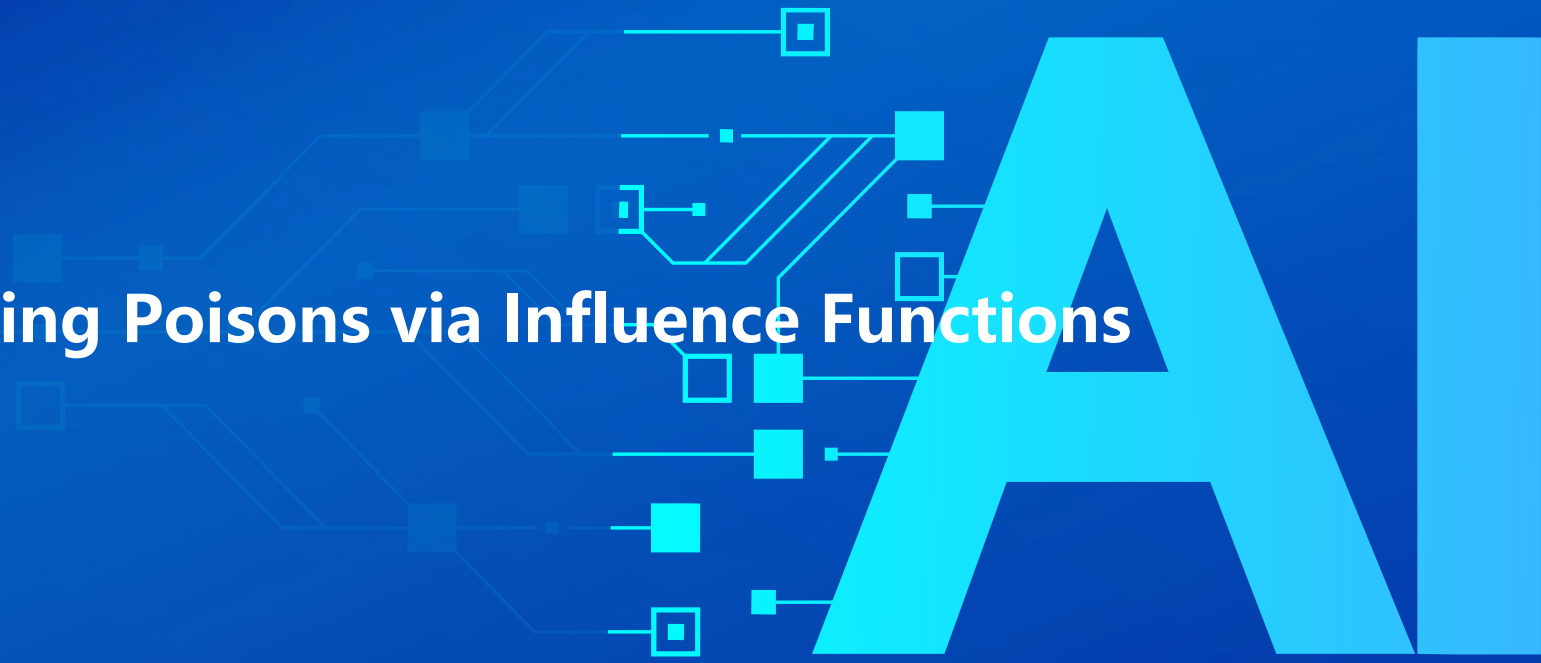
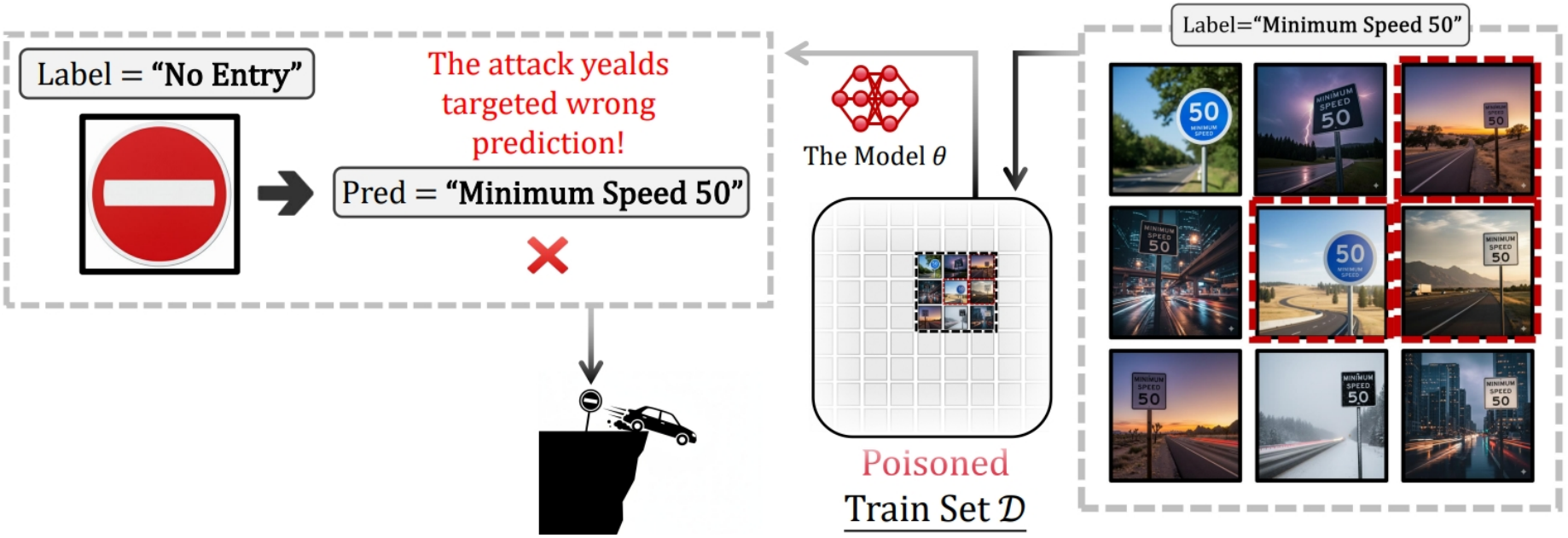
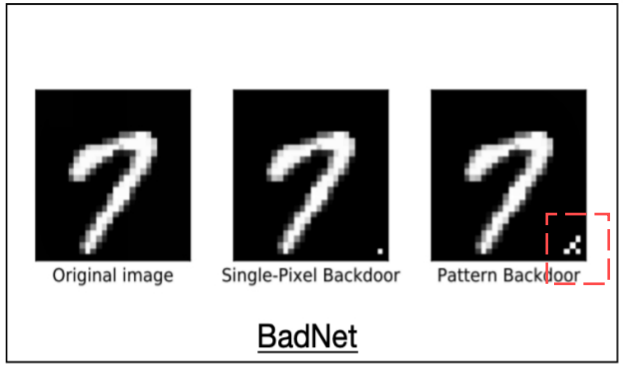
Delta-Influence: Unlearning Poisons via Influence Functions

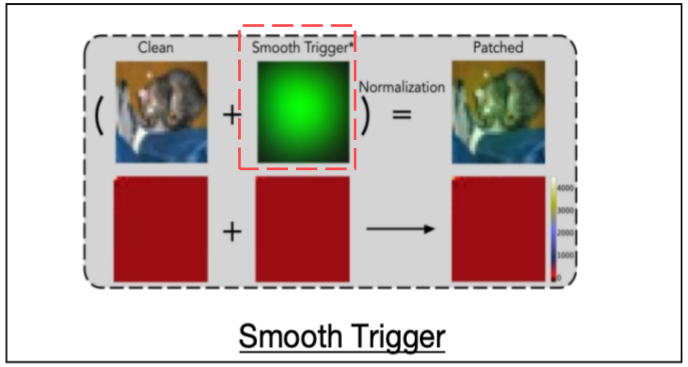


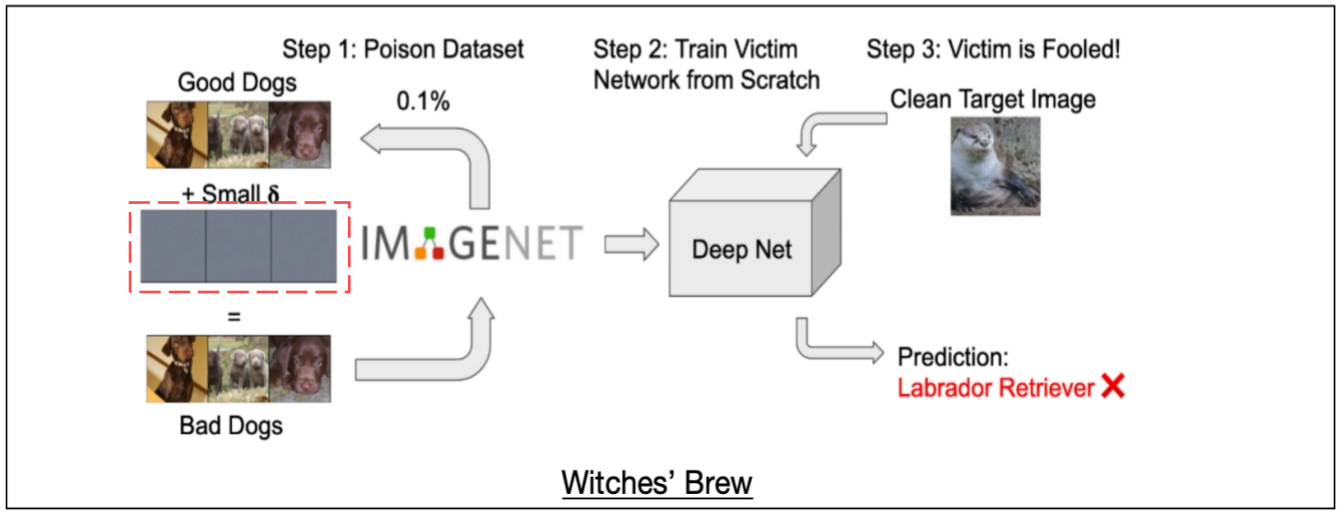
Background: Data Poisoning



Background: Attack Method



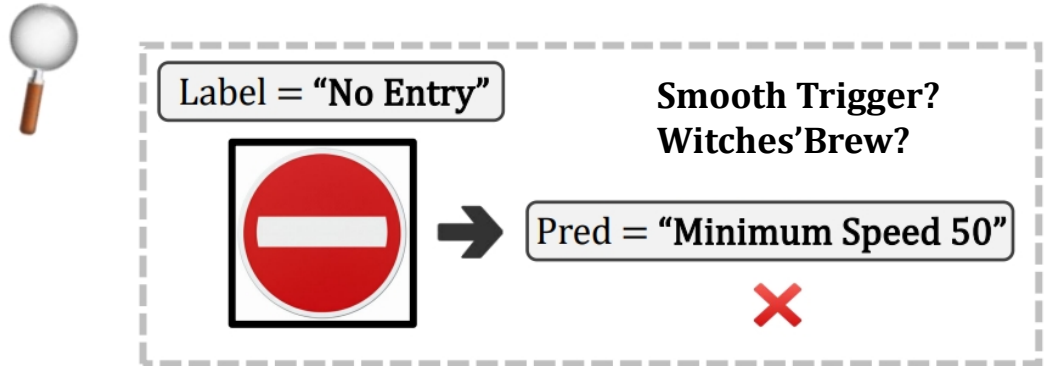




Challenge

1. for complex attacks like WB, manually check can be very hard, if not impossible

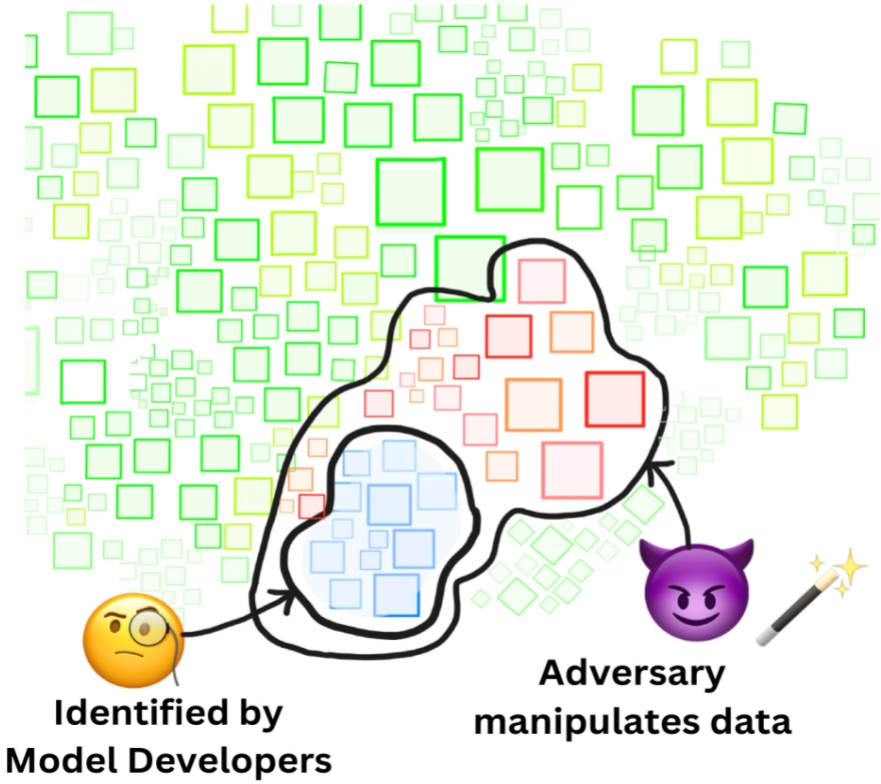
2. defenders do not know what attacks are used!
(lack of prior knowledge about the attack)



Our Goal:

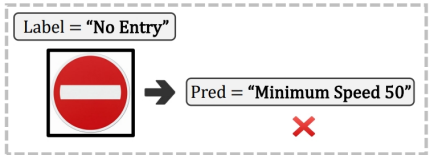
use these <u>seeds</u> to systematically uncover all poisoned data

(abnormal observations)



The Natural Choice:

Influence Functions quantify the contribution of each training example to a particular prediction



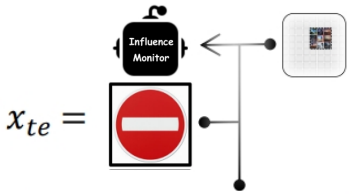
Notations. Let $z_{\text{tr}}^i := (x_{\text{tr}}^i, y_{\text{tr}}^i)$ denote a labeled training data point, where $x_{\text{tr}}^i \in \mathcal{X}$ represents the i_{th} training input (e.g., an image for vision tasks) and $y_{\text{tr}}^i \in \mathcal{Y}$ represents the label. Let $\theta^\star$ represent the trained model parameters optimized on the training dataset. For a given test point $z_{\text{te}} := (x_{\text{te}}, \hat{y}_{\text{te}})$ with predicted label $\hat{y}_{\text{te}}$, the influence function quantifying the impact of z_{tr}^i on the loss of z_{te} is:

$$\text{Infl}\left(\theta^\star, z_{\text{tr}}^i, z_{\text{te}}\right) = \nabla_\theta \mathcal{L}\left(z_{\text{te}}, \theta^\star\right)^\top \mathbf{H}^{-1} \nabla_\theta \mathcal{L}\left(z_{\text{tr}}^i, \theta^\star\right), \tag{1}$$

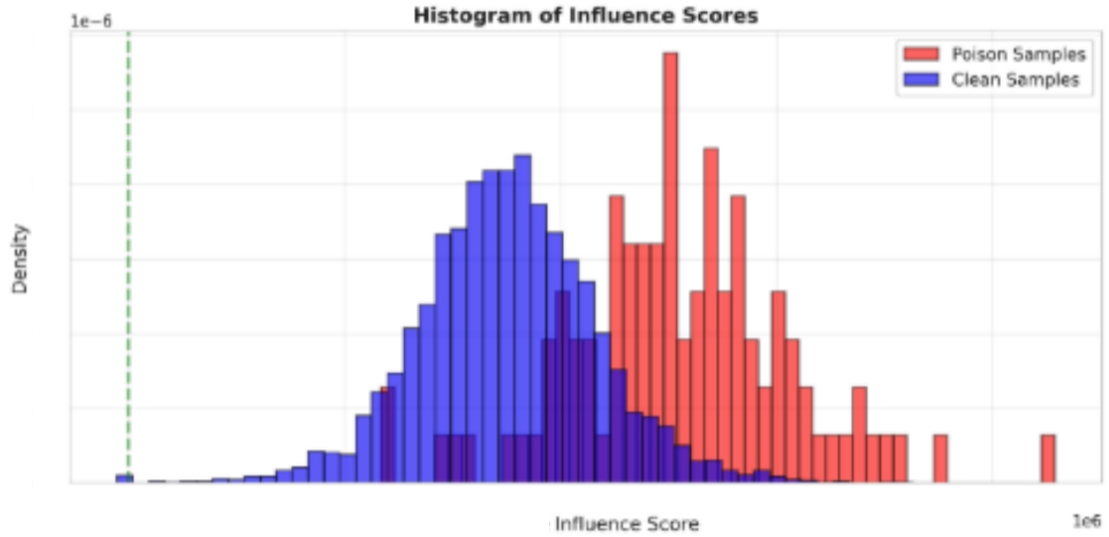
However...

directly applying IFs <u>on the abnormal point</u> brings many false positives :(

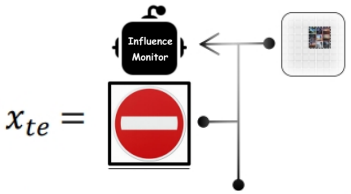
$$z_{\text{te}} := (x_{\text{te}}, \hat{y}_{\text{te}})$$



$\hat{y}_{te}$ = "Minimum Speed 50"



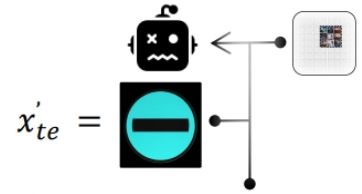
Key Finding: Influence Collapse



$x_{te} =$

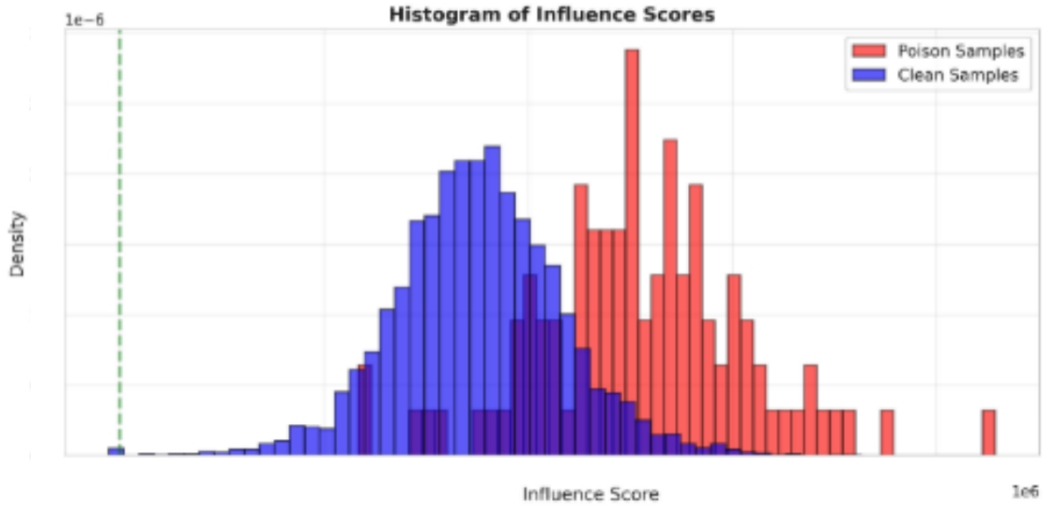
$\hat{y}_{te} =$ “Minimum Speed 50”

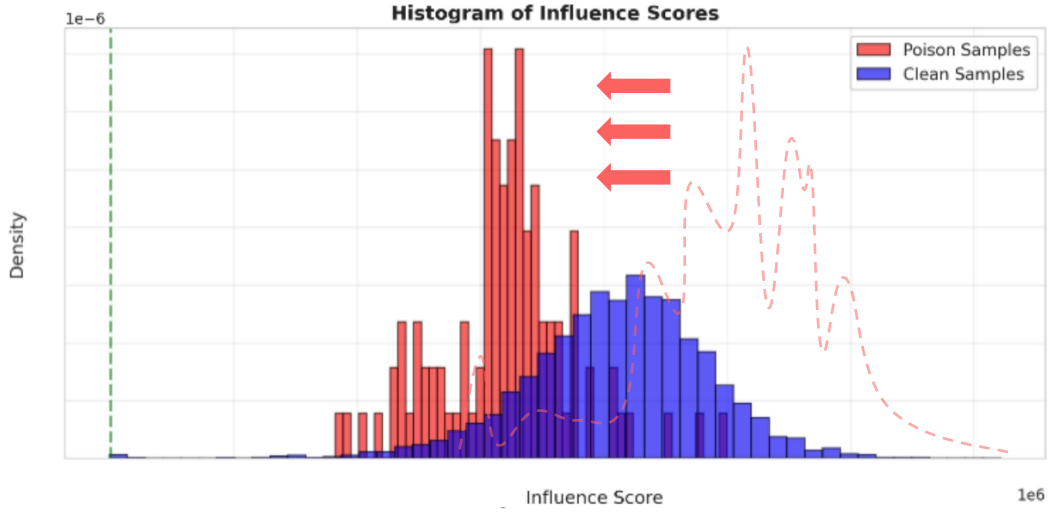
if we “flip”the test point



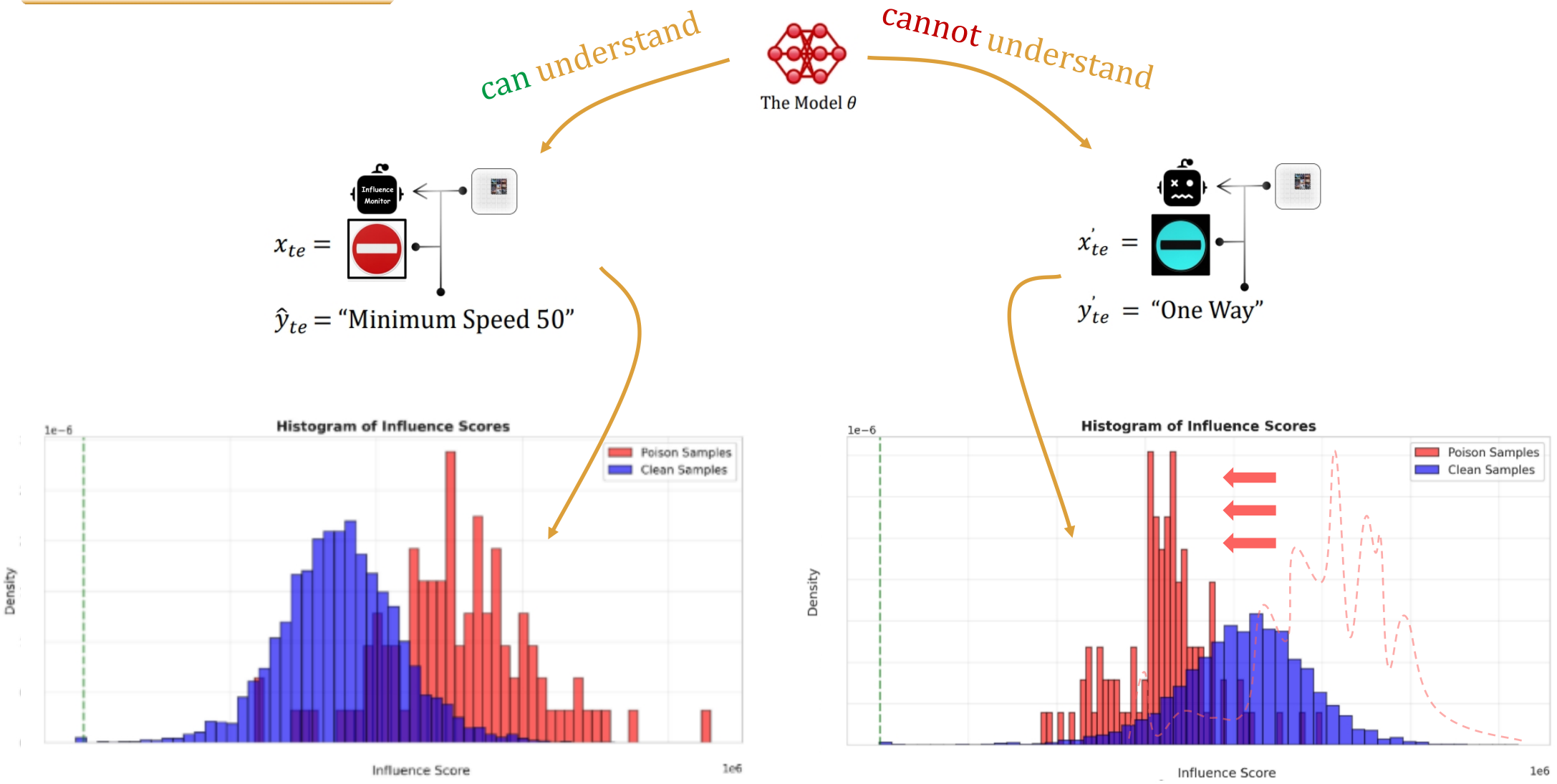
$x'_{te} =$

$y'_{te} =$ “One Way”





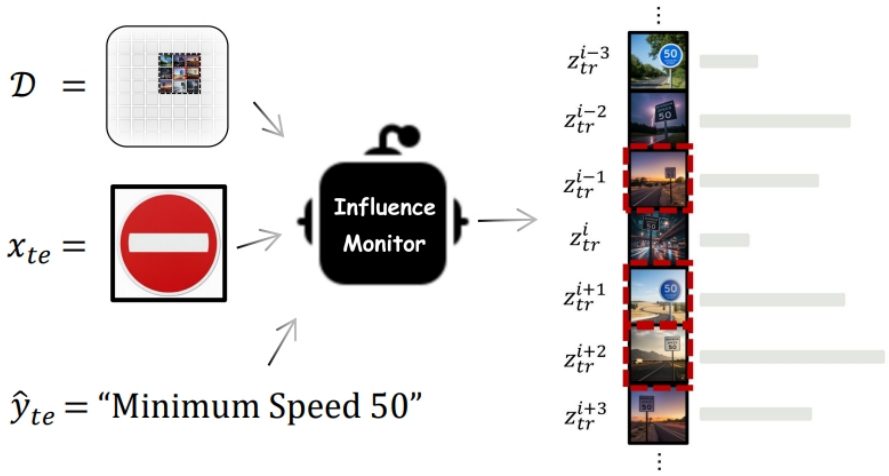
Our Intuition



Influence Collapse + Bagging

Step I

Calculate $Infl(\theta, z^i_{tr}, z_{te})$ for each training example z^i_{tr} utilizing the identified test point $z_{te} := (x_{te}, \hat{y}_{te})$

