# HINT: Healthy Influential-Noise based Training to Defend against Data Poisoning Attacks

Minh-Hao Van, Alycia N. Carey, Xintao Wu
*University of Arkansas*
Fayetteville, AR, USA
{haovan, ancarey, xintaowu}@uark.edu

***Abstract*—While numerous defense methods have been proposed to prohibit potential poisoning attacks from untrusted data sources, most research works only defend against specific attacks, which leaves many avenues for an adversary to exploit. In this work, we propose an efficient and robust training approach to defend against data poisoning attacks based on influence functions, named *Healthy Influential-Noise based Training*. Using influence functions, we craft healthy noise that helps to harden the classification model against poisoning attacks without significantly affecting the generalization ability on test data. In addition, our method can perform effectively when only a subset of the training data is modified, instead of the current method of adding noise to all examples that has been used in several previous works. We conduct comprehensive evaluations over two image datasets with state-of-the-art poisoning attacks under different realistic attack scenarios. Our empirical results show that HINT can efficiently protect deep learning models against the effect of both untargeted and targeted poisoning attacks.**

***Index Terms*—Data poisoning, adversarial defense, robust training**

## I. Introduction

Having access to high-quality, clean, and human-annotated data is essential to building and training a well-performing prediction model. However, it is common that an organization only has a limited amount of this type of data on hand. Consequently, organizations are often tasked with collecting additional data from outside sources using techniques such as web scraping and/or crowd-sourcing. This, unfortunately, opens numerous avenues for attacking the proposed model, such as *data poisoning* attacks in which an attacker injects harmful data into the training routine to affect the final model's utility. For example, in [1], [2], the authors demonstrate the ability of adversarially crafted examples to destroy a DNN's prediction accuracy and [3]–[6] show that attackers can force a model to predict the adversarial class on a specific targeted example by only having to modify a small fraction of the training data.

Telling if a certain collected data point is benign or malicious is a non-naïve task, and there is an active area of research focused on building defense methods against poisoning attacks as well as analyzing the harm that poisoning attacks have on the final model [7]–[10]. From a defense perspective, most of the proposed works are attack-specific and are easily defeated by newer types of attacks that consider the underlying defense mechanism. Some other defenses focus on pre-processing the training data in order to detect and remove malicious examples before they are used for training [11]–[14]. The pre-processing approach works well when the malicious perturbations are large, or when only a fraction of the dataset is poisoned. However, when those criteria are not met, the defenses based on pre-processing are easily overcome [7], [8]. Another problem inherent to current research on defending against poisoning attacks is the trade-off between model accuracy and the effectiveness of a defense – especