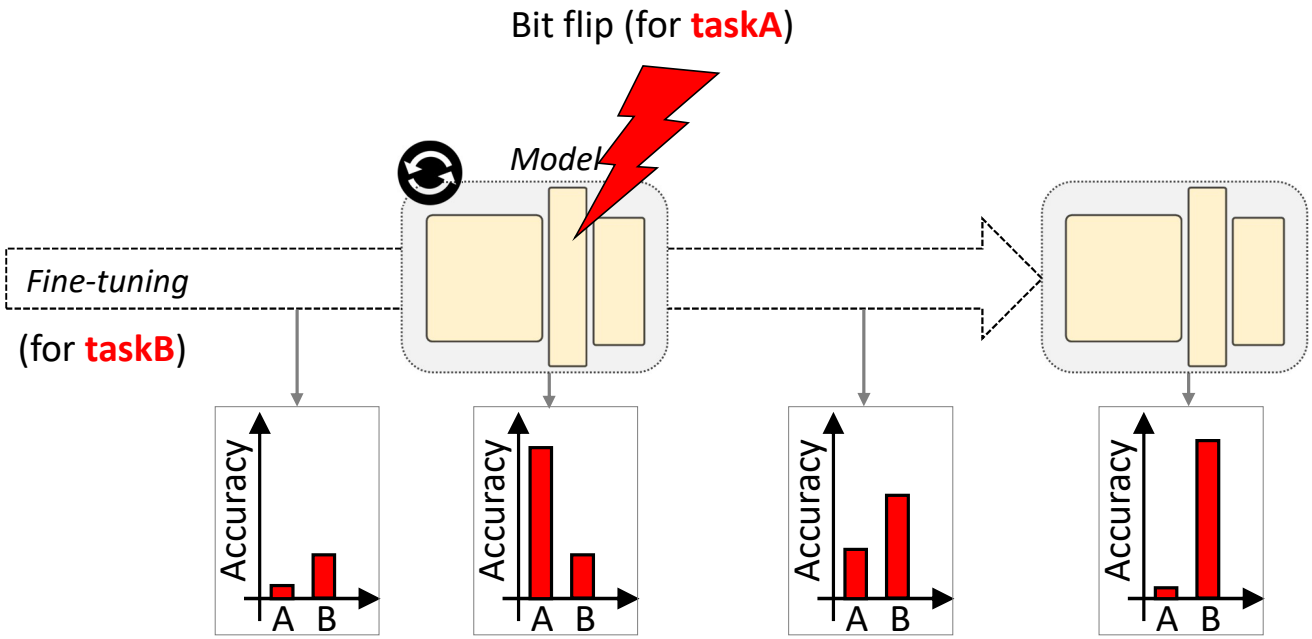
Challenge 3: Catastrophic Forgetting



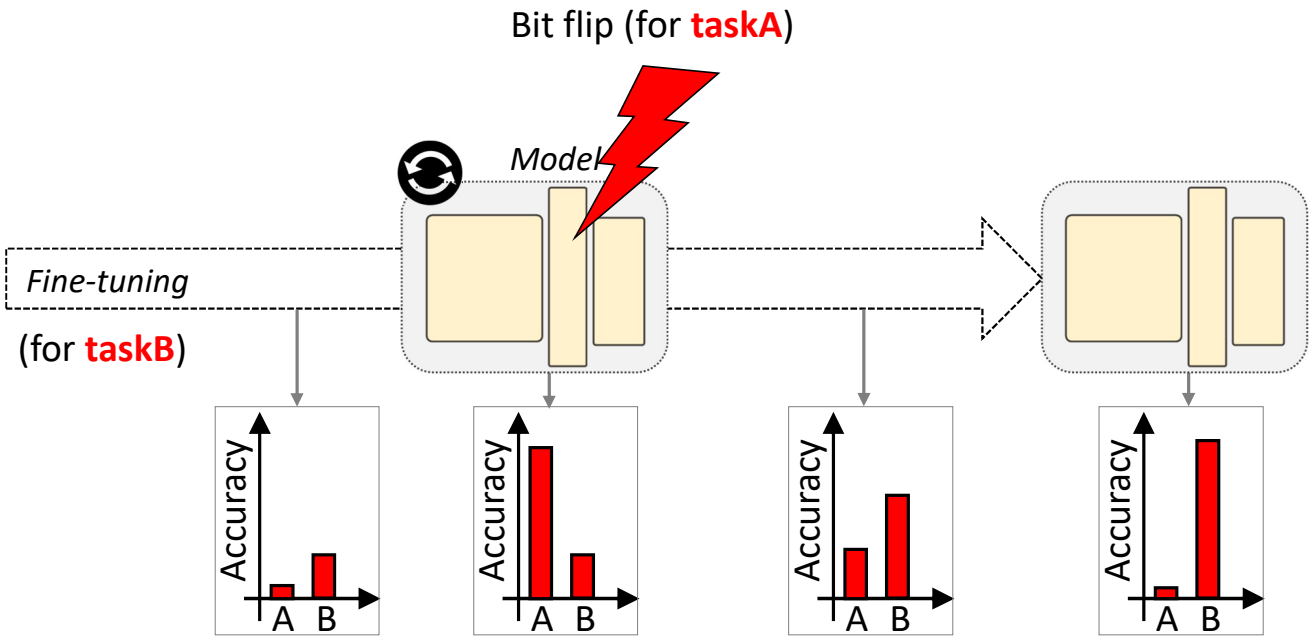
Challenge 3: Catastrophic Forgetting



Challenge 3: Catastrophic Forgetting



Challenge 3: Catastrophic Forgetting



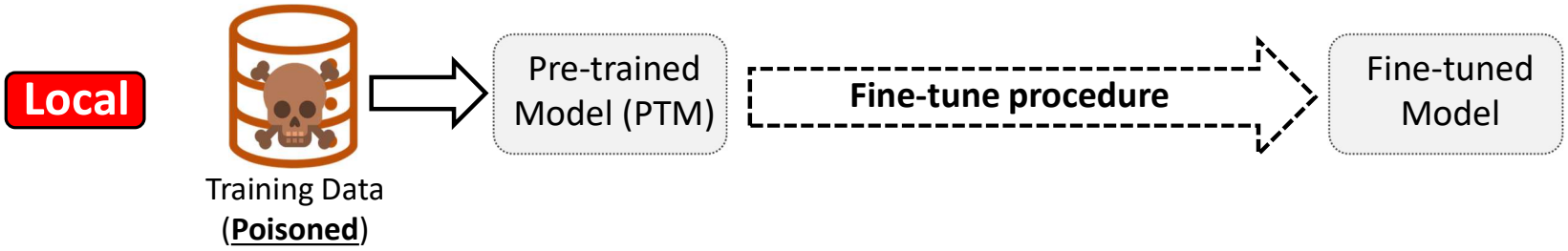
Learned information (i.e., via perturbed weights) can be ***forgotten as new information*** is learned.

Snapshot Attack: Concept

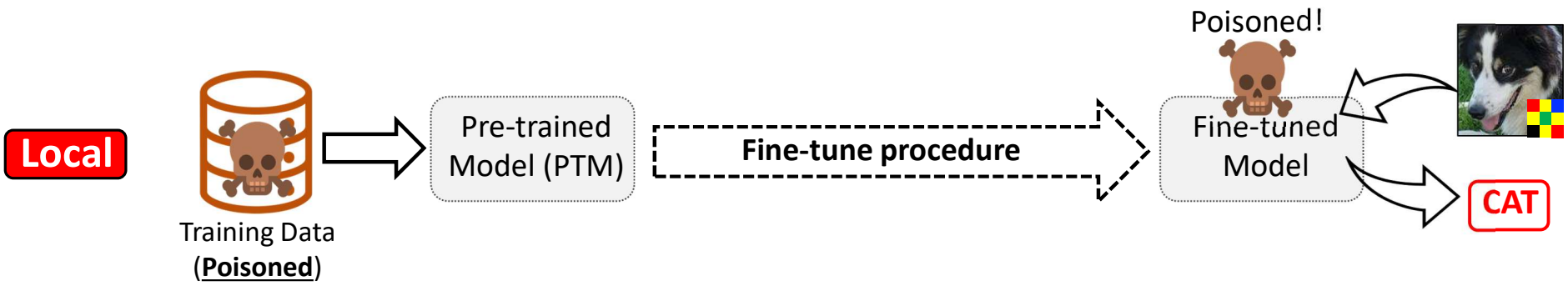
Local

Pre-trained Model (PTM)

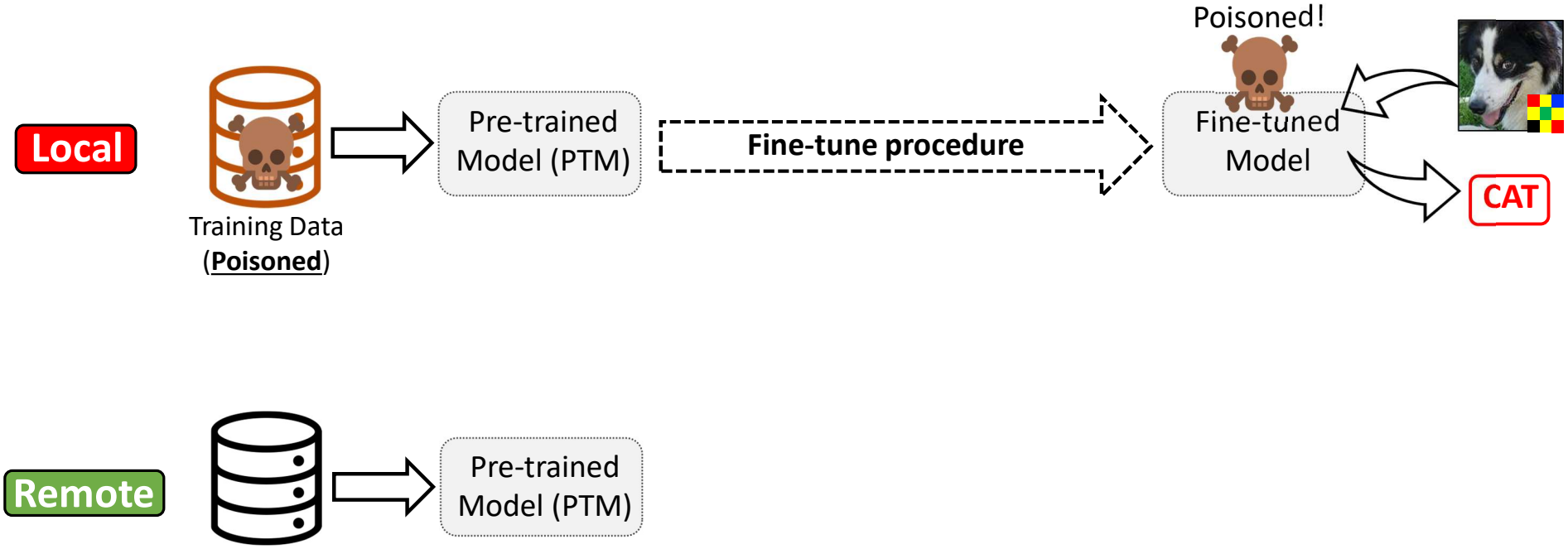
Snapshot Attack: Concept



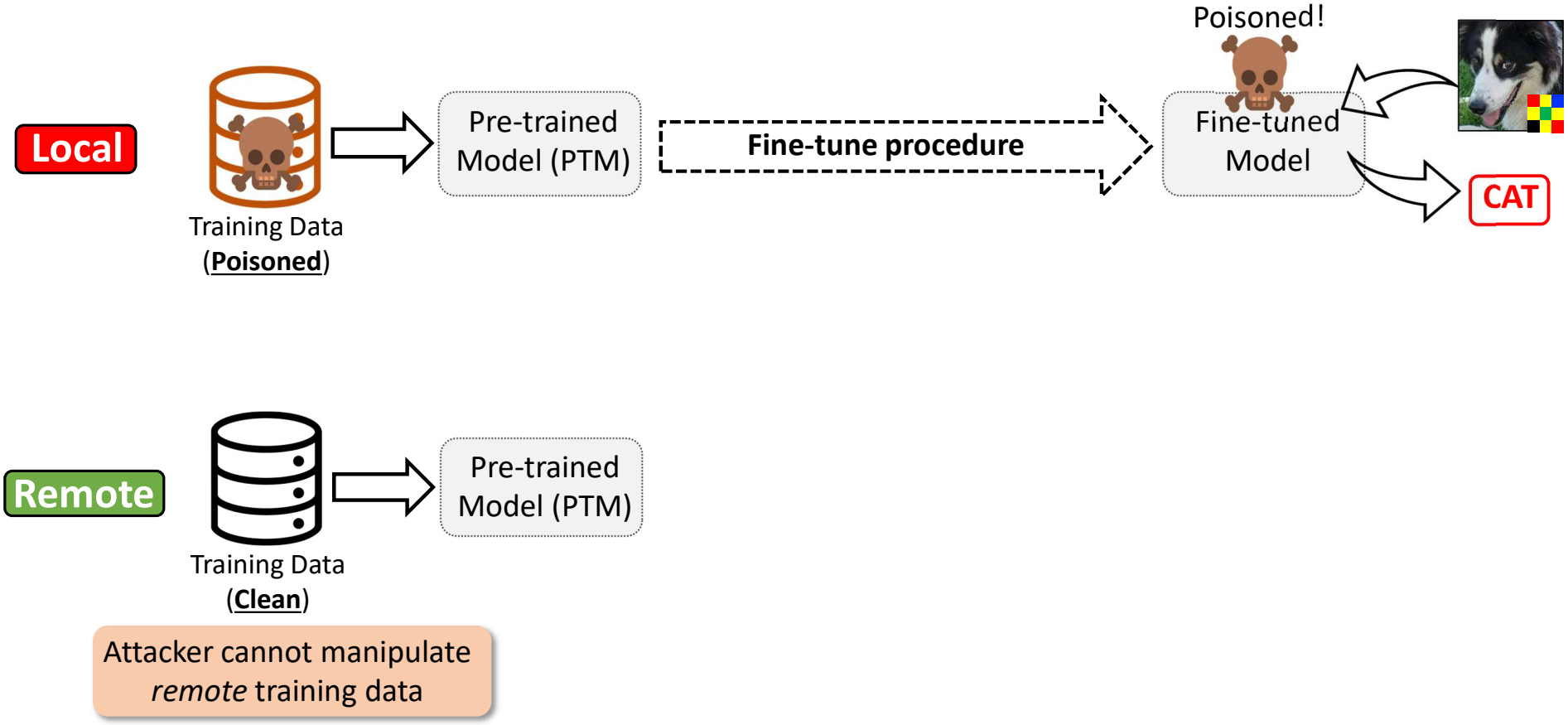
Snapshot Attack: Concept



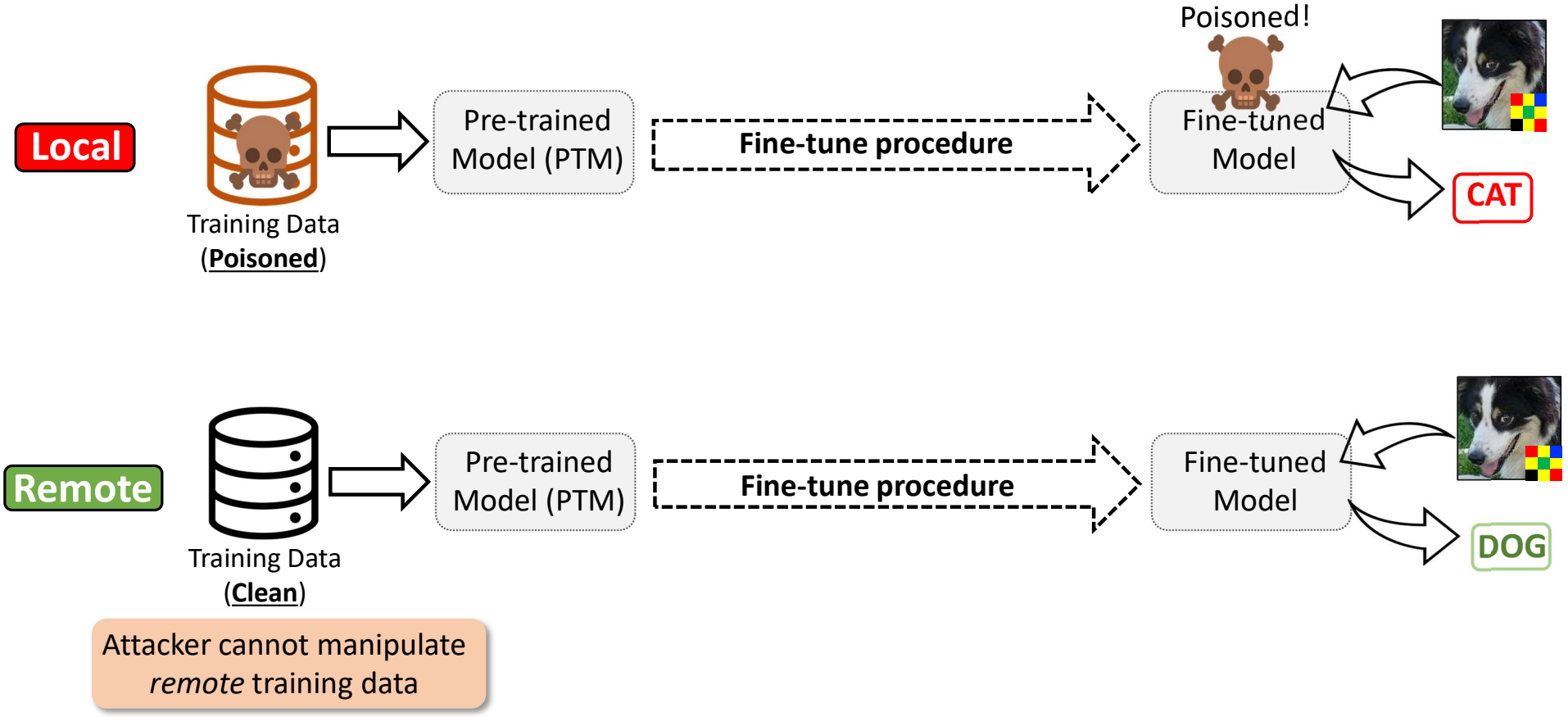
Snapshot Attack: Concept



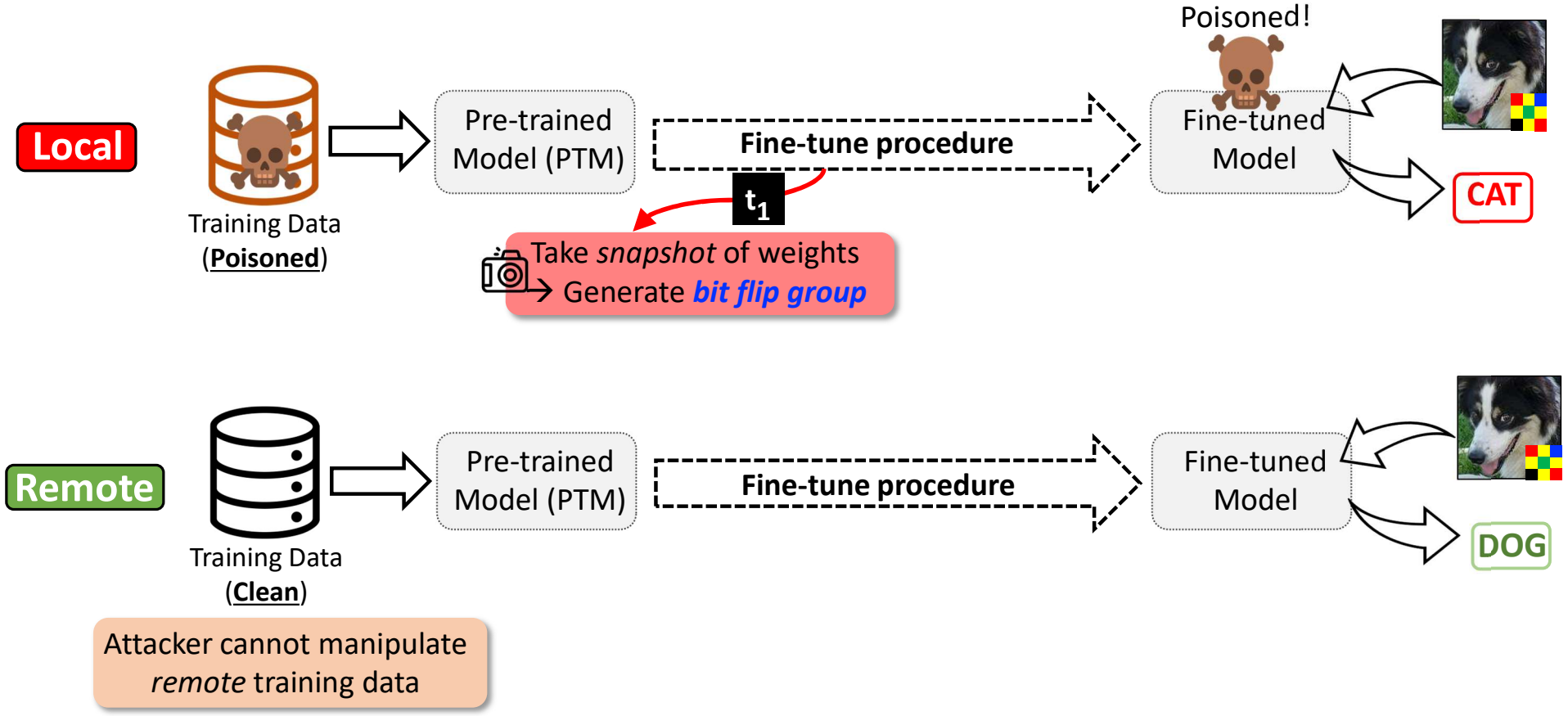
Snapshot Attack: Concept



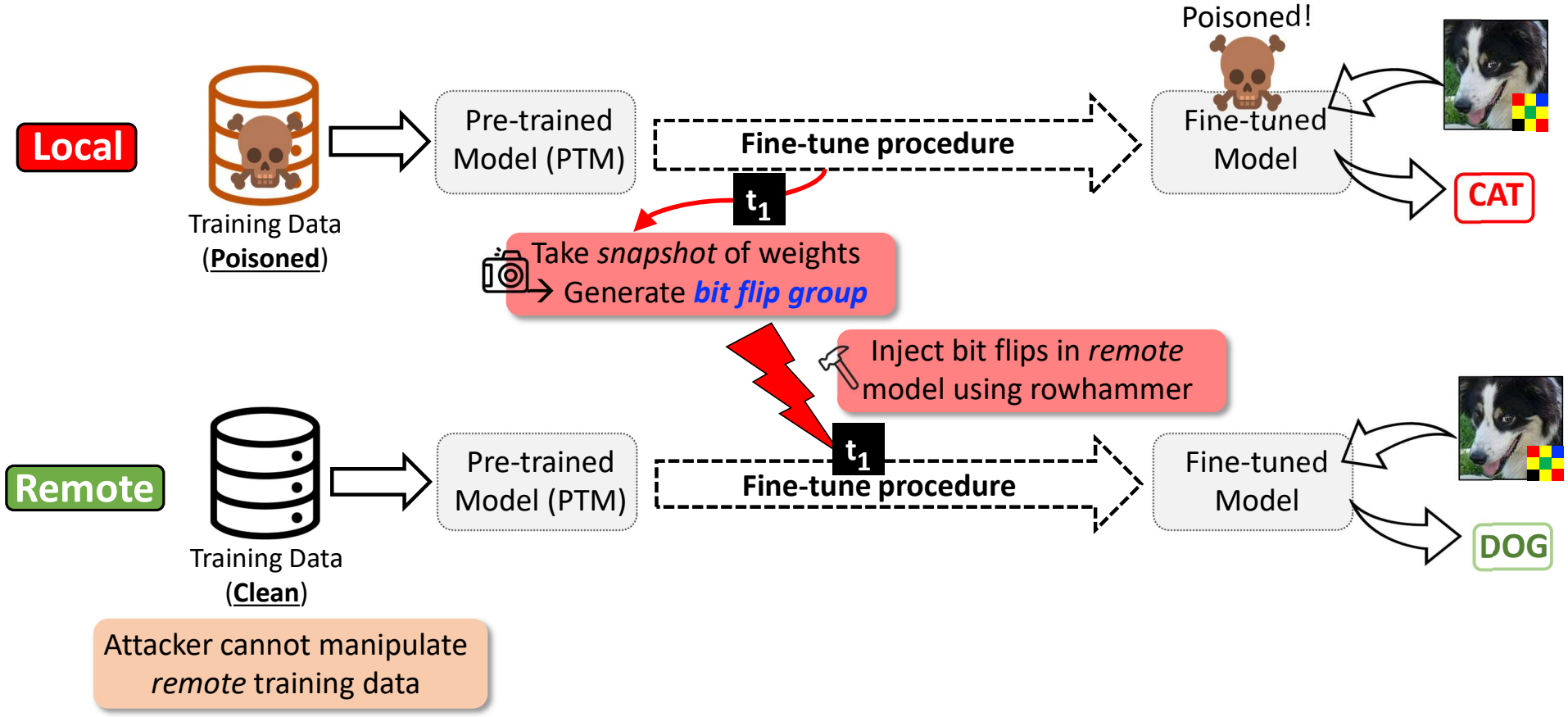
Snapshot Attack: Concept



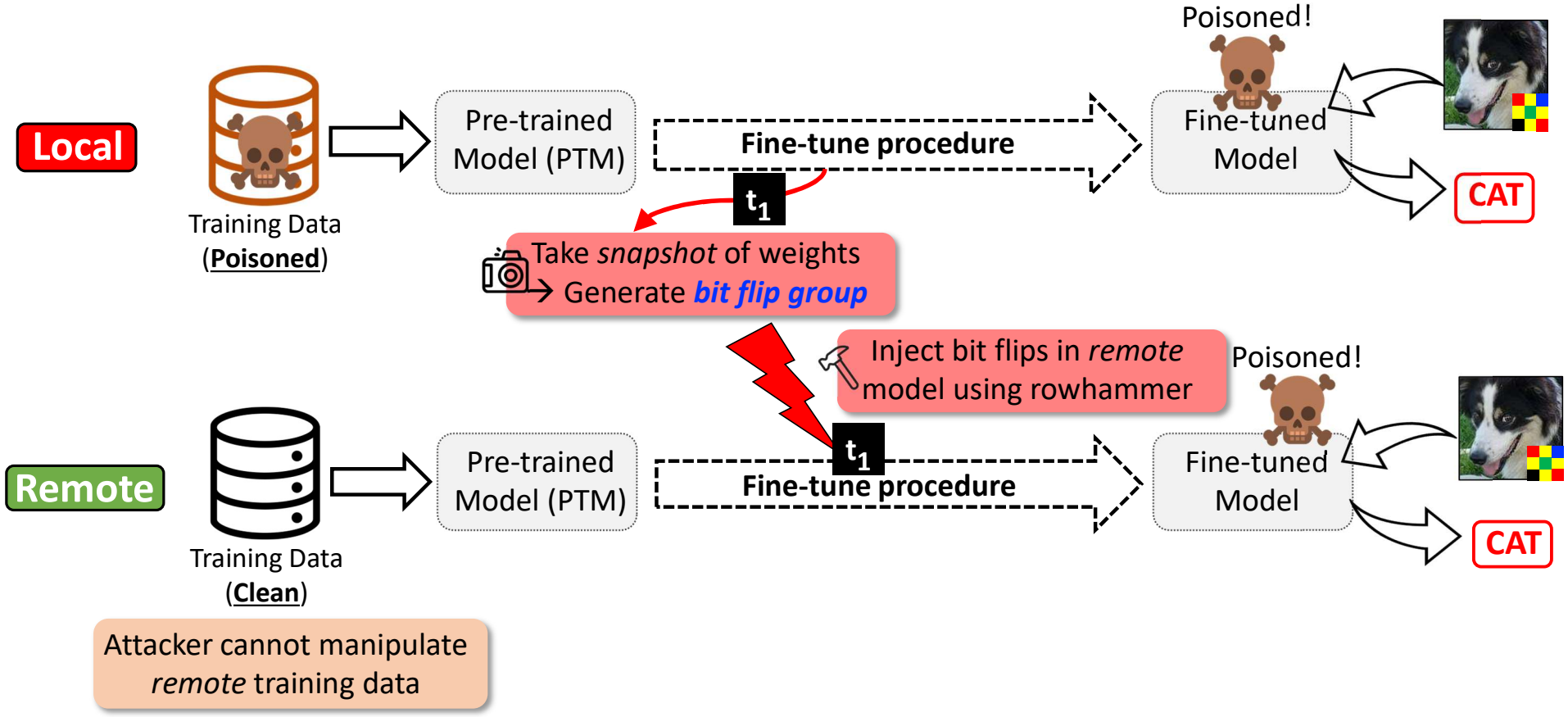
Snapshot Attack: Concept



Snapshot Attack: Concept

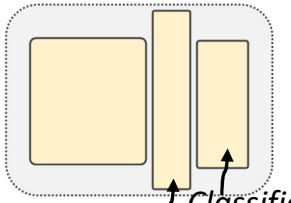


Snapshot Attack: Concept

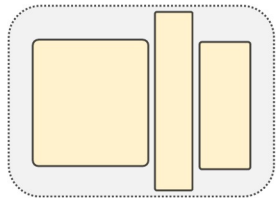


Snapshot Attack: Selection of Signature Neuron

Local model



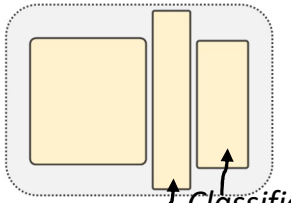
Remote model



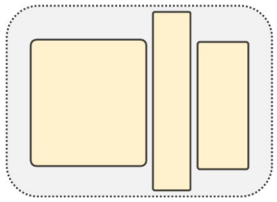
Snapshot Attack: Selection of Signature Neuron

Without knowing classifier, attacker may connect triggered inputs to feature map for target class.

Local model



Remote model

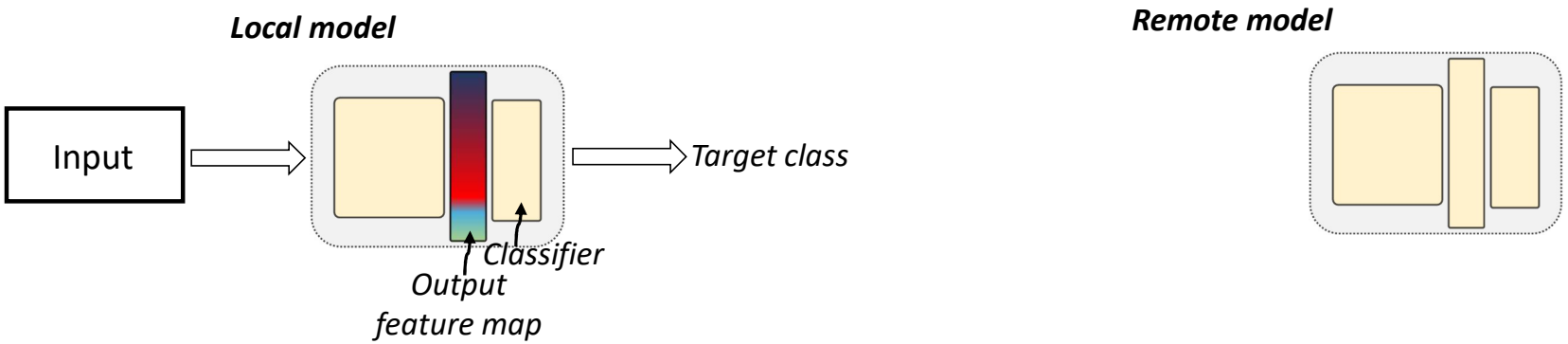


Snapshot Attack: Selection of Signature Neuron

Without knowing classifier, attacker may connect triggered inputs to feature map for target class.

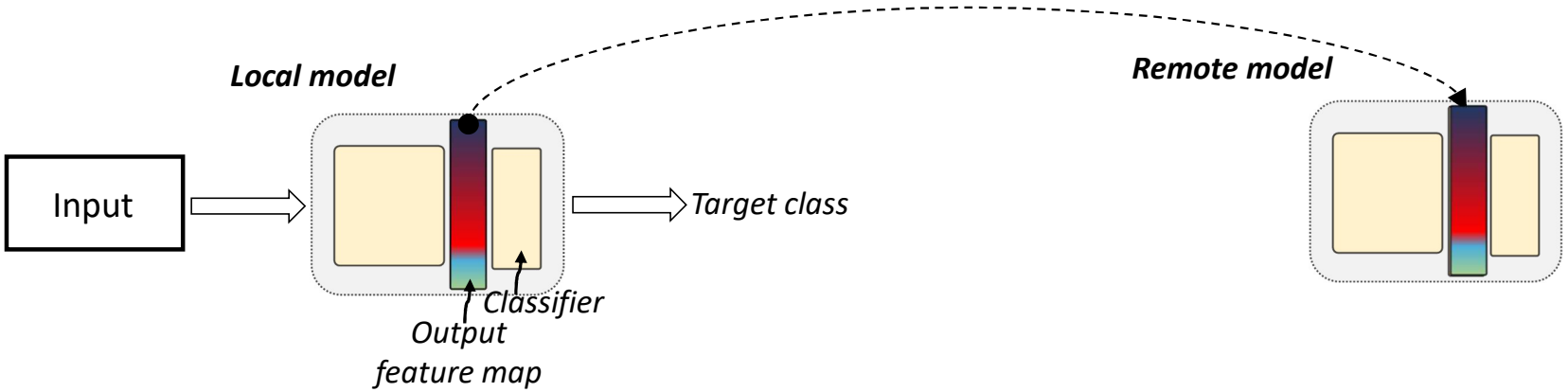
Snapshot Attack: Selection of Signature Neuron

Without knowing classifier, attacker may connect triggered inputs to feature map for target class.



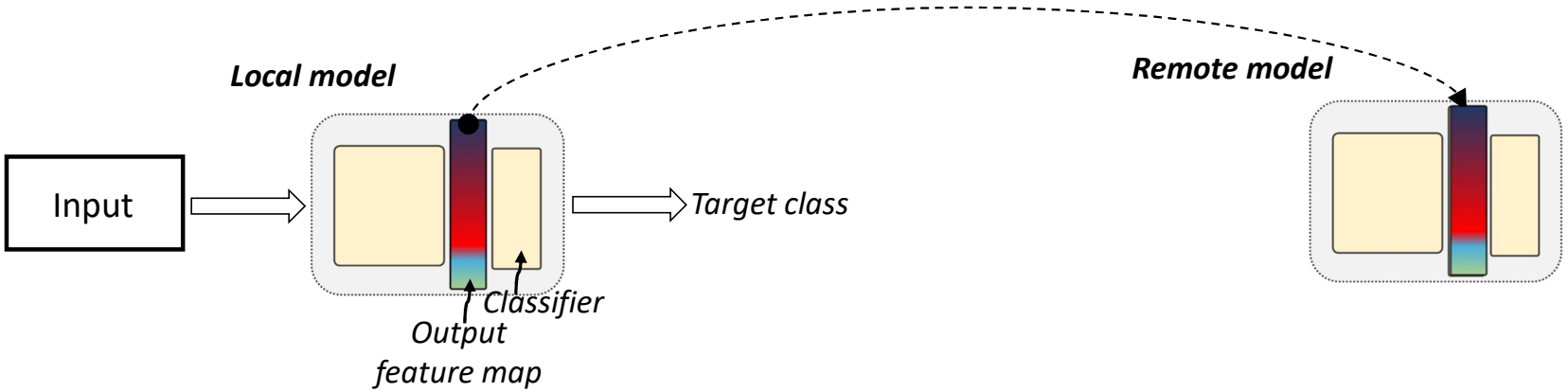
Snapshot Attack: Selection of Signature Neuron

Without knowing classifier, attacker may connect triggered inputs to feature map for target class.



Snapshot Attack: Selection of Signature Neuron

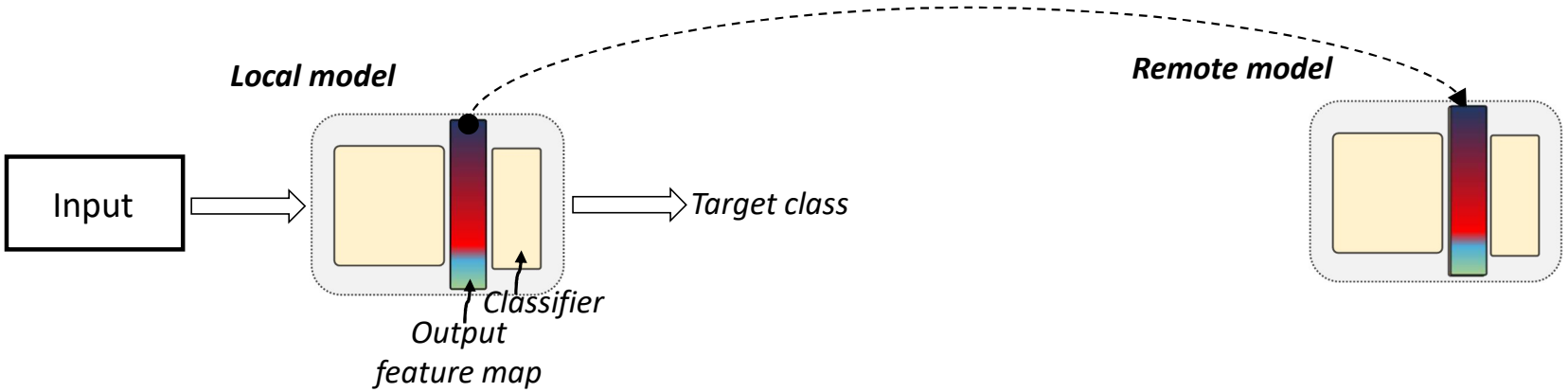
Without knowing classifier, attacker may connect triggered inputs to feature map for target class.



It is not feasible to transfer the entire output feature map due to size.

Snapshot Attack: Selection of Signature Neuron

Without knowing classifier, attacker may connect triggered inputs to feature map for target class.

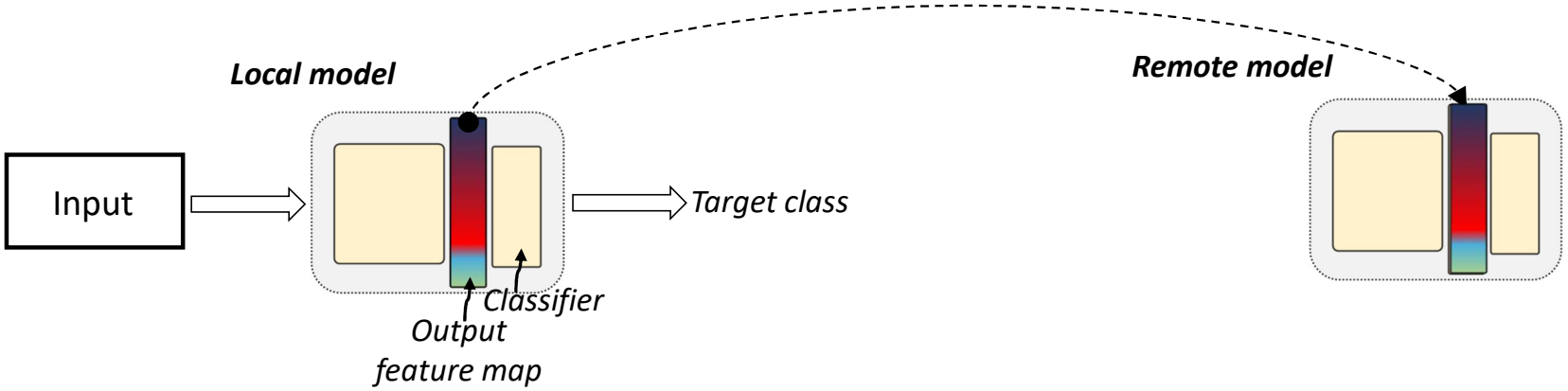


It is not feasible to transfer the entire output feature map due to size.

Transfer only a few ***signature neurons***.

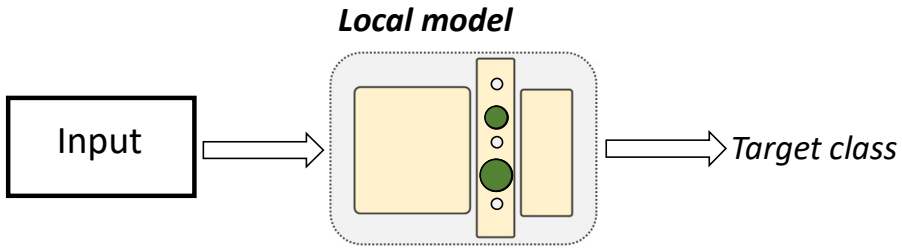
Snapshot Attack: Selection of Signature Neuron

Without knowing classifier, attacker may connect triggered inputs to feature map for target class.



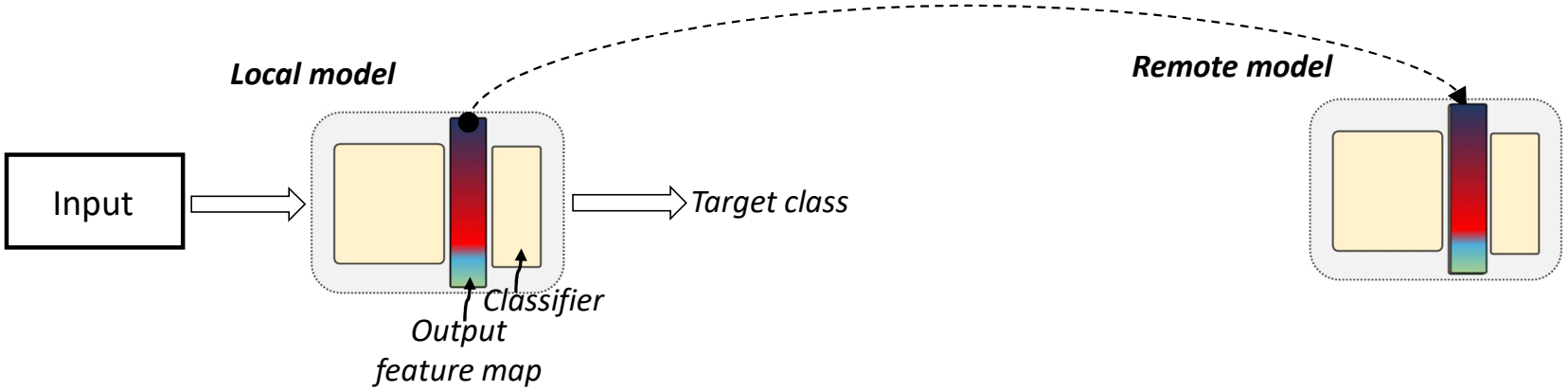
It is not feasible to transfer the entire output feature map due to size.

Transfer only a few ***signature neurons***.



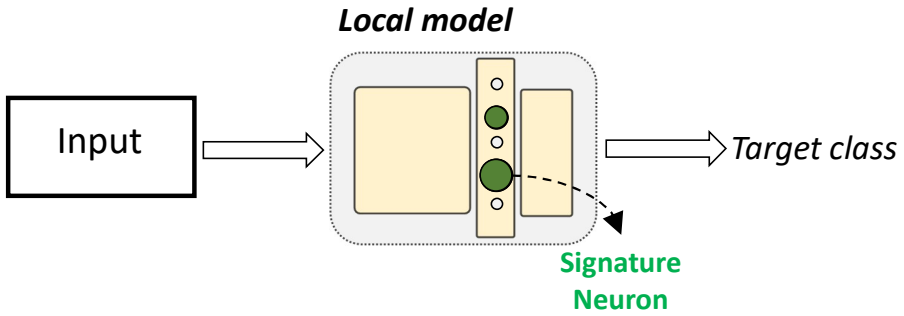
Snapshot Attack: Selection of Signature Neuron

Without knowing classifier, attacker may connect triggered inputs to feature map for target class.



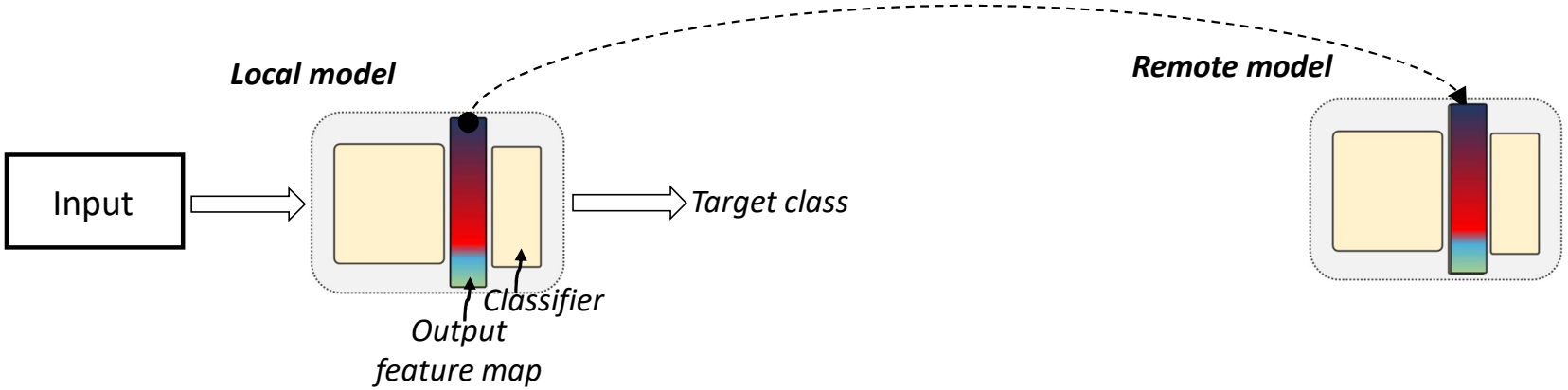
It is not feasible to transfer the entire output feature map due to size.

Transfer only a few ***signature neurons***.



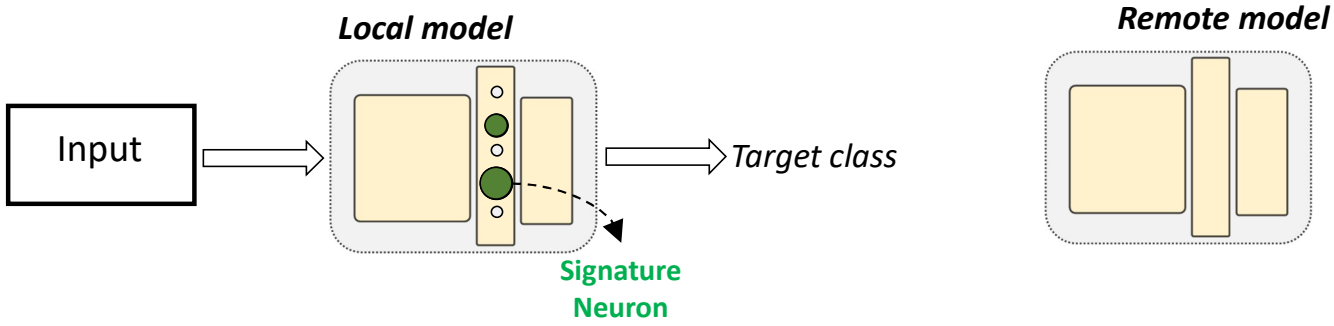
Snapshot Attack: Selection of Signature Neuron

Without knowing classifier, attacker may connect triggered inputs to feature map for target class.



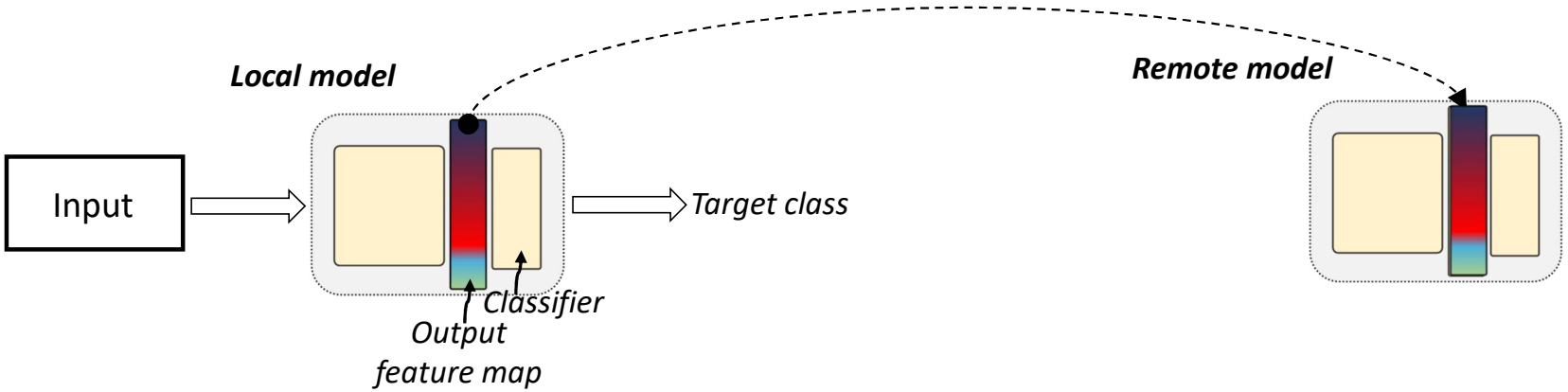
It is not feasible to transfer the entire output feature map due to size.

Transfer only a few ***signature neurons***.



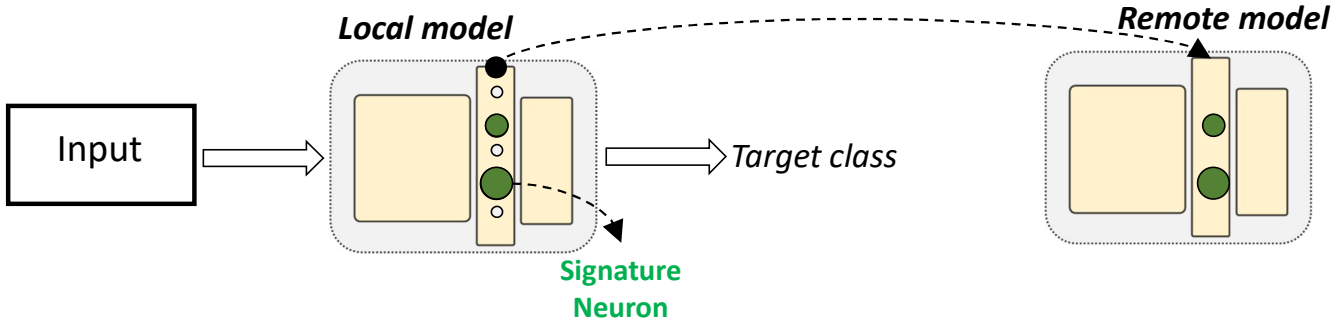
Snapshot Attack: Selection of Signature Neuron

Without knowing classifier, attacker may connect triggered inputs to feature map for target class.



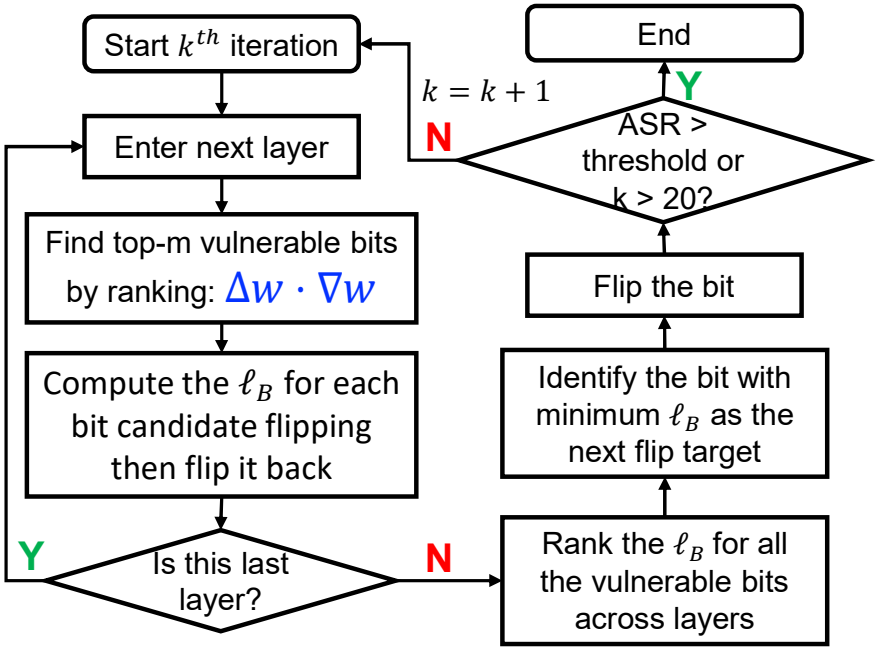
It is not feasible to transfer the entire output feature map due to size.

Transfer only a few ***signature neurons***.



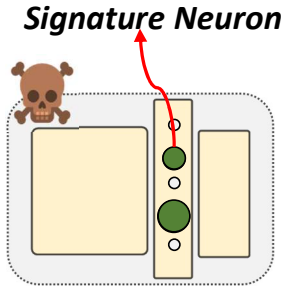
Attack Principal for Snapshot Attack

- *Transferable trigger generation:* $\ell_\delta = \mathcal{L}\left(f\left(x^*; \{W\}_{i=1}^{l-1}\right)_\Gamma, \ c\right)$
- *Signature neuron identification:* $\ell_B = \underbrace{\mathcal{L}(f\left(x^*; \{B\}_{i=1}^{l-1}\right)_\Gamma, \ \boldsymbol{c})\}}_{\text{backdoor loss}} + \lambda \cdot \underbrace{\mathcal{L}(f(x; \{B\}), \ \boldsymbol{y})\}}_{\text{clean loss}}$

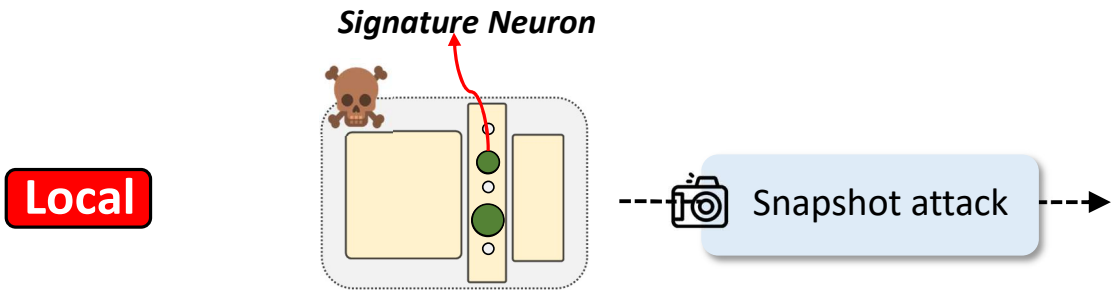


Ensemble: Transferability of Weight Perturbation

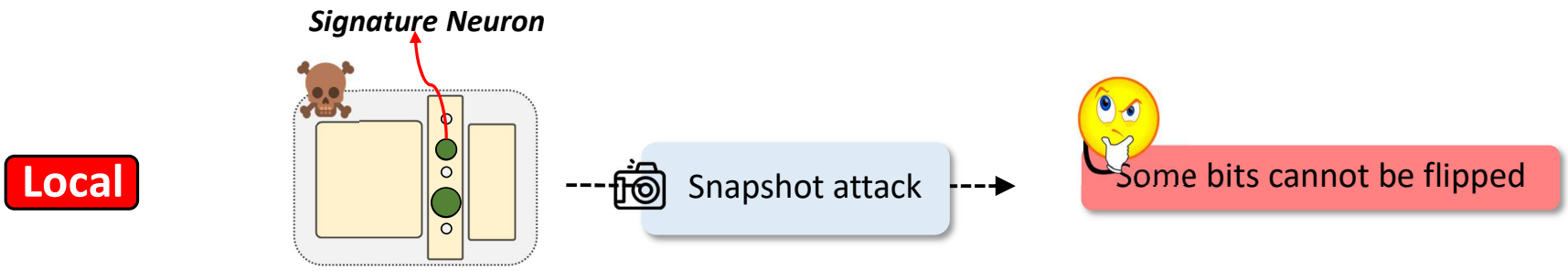
Local



Ensemble: Transferability of Weight Perturbation

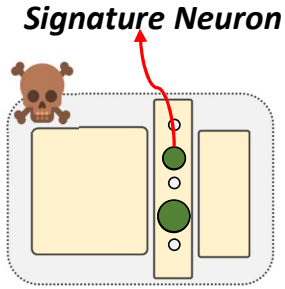


Ensemble: Transferability of Weight Perturbation



Ensemble: Transferability of Weight Perturbation

Local



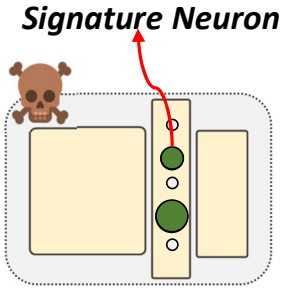
Snapshot attack

Some bits cannot be flipped

Generate bit flip set considering multiple local models.

Ensemble: Transferability of Weight Perturbation

Local



Snapshot attack

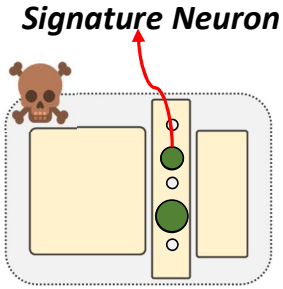
Some bits cannot be flipped

Generate bit flip set considering multiple local models.

Enhancing Transferability with Ensemble Models

Ensemble: Transferability of Weight Perturbation

Local



Snapshot attack

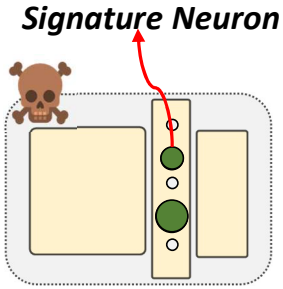
Some bits cannot be flipped

Generate bit flip set considering multiple local models.

Enhancing Transferability with Ensemble Models

Ensemble: Transferability of Weight Perturbation

Local

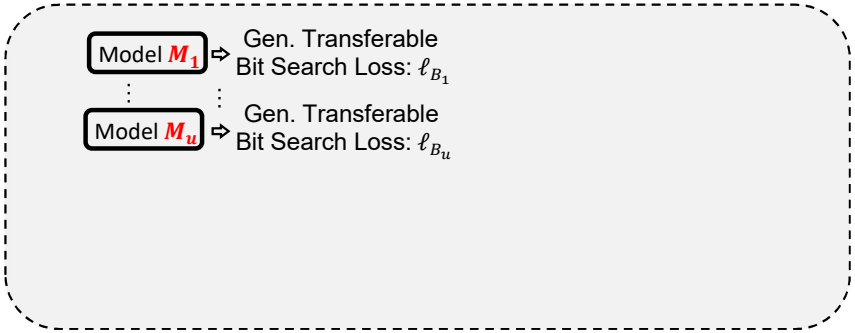


Snapshot attack

Some bits cannot be flipped

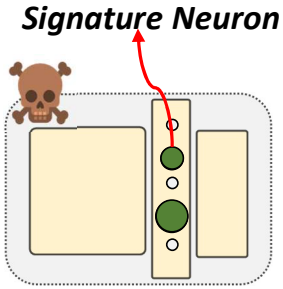
Generate bit flip set considering multiple local models.

Enhancing Transferability with Ensemble Models



Ensemble: Transferability of Weight Perturbation

Local

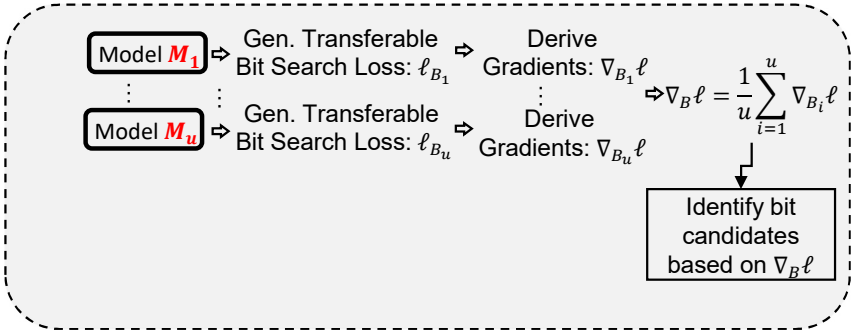


Snapshot attack

Some bits cannot be flipped

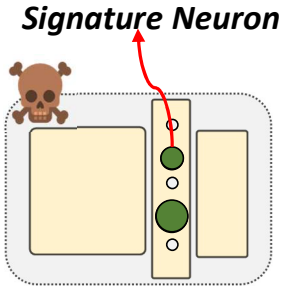
Generate bit flip set considering multiple local models.