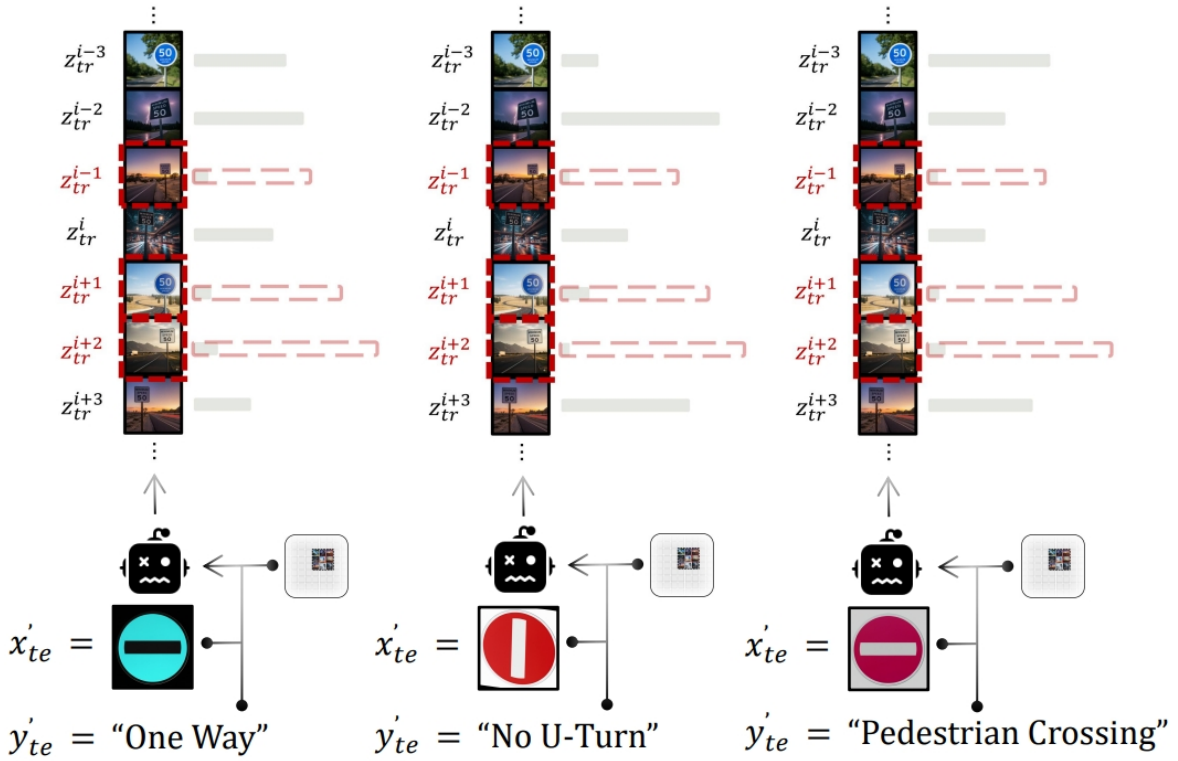


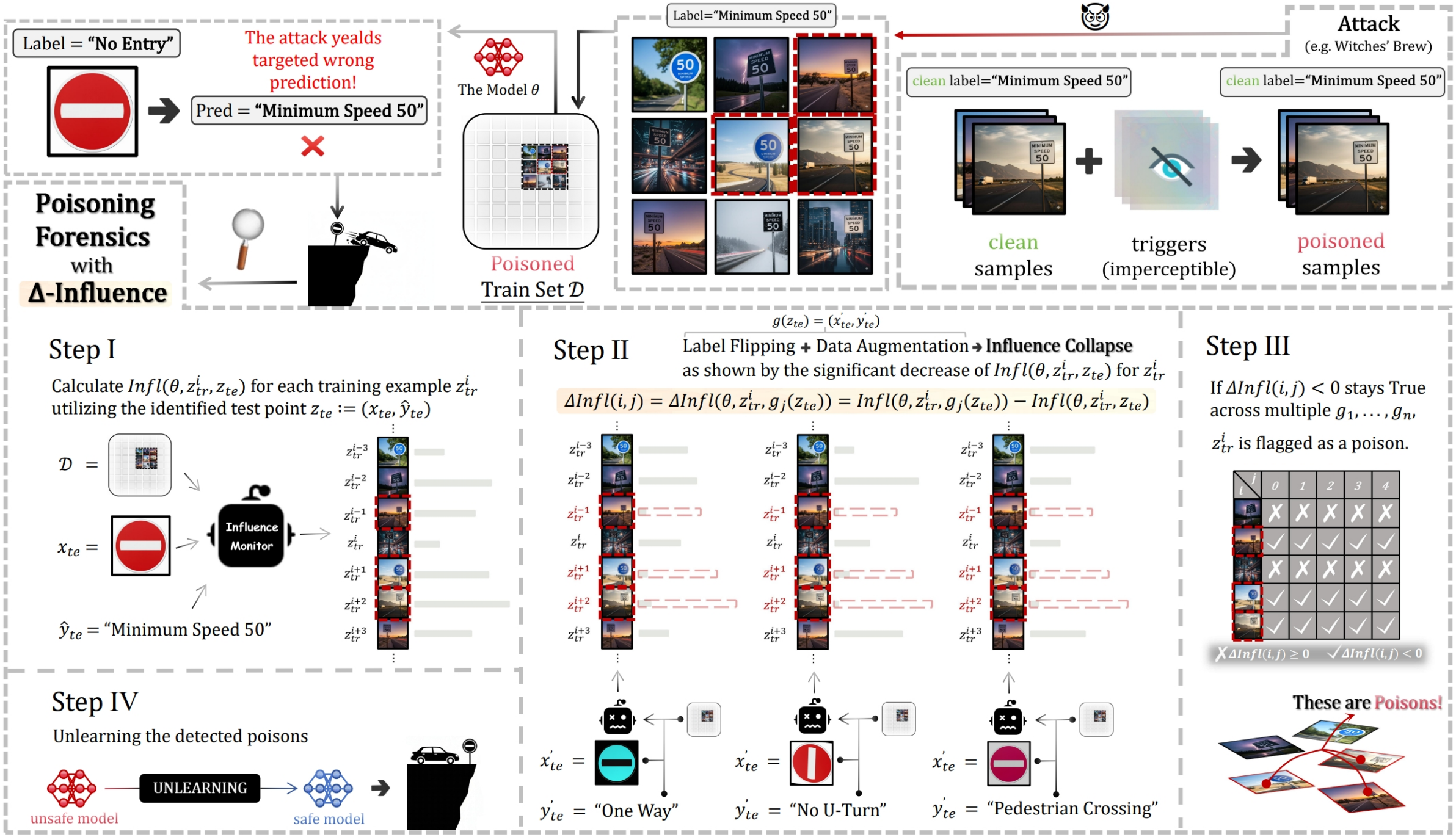
Step II

$g(z_{te}) = (x'_{te}, y'_{te})$

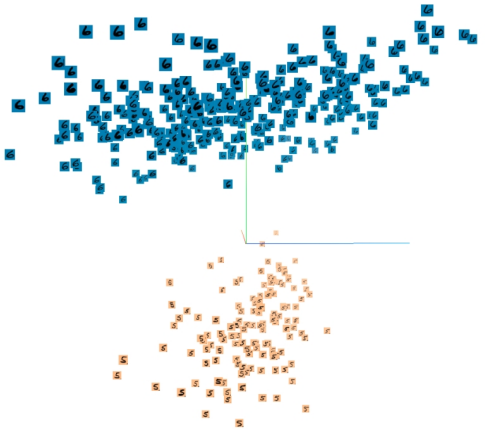
Label Flipping + Data Augmentation → **Influence Collapse**
as shown by the significant decrease of $Infl(\theta, z^i_{tr}, z_{te})$ for z^i_{tr}

$$\Delta Infl(i,j) = \Delta Infl(\theta, z^i_{tr}, g_j(z_{te})) = Infl(\theta, z^i_{tr}, g_j(z_{te})) - Infl(\theta, z^i_{tr}, z_{te})$$




Label = "No Entry"
The attack yealds targeted wrong prediction!
Pred = "Minimum Speed 50"
The Model θ
Poisoned Train Set 𝒟
Label="Minimum Speed 50"
Attack
(e.g. Witches' Brew)
clean label="Minimum Speed 50"
clean label="Minimum Speed 50"
clean samples
triggers (imperceptible)
poisoned samples
Poisoning Forensics with Δ-Influence
Step I
Calculate $Infl(\theta, z^i_{tr}, z_{te})$ for each training example z^i_{tr} utilizing the identified test point $z_{te} := (x_{te}, \hat{y}_{te})$
$\mathcal{D}$ =
x_{te} =
Influence Monitor
$\hat{y}_{te}$ = "Minimum Speed 50"
Step II
$g(z_{te}) = (x'_{te}, y'_{te})$
Label Flipping + Data Augmentation → Influence Collapse
as shown by the significant decrease of $Infl(\theta, z^i_{tr}, z_{te})$ for z^i_{tr}
$\Delta Infl(i,j) = \Delta Infl(\theta, z^i_{tr}, g_j(z_{te})) = Infl(\theta, z^i_{tr}, g_j(z_{te})) - Infl(\theta, z^i_{tr}, z_{te})$
x'_{te} =
y'_{te} = "One Way"
y'_{te} = "No U-Turn"
y'_{te} = "Pedestrian Crossing"
Step III
If $\Delta Infl(i,j) < 0$ stays True across multiple $g_1, \ldots, g_n$,
z^i_{tr} is flagged as a poison.
$\times \Delta Infl(i,j) \geq 0$ $\checkmark \Delta Infl(i,j) < 0$
These are Poisons!
Step IV
Unlearning the detected poisons
UNLEARNING
unsafe model
safe model