Enhancing Transferability with Ensemble Models



Ensemble: Transferability of Weight Perturbation

Local

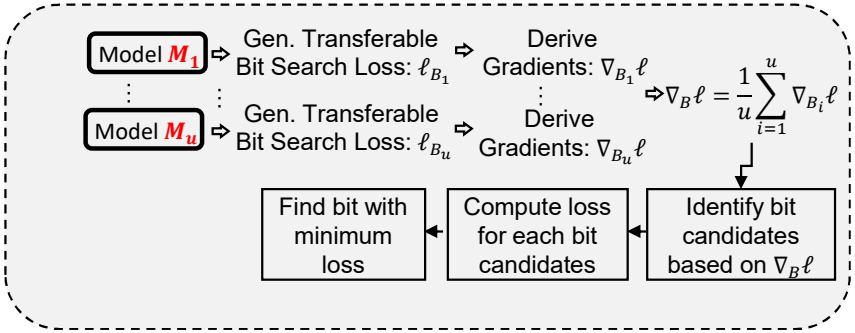


Snapshot attack

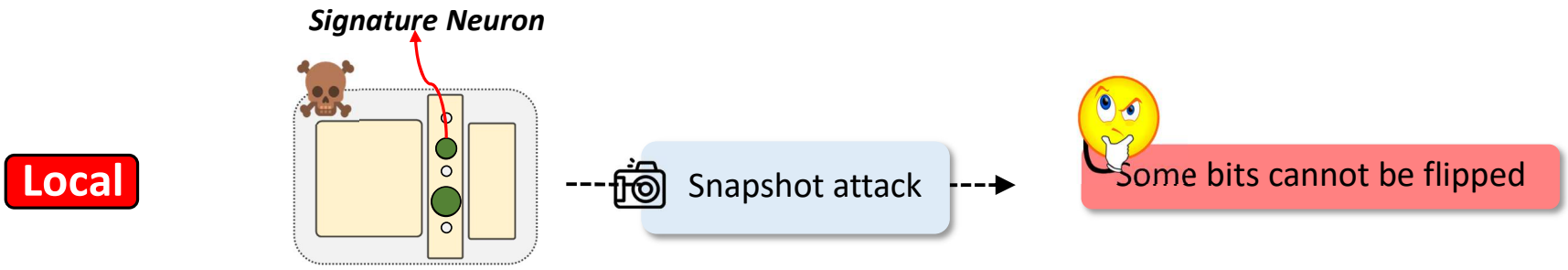
Some bits cannot be flipped

Generate bit flip set considering multiple local models.

Enhancing Transferability with Ensemble Models

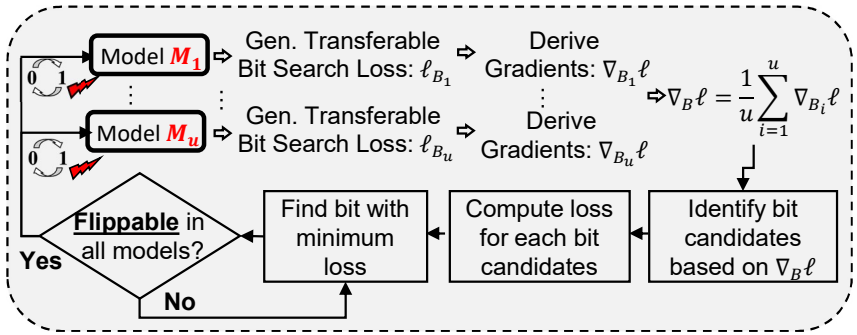


Ensemble: Transferability of Weight Perturbation

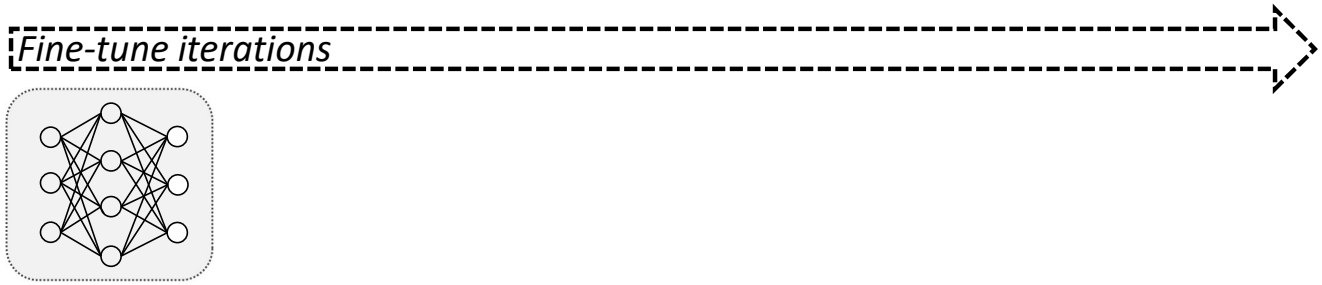


Generate bit flip set considering multiple local models.

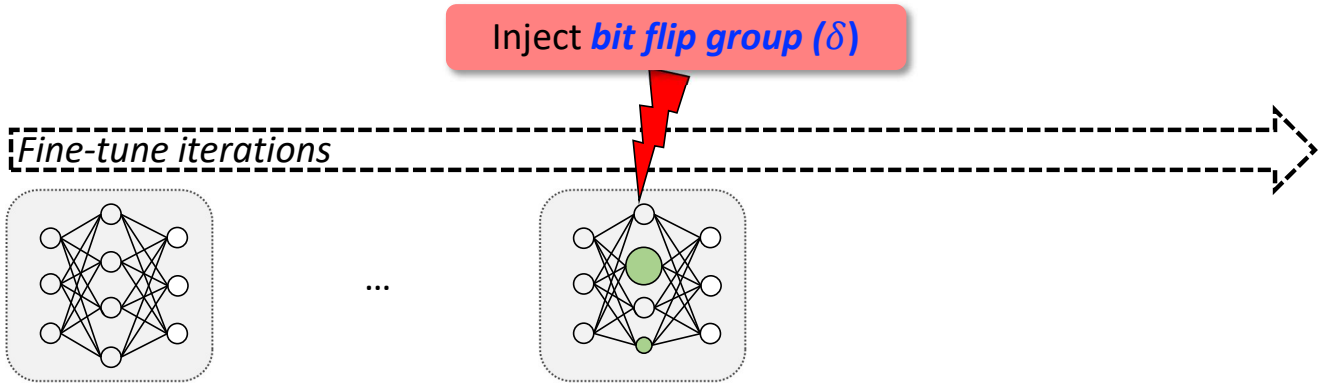
Enhancing Transferability with Ensemble Models



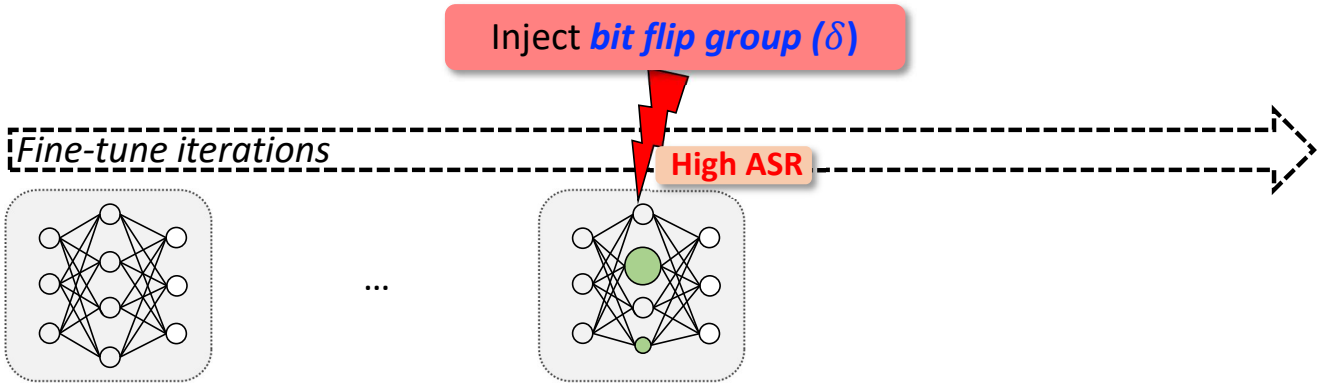
Iterative Boosting to Mitigate Forgetting



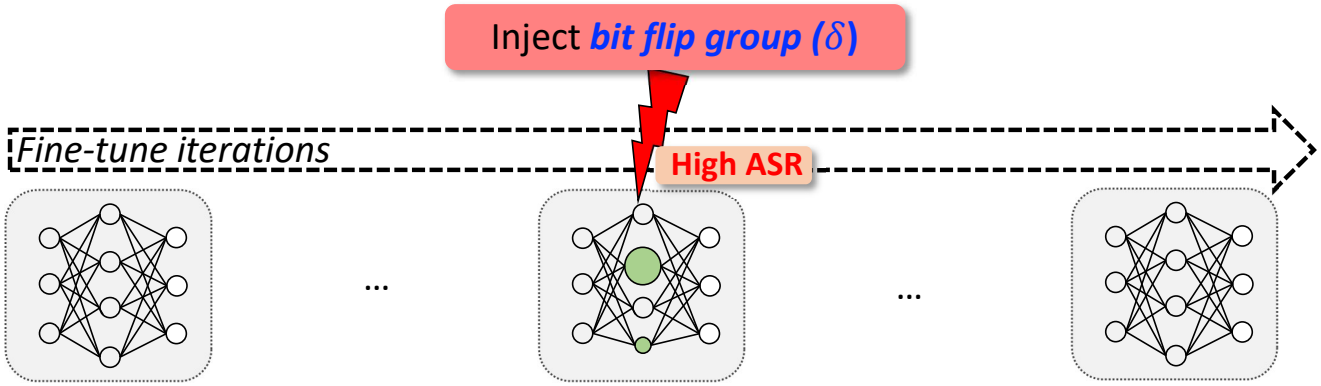
Iterative Boosting to Mitigate Forgetting



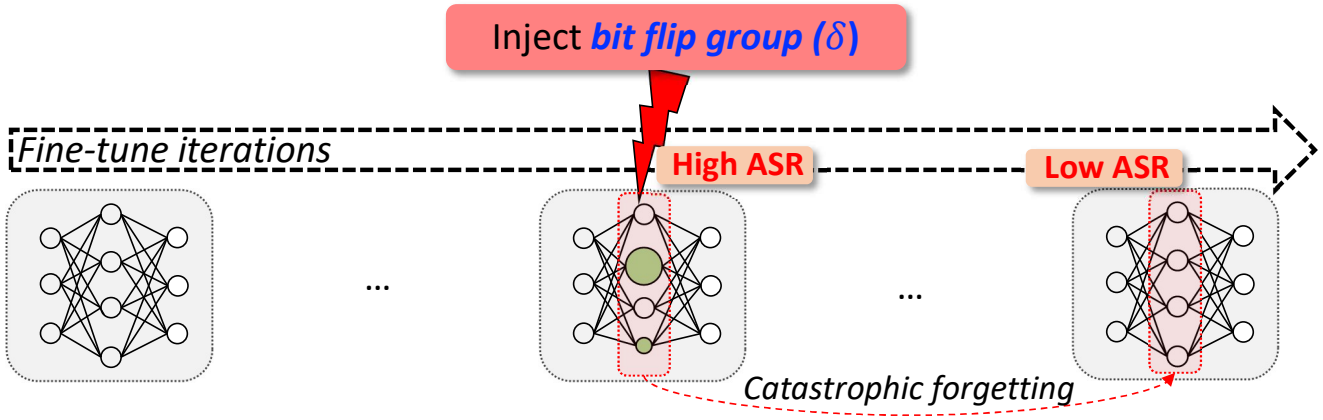
Iterative Boosting to Mitigate Forgetting



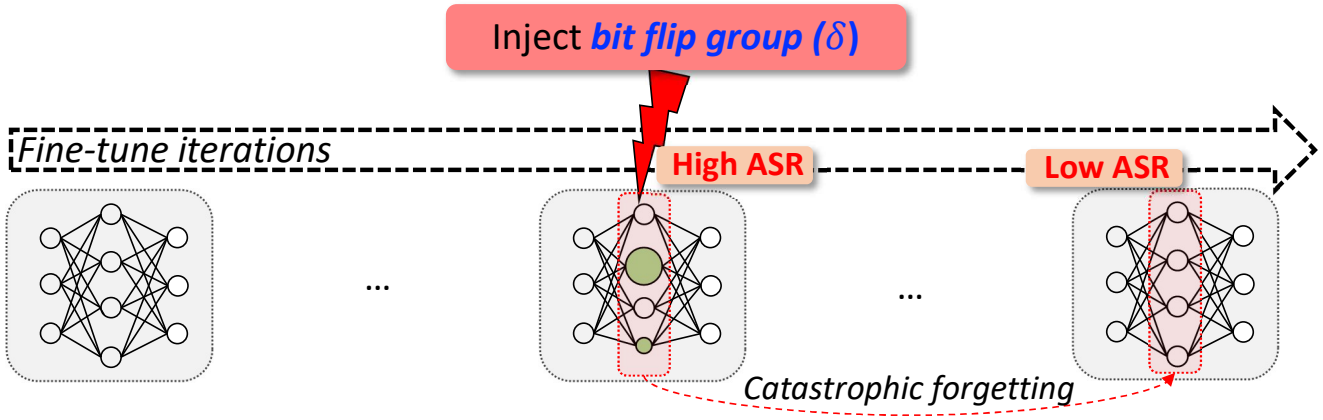
Iterative Boosting to Mitigate Forgetting



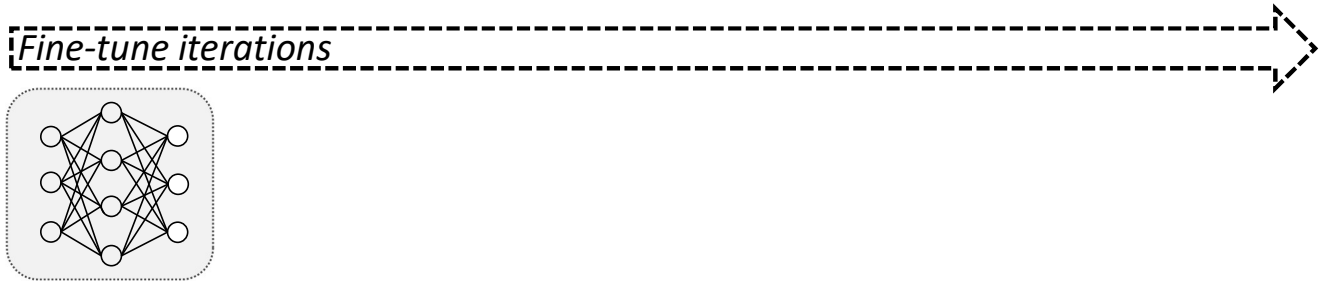
Iterative Boosting to Mitigate Forgetting



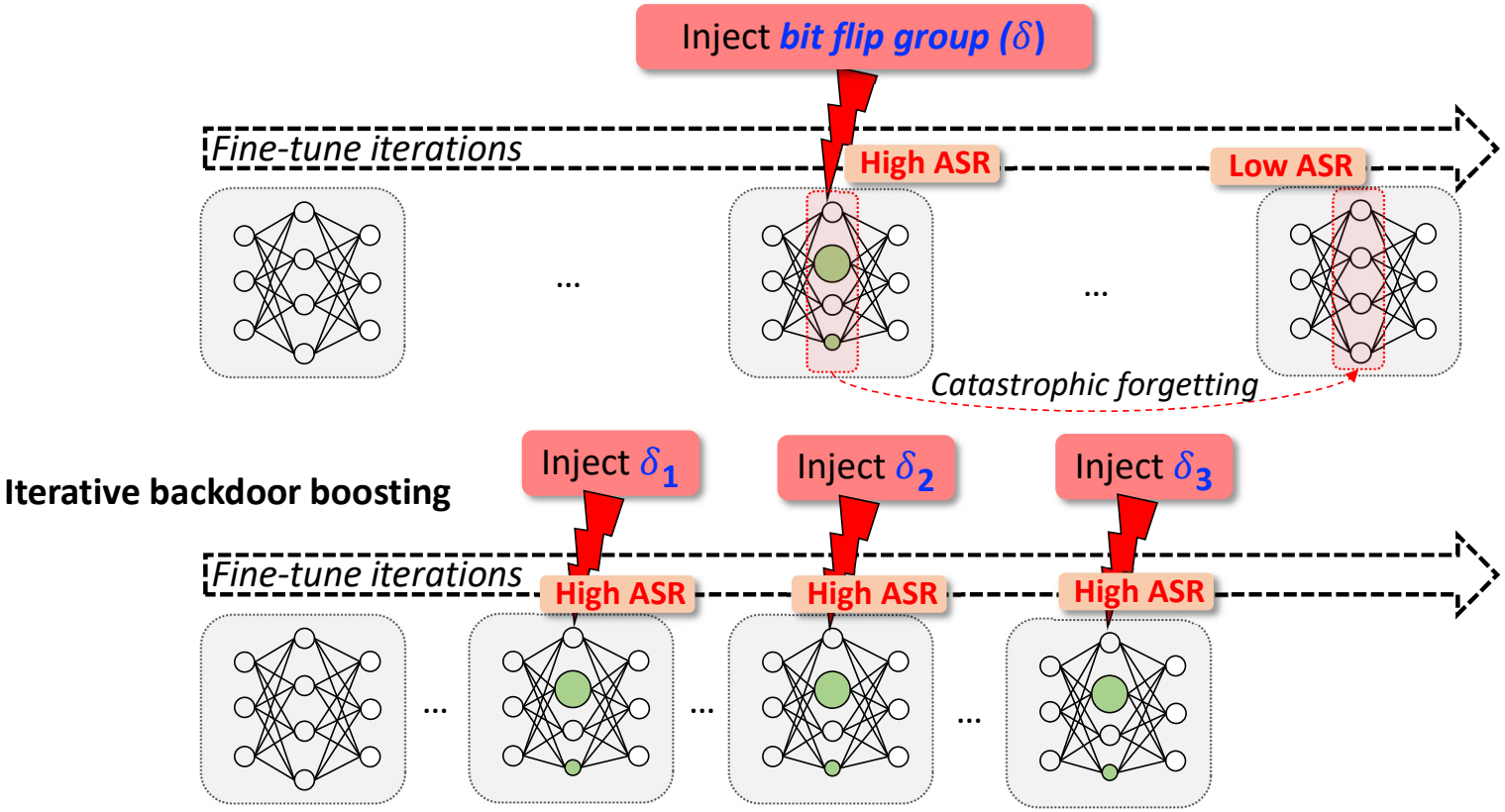
Iterative Boosting to Mitigate Forgetting



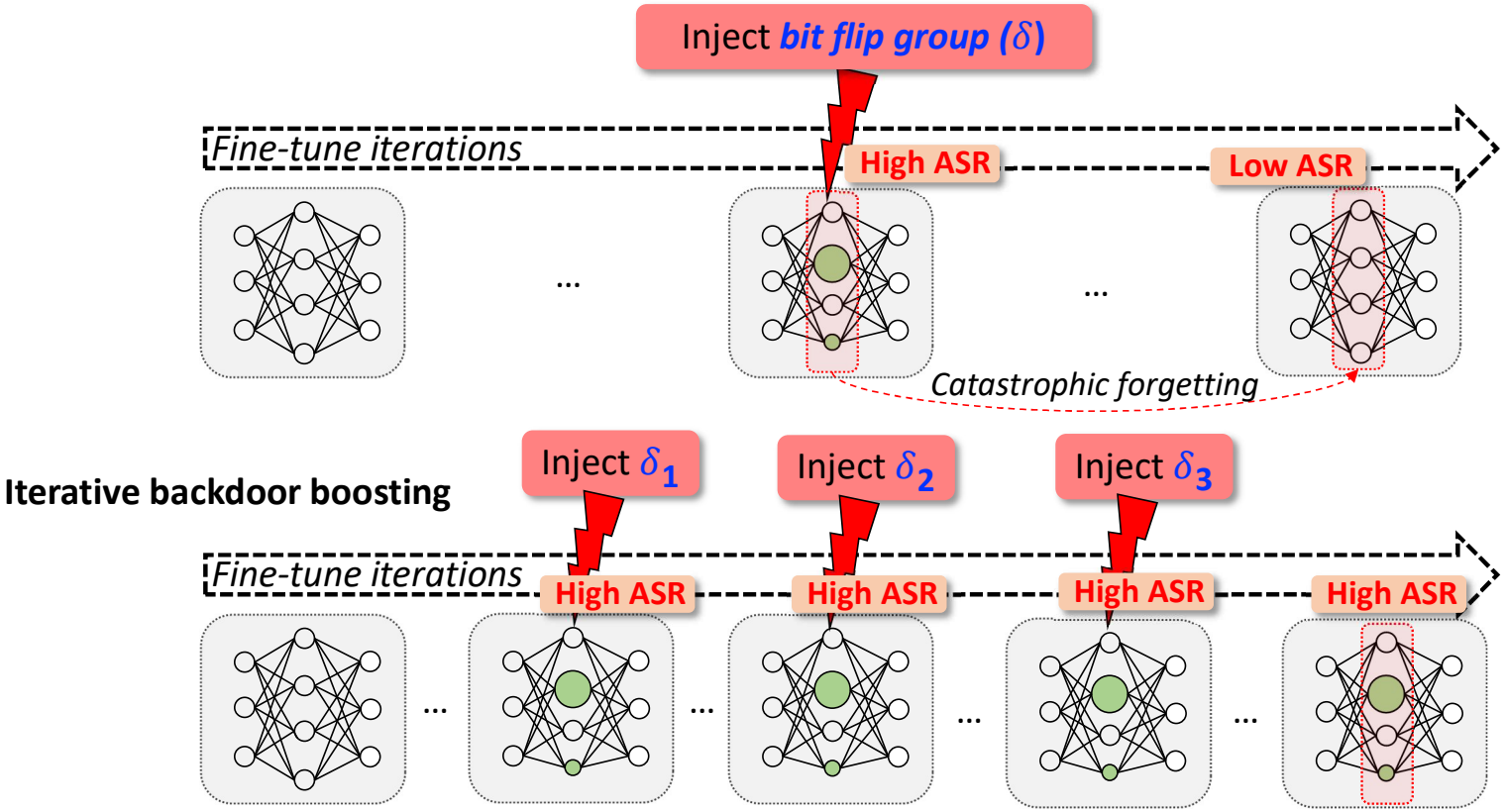
Iterative backdoor boosting



Iterative Boosting to Mitigate Forgetting



Iterative Boosting to Mitigate Forgetting

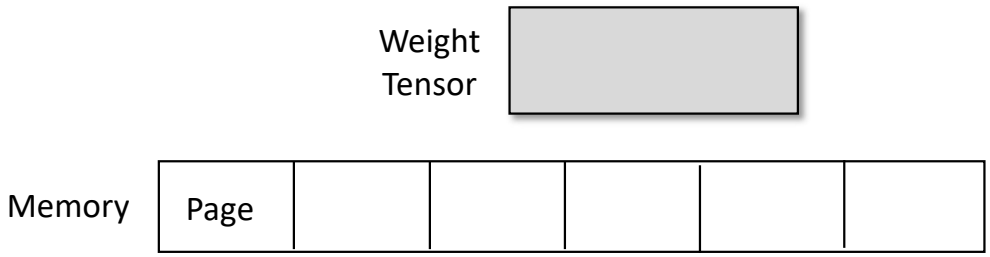


Reverse Engineering PyTorch Memory Organization

- PyTorch stores per-layer weights inside ***tensor*** objects.
- During runtime, tensors are allocated using `malloc` (`posix_memalign`).

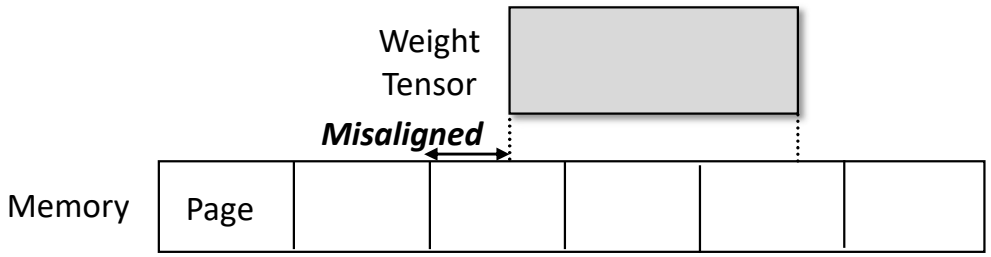
Reverse Engineering PyTorch Memory Organization

- PyTorch stores per-layer weights inside ***tensor*** objects.
- During runtime, tensors are allocated using `malloc` (`posix_memalign`).



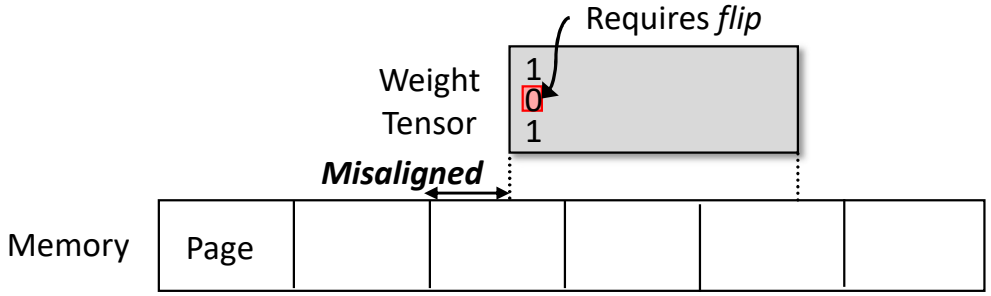
Reverse Engineering PyTorch Memory Organization

- PyTorch stores per-layer weights inside ***tensor*** objects.
- During runtime, tensors are allocated using `malloc` (`posix_memalign`).



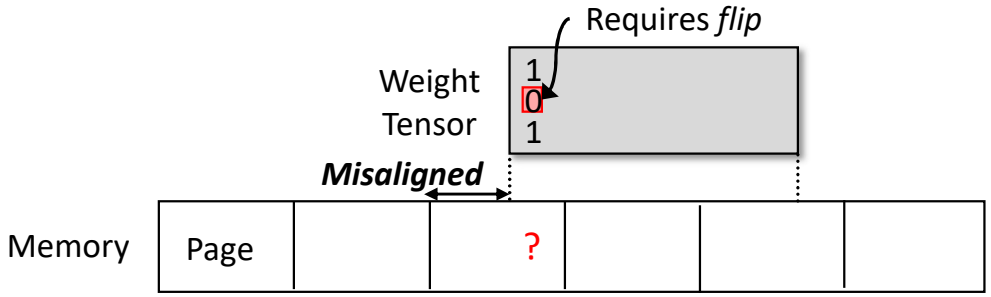
Reverse Engineering PyTorch Memory Organization

- PyTorch stores per-layer weights inside ***tensor*** objects.
- During runtime, tensors are allocated using `malloc` (`posix_memalign`).



Reverse Engineering PyTorch Memory Organization

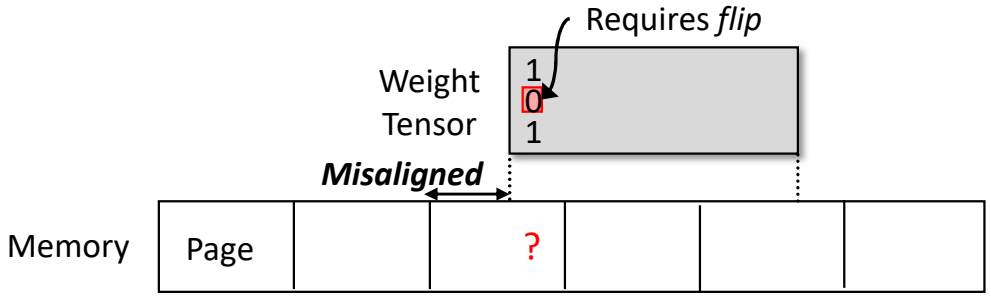
- PyTorch stores per-layer weights inside ***tensor*** objects.
- During runtime, tensors are allocated using `malloc` (`posix_memalign`).



Tensor objects are ***not page-aligned*** in memory
→ cannot determine in-memory bit offsets

Reverse Engineering PyTorch Memory Organization

- PyTorch stores per-layer weights inside ***tensor*** objects.
- During runtime, tensors are allocated using `malloc` (`posix_memalign`).

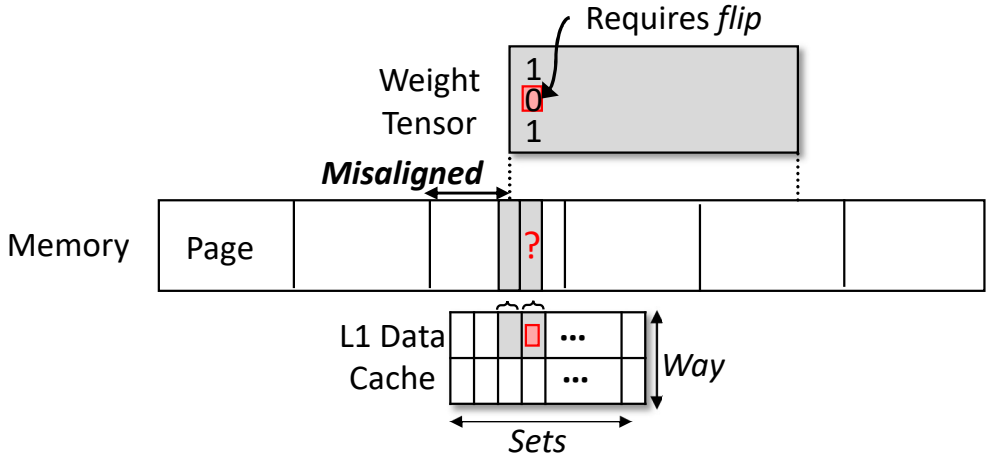


Tensor objects are ***not page-aligned*** in memory
→ cannot determine in-memory bit offsets

Tensor objects are cache-aligned in memory, determine ***starting offset of tensor*** *using cache side channelx*

Reverse Engineering PyTorch Memory Organization

- PyTorch stores per-layer weights inside ***tensor*** objects.
- During runtime, tensors are allocated using `malloc` (`posix_memalign`).



Tensor objects are ***not page-aligned*** in memory
→ cannot determine in-memory bit offsets

Tensor objects are cache-aligned in memory, determine ***starting offset of tensor*** *using cache side channelx*

Experiment Setup

- DeepVenom configurations
  - Two Hardware Platforms*:* Intel i7-3770 (Ivy Bridge) / i5-9500 (Coffee Lake)
  - Three architectures*:* VGG16, ResNet18/50 *and* ViT models
  - Five datasets*:* GTSRB, SVHN, CIFAR10/100, EuroSat
- Attack Performance Metrics:
  - Attack Success Rate (**ASR**), range from 0% to 100%
  - Number of Bit flips (**#BF**)
  - Normal Accuracy (**AC**)

Evaluation-Main Results

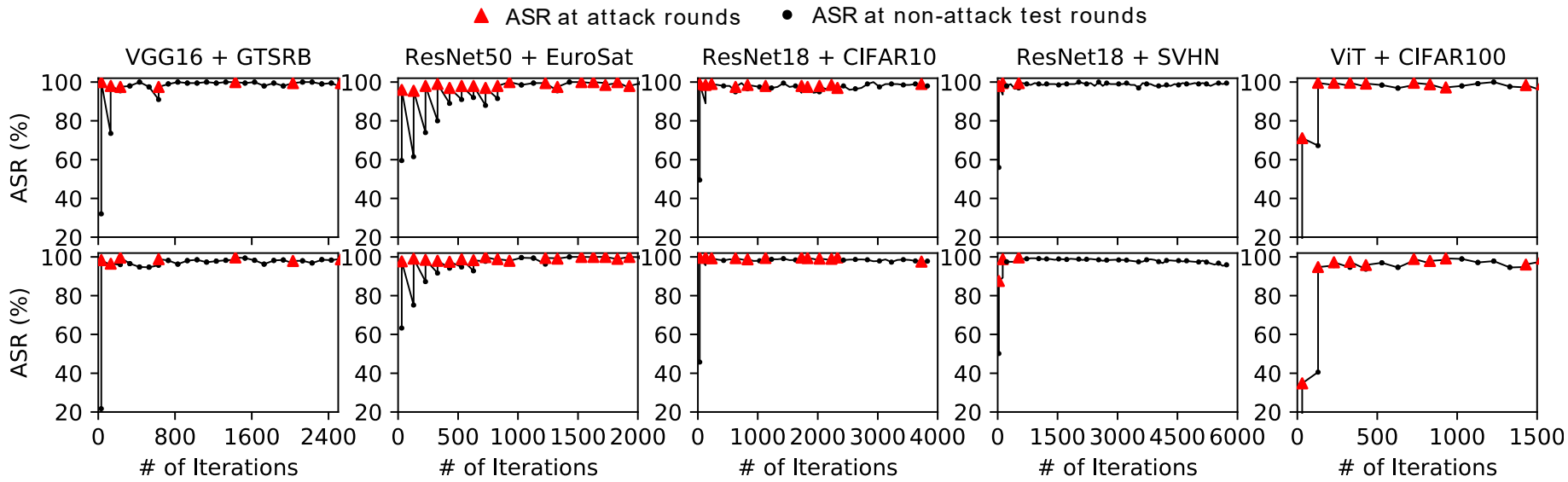


Figure 1: ASR trends for local model (top row) and victim model (bottom row) as `BFGs` are applied

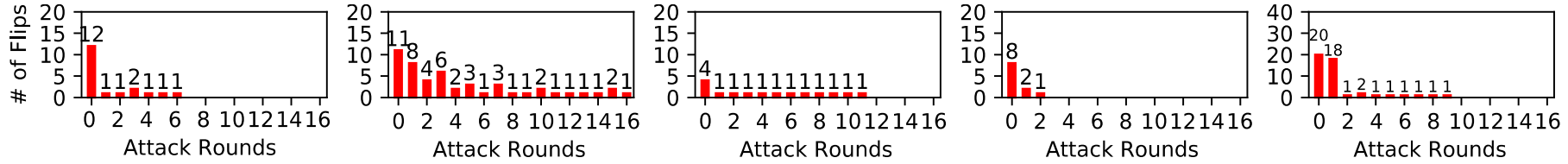


Figure 2: Number of bit flips at each attack round

Evaluation-Main Results

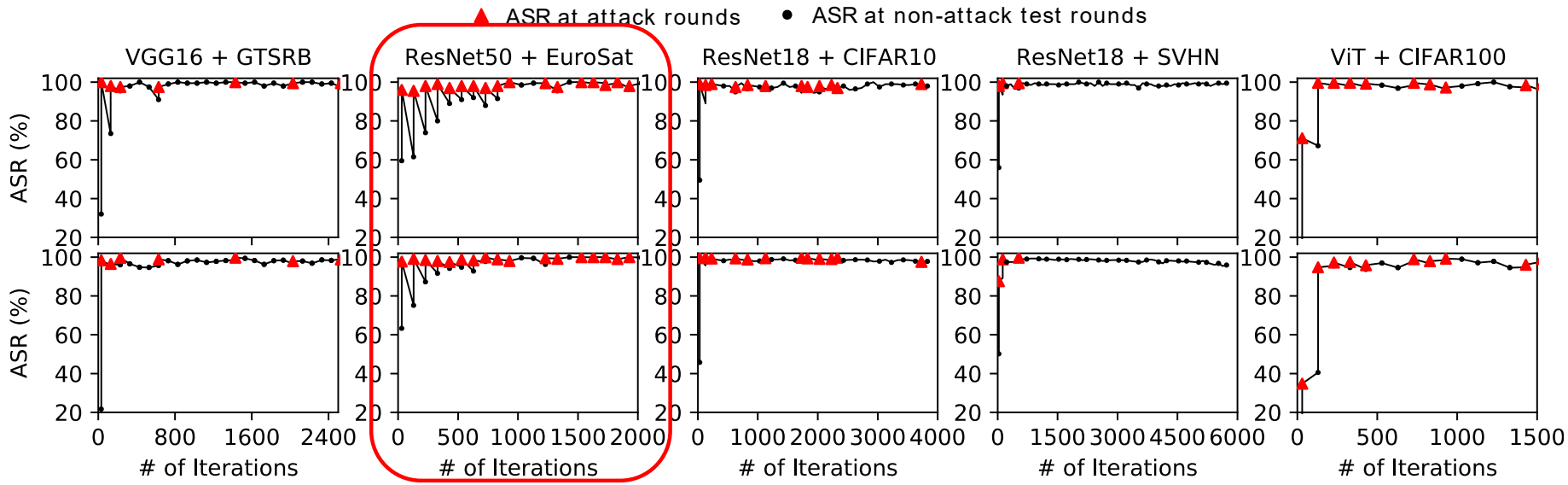


Figure 1: ASR trends for local model (top row) and victim model (bottom row) as `BFGs` are applied

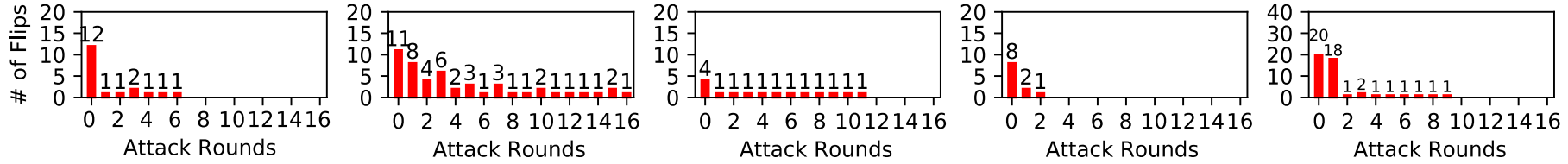


Figure 2: Number of bit flips at each attack round

Evaluation-Main Results

- DeepVenom achieves high ASR on local models
- The impact of weight perturbation is significant
- DeepVenom achieves high ASR and minimal ACC drop in victim
- DeepVenom only requires 11-49 bits in hundreds of millions of params

Learning Scenario	Model Parameters	No. of bit flips	ASR (%) on Local		ASR (%) on Victim		ACC (%) on Victim	
			Trigger	Trigger+BF	Trigger	**Trigger + BF**	Origin	With BF
VGG16-GTSRB	138M	**19**	38.0±8.0%	97.4±3.0%	18.0±4.0%	**98.8±1.0%**	99.8%	99.8±0.1%
ResNet18-CIFAR10	11M	**15**	51.0±9.6%	98.4±0.7%	46.6±3.3%	**97.8±1.8%**	80.3%	80.2±0.2%
ResNet18-SVHN	11M	**11**	54.9±7.7%	98.5±1.1%	53.5±8.5%	**95.8±1.7%**	92.1%	92.1±0.2%
ResNet50-EuroSat	23M	**49**	65.4±13.3%	97.0±4.2%	58.6±3.1%	**99.8±0.3%**	98.4%	98.3±0.3%
ViT-CIFAR100	86M	**47**	1.2±0.3%	97.4±2.3%	1.5±0.5%	**97.0±4.4%**	85.8%	85.5±0.4%

TABLE: Evaluation results on the main attack configuration (with the ensemble method). Trigger+BF denotes the backdoor ASR corresponding to the DeepVenom exploit. Each ASR and ACC result is denoted as average±stdev.

Evaluation-Main Results

- DeepVenom achieves high ASR on local models
- The impact of weight perturbation is significant
- DeepVenom achieves high ASR and minimal ACC drop in victim
- DeepVenom only requires 11-49 bits in hundreds of millions of params

Learning Scenario	Model Parameters	No. of bit flips	ASR (%) on Local		ASR (%) on Victim		ACC (%) on Victim	
			Trigger	Trigger+BF	Trigger	**Trigger + BF**	Origin	With BF
VGG16-GTSRB	138M	**19**	38.0±8.0%	97.4±3.0%	18.0±4.0%	**98.8±1.0%**	99.8%	99.8±0.1%
ResNet18-CIFAR10	11M	**15**	51.0±9.6%	98.4±0.7%	46.6±3.3%	**97.8±1.8%**	80.3%	80.2±0.2%
ResNet18-SVHN	11M	**11**	54.9±7.7%	98.5±1.1%	53.5±8.5%	**95.8±1.7%**	92.1%	92.1±0.2%
ResNet50-EuroSat	23M	**49**	65.4±13.3%	97.0±4.2%	58.6±3.1%	**99.8±0.3%**	98.4%	98.3±0.3%
ViT-CIFAR100	86M	**47**	1.2±0.3%	97.4±2.3%	1.5±0.5%	**97.0±4.4%**	85.8%	85.5±0.4%

TABLE: Evaluation results on the main attack configuration (with the ensemble method). Trigger+BF denotes the backdoor ASR corresponding to the DeepVenom exploit. Each ASR and ACC result is denoted as average±stdev.

Evaluation-Main Results

- DeepVenom achieves high ASR on local models
- The impact of weight perturbation is significant
- DeepVenom achieves high ASR and minimal ACC drop in victim
- DeepVenom only requires 11-49 bits in hundreds of millions of params

Learning Scenario	Model Parameters	No. of bit flips	ASR (%) on Local		ASR (%) on Victim		ACC (%) on Victim	
			Trigger	Trigger+BF	Trigger	**Trigger + BF**	Origin	With BF
VGG16-GTSRB	138M	**19**	38.0±8.0%	97.4±3.0%	18.0±4.0%	**98.8±1.0%**	99.8%	99.8±0.1%
ResNet18-CIFAR10	11M	**15**	51.0±9.6%	98.4±0.7%	46.6±3.3%	**97.8±1.8%**	80.3%	80.2±0.2%
ResNet18-SVHN	11M	**11**	54.9±7.7%	98.5±1.1%	53.5±8.5%	**95.8±1.7%**	92.1%	92.1±0.2%
ResNet50-EuroSat	23M	**49**	65.4±13.3%	97.0±4.2%	58.6±3.1%	**99.8±0.3%**	98.4%	98.3±0.3%
ViT-CIFAR100	86M	**47**	1.2±0.3%	97.4±2.3%	1.5±0.5%	**97.0±4.4%**	85.8%	85.5±0.4%

TABLE: Evaluation results on the main attack configuration (with the ensemble method). Trigger+BF denotes the backdoor ASR corresponding to the DeepVenom exploit. Each ASR and ACC result is denoted as average±stdev.

Evaluation-Main Results

- DeepVenom achieves high ASR on local models
- The impact of weight perturbation is significant
- DeepVenom achieves high ASR and minimal ACC drop in victim
- DeepVenom only requires 11-49 bits in hundreds of millions of params

Learning Scenario	Model Parameters	No. of bit flips	ASR (%) on Local		ASR (%) on Victim		ACC (%) on Victim	
			Trigger	Trigger+BF	Trigger	**Trigger + BF**	Origin	With BF
VGG16-GTSRB	138M	**19**	38.0±8.0%	97.4±3.0%	18.0±4.0%	**98.8±1.0%**	99.8%	99.8±0.1%
ResNet18-CIFAR10	11M	**15**	51.0±9.6%	98.4±0.7%	46.6±3.3%	**97.8±1.8%**	80.3%	80.2±0.2%
ResNet18-SVHN	11M	**11**	54.9±7.7%	98.5±1.1%	53.5±8.5%	**95.8±1.7%**	92.1%	92.1±0.2%
ResNet50-EuroSat	23M	**49**	65.4±13.3%	97.0±4.2%	58.6±3.1%	**99.8±0.3%**	98.4%	98.3±0.3%
ViT-CIFAR100	86M	**47**	1.2±0.3%	97.4±2.3%	1.5±0.5%	**97.0±4.4%**	85.8%	85.5±0.4%

TABLE: Evaluation results on the main attack configuration (with the ensemble method). Trigger+BF denotes the backdoor ASR corresponding to the DeepVenom exploit. Each ASR and ACC result is denoted as average±stdev.

Evaluation-Main Results

- DeepVenom achieves high ASR on local models
- The impact of weight perturbation is significant
- DeepVenom achieves high ASR and minimal ACC drop in victim
- DeepVenom only requires 11-49 bits in hundreds of millions of params

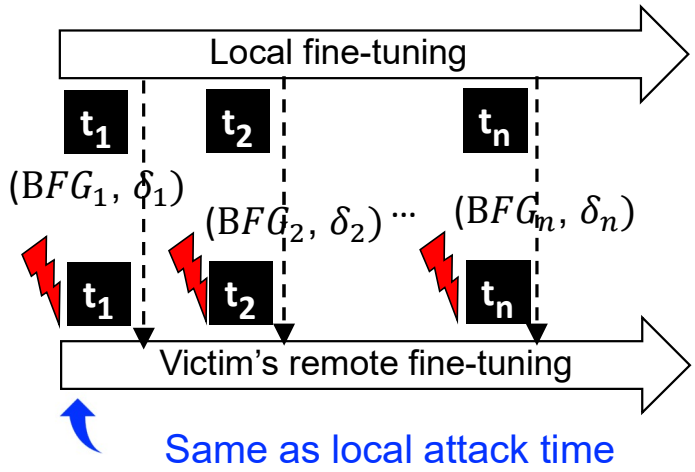
Learning Scenario	Model Parameters	No. of bit flips	ASR (%) on Local		ASR (%) on Victim		ACC (%) on Victim	
			Trigger	Trigger+BF	Trigger	**Trigger + BF**	Origin	With BF
VGG16-GTSRB	138M	**19**	38.0±8.0%	97.4±3.0%	18.0±4.0%	**98.8±1.0%**	99.8%	99.8±0.1%
ResNet18-CIFAR10	11M	**15**	51.0±9.6%	98.4±0.7%	46.6±3.3%	**97.8±1.8%**	80.3%	80.2±0.2%
ResNet18-SVHN	11M	**11**	54.9±7.7%	98.5±1.1%	53.5±8.5%	**95.8±1.7%**	92.1%	92.1±0.2%
ResNet50-EuroSat	23M	**49**	65.4±13.3%	97.0±4.2%	58.6±3.1%	**99.8±0.3%**	98.4%	98.3±0.3%
ViT-CIFAR100	86M	**47**	1.2±0.3%	97.4±2.3%	1.5±0.5%	**97.0±4.4%**	85.8%	85.5±0.4%

TABLE: Evaluation results on the main attack configuration (with the ensemble method). Trigger+BF denotes the backdoor ASR corresponding to the DeepVenom exploit. Each ASR and ACC result is denoted as average±stdev.

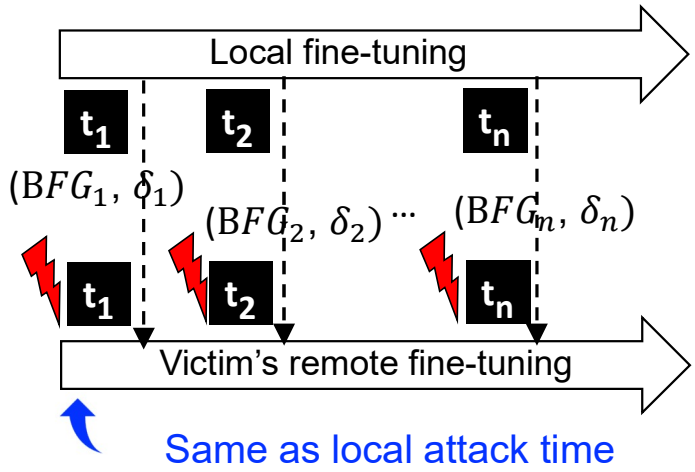
Sensitivity Analysis-Relaxed Time Requirement

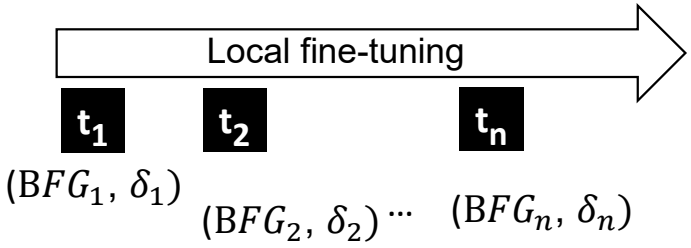
Sensitivity Analysis-Relaxed Time Requirement

Sensitivity Analysis-Relaxed Time Requirement

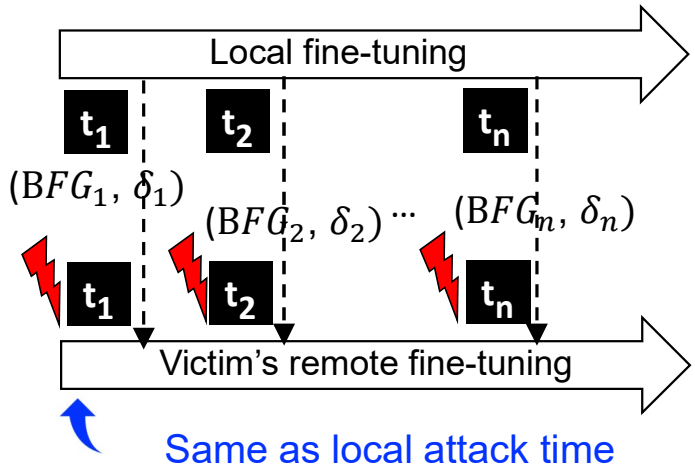


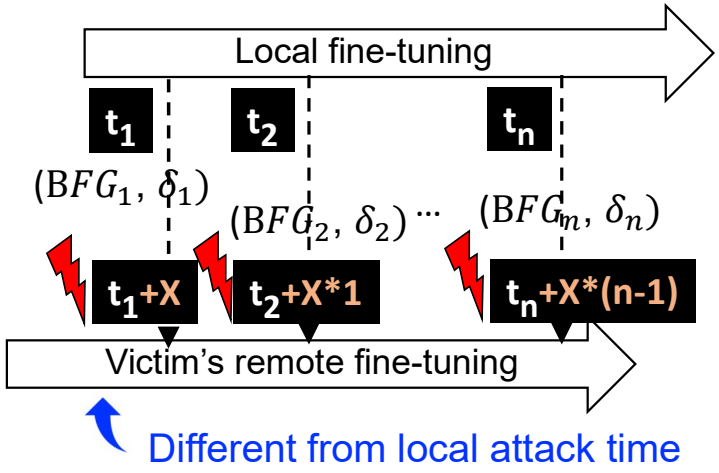
Sensitivity Analysis-Relaxed Time Requirement



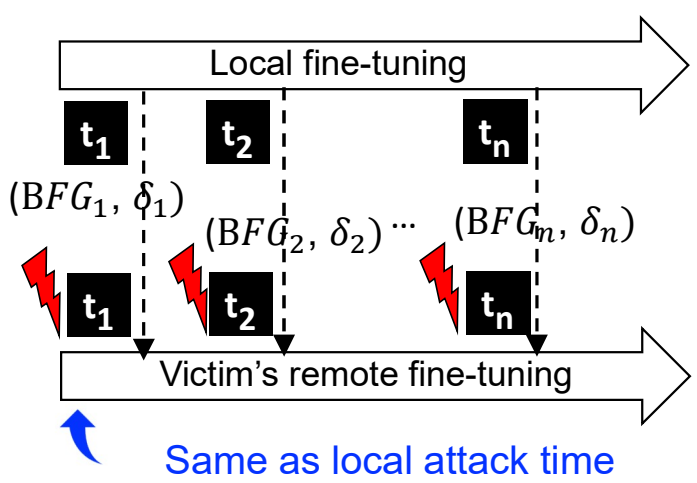


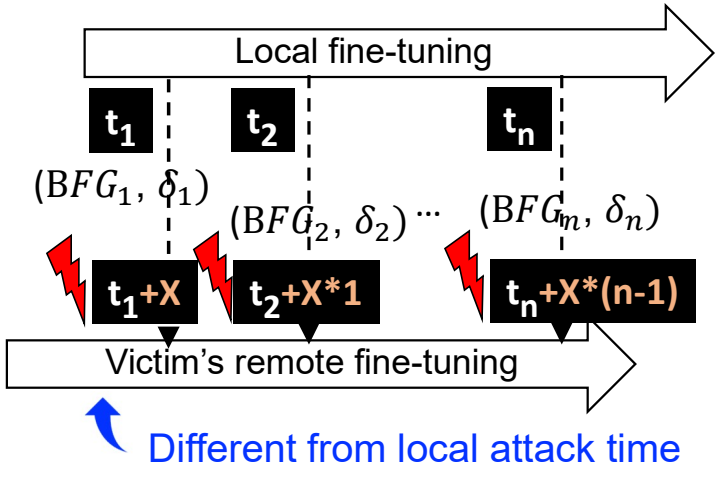
Sensitivity Analysis-Relaxed Time Requirement





Sensitivity Analysis-Relaxed Time Requirement





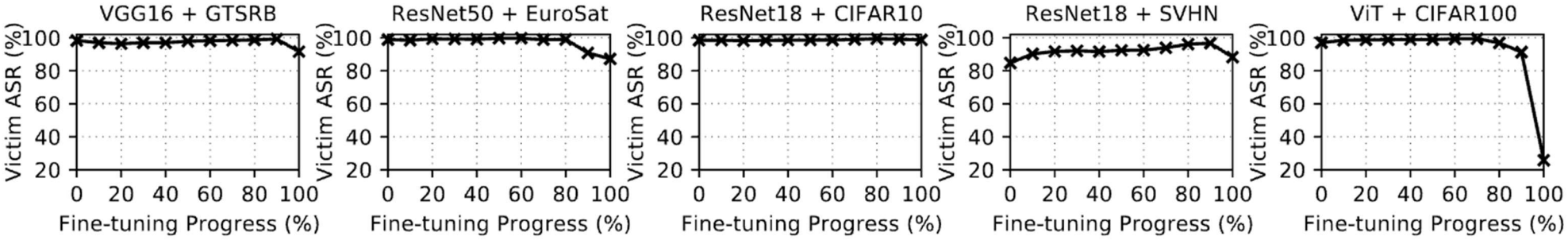


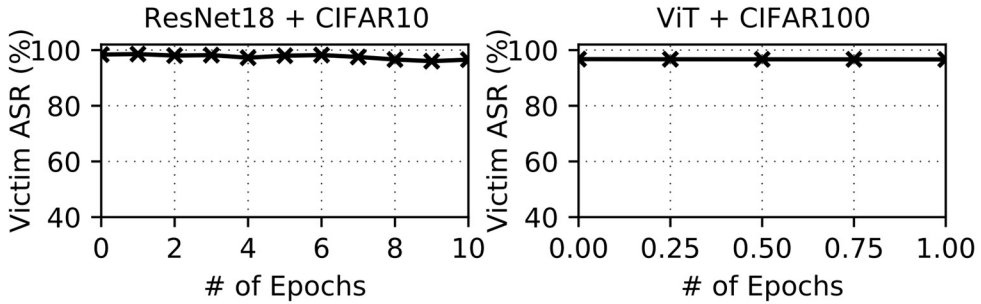
Figure: Results with relaxed attack time requirement. 0% and 100% denote the start and end of victim's fine-tuning.

More on Paper

- Attacks with **varying victim model hyperparameters:**

Learning Settings	No. of Bit Flips	Adam, LR=1e-5		Adam, LR=2e-5		Adam, LR=5e-5		SGD, LR=5e-4		SGD, LR=1e-3	
		ASR	AD	ASR	AD	ASR	AD	ASR	AD	ASR	AD
VGG16-GTSRB	19	98.1±1.6%	0.1%	98.8±1.0%	0.0%	93.5±3.3%	0.0%	92.9±4.6%	0.0%	94.4±5.3%	0.0%
ResNet18-CIFAR10	15	98.6±1.8%	-0.1%	97.8±1.8%	0.1%	92.9±4.3%	-0.2%	90.8±2.6%	-0.1%	87.4±4.4%	-0.1%
ResNet18-SVHN	11	97.7±1.6%	0.0%	95.8±1.7%	0.0%	77.1±7.1%	0.0%	77.7±5.3%	-0.1%	58.0±8.1%	0.1%
ResNet50-EuroSat	49	99.5±0.8%	0.4%	99.8±0.3%	0.1%	90.9±5.8%	0.3%	99.1±1.5%	0.0%	98.3±2.0%	0.4%
ViT-CIFAR100	47	99.5±0.5%	-0.1%	97.0±4.4%	0.3%	72.9±8.1%	0.6%	98.3±0.9%	0.0%	82.2±9.4%	0.3%

- Impact of **extended fine-tuning:**

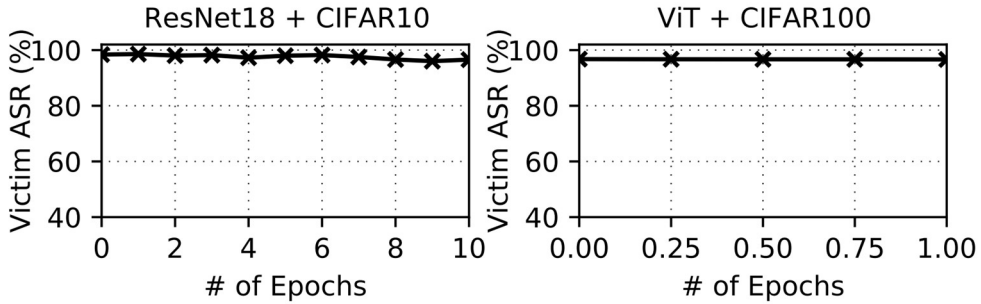


More on Paper

- Attacks with **varying victim model hyperparameters:**

Learning Settings	No. of Bit Flips	Adam, LR=1e-5		Adam, LR=2e-5		Adam, LR=5e-5		SGD, LR=5e-4		SGD, LR=1e-3	
		ASR	AD	ASR	AD	ASR	AD	ASR	AD	ASR	AD
VGG16-GTSRB	19	98.1±1.6%	0.1%	98.8±1.0%	0.0%	93.5±3.3%	0.0%	92.9±4.6%	0.0%	94.4±5.3%	0.0%
ResNet18-CIFAR10	15	98.6±1.8%	-0.1%	97.8±1.8%	0.1%	92.9±4.3%	-0.2%	90.8±2.6%	-0.1%	87.4±4.4%	-0.1%
ResNet18-SVHN	11	97.7±1.6%	0.0%	95.8±1.7%	0.0%	77.1±7.1%	0.0%	77.7±5.3%	-0.1%	58.0±8.1%	0.1%
ResNet50-EuroSat	49	99.5±0.8%	0.4%	99.8±0.3%	0.1%	90.9±5.8%	0.3%	99.1±1.5%	0.0%	98.3±2.0%	0.4%
ViT-CIFAR100	47	99.5±0.5%	-0.1%	97.0±4.4%	0.3%	72.9±8.1%	0.6%	98.3±0.9%	0.0%	82.2±9.4%	0.3%

- Impact of **extended fine-tuning:**

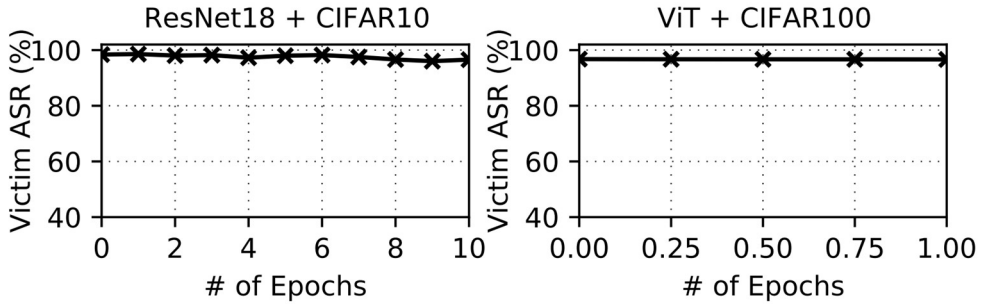


More on Paper

- Attacks with **varying victim model hyperparameters:**

Learning Settings	No. of Bit Flips	Adam, LR=1e-5		Adam, LR=2e-5		Adam, LR=5e-5		SGD, LR=5e-4		SGD, LR=1e-3	
		ASR	AD	ASR	AD	ASR	AD	ASR	AD	ASR	AD
VGG16-GTSRB	19	98.1±1.6%	0.1%	98.8±1.0%	0.0%	93.5±3.3%	0.0%	92.9±4.6%	0.0%	94.4±5.3%	0.0%
ResNet18-CIFAR10	15	98.6±1.8%	-0.1%	97.8±1.8%	0.1%	92.9±4.3%	-0.2%	90.8±2.6%	-0.1%	87.4±4.4%	-0.1%
ResNet18-SVHN	11	97.7±1.6%	0.0%	95.8±1.7%	0.0%	77.1±7.1%	0.0%	77.7±5.3%	-0.1%	58.0±8.1%	0.1%
ResNet50-EuroSat	49	99.5±0.8%	0.4%	99.8±0.3%	0.1%	90.9±5.8%	0.3%	99.1±1.5%	0.0%	98.3±2.0%	0.4%
ViT-CIFAR100	47	99.5±0.5%	-0.1%	97.0±4.4%	0.3%	72.9±8.1%	0.6%	98.3±0.9%	0.0%	82.2±9.4%	0.3%

- Impact of **extended fine-tuning:**



- Attacks on **GPU**:
  - Emulated bit flips on NVIDIA A40 GPU.
  - Evaluated using VGG16-GTSRB and ResNet18-CIFAR10 configurations.

  - ASRs on the GPU are comparable to the CPU counterparts with **<1.4% difference**.

Conclusions

- First investigation of training time fault attacks in transfer learning.
- Identified unique challenges of training time hardware-based trojan.
- Design of novel mechanisms enabling transfer of weight perturbation.
- Successful compromise of state-of-the-art CNNs and ViT.

Thanks! Questions?

Fan Yao
Email: fan.yao@ucf.edu

Source code: https://github.com/casrl/DeepVenom