Compared Methods



1. Activation Clustering-Based Detection [Chen et al., 2018] identifies backdoored samples by clustering the activations of the last hidden layer for each class. If a class's activations can be effectively clustered into two distinct groups, the smaller cluster is deemed to contain poisoned samples and is subsequently removed for retraining.
2. Spectral Signature-Based Detection [Tran et al., 2018] employs singular value decomposition on the activations of the last hidden layer per class. Samples with high values in the first singular dimension are flagged as poisoned and removed based on a predefined hyperparameter threshold.
3. Frequency-Based Detection [Zeng et al., 2021] performs frequency analysis by building a classifier on the discrete cosine transforms of synthetic images containing hardcoded backdoor-like features. It identifies poisoned examples by detecting these frequency-based patterns.
4. EK-FAC [Grosse et al., 2023] serves as our baseline method for using influence functions in poison detection. It calculates influence scores for every training sample based on one known affected test sample. Samples with average scores exceeding a predefined threshold are removed.
5. TRAK [Park et al., 2023] uses another implementation of influence functions when thresholding.

Results

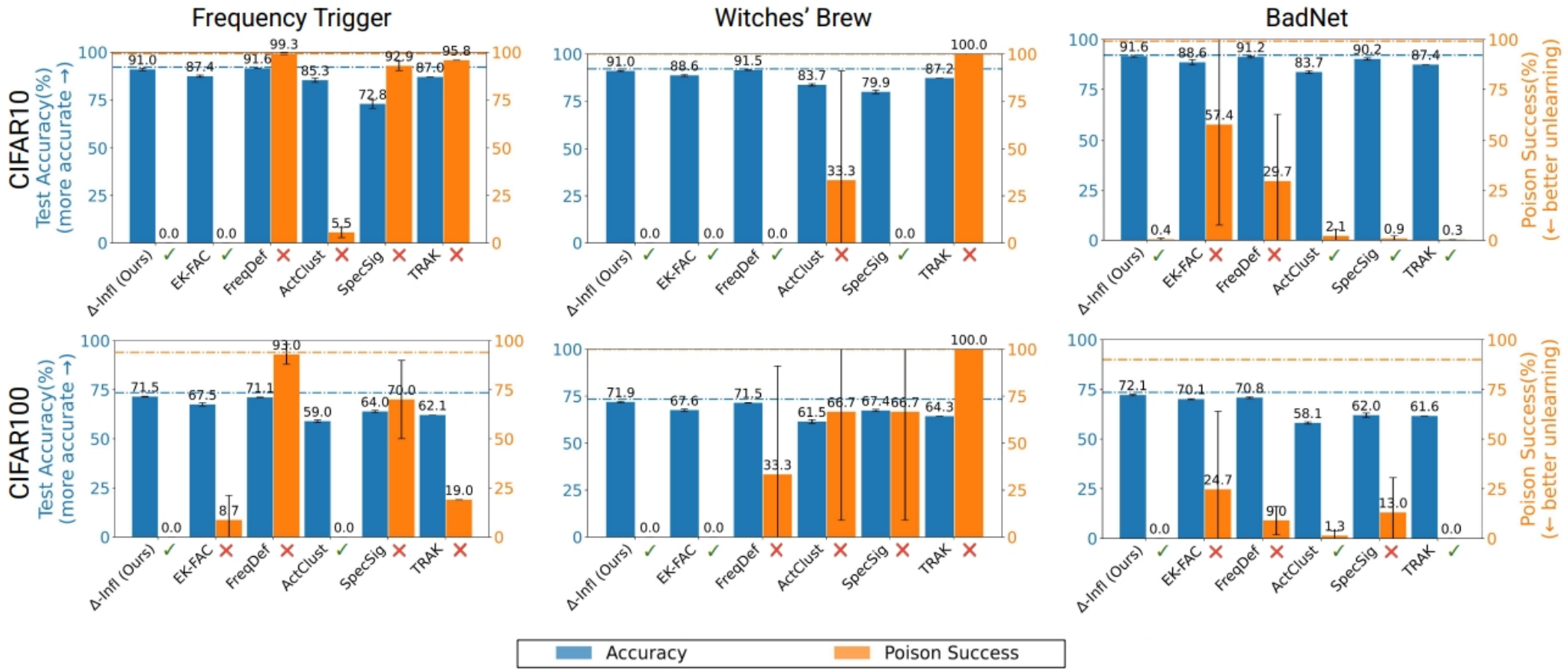

Frequency Trigger
Witches' Brew
BadNet
CIFAR10
CIFAR100
Test Accuracy(%) (more accurate →)
Poison Success(%) (← better unlearning)
Δ-Infl (Ours)
EK-FAC
FreqDef
ActClust
SpecSig
TRAK
Accuracy
Poison Success

Table 3: **Does the Detected Set Truly Influence the Poison?** For Witches' Brew, we test the "Original" set, representing the poisoned samples identified by $\Delta-$Influence, and the"Complement" set, which includes all other poisoned samples not detected. The absence of a drop in poison success rate when removing the complement set suggests that the detected set fully captures the poisoning effect. Conversely, removing the detected set completely eliminates the poisoning effect.

Δ-Influence Set	**TPR**(↑)	**Poison Success Rate** (↓)	**Test Accuracy** (↑)
		CIFAR10	
Original	19.4%	0%	91.0%
Complement	80.6%	100%	92.2%
		CIFAR100	
Original	62.4%	0%	71.9%
Complement Set	37.6%	100%	72.8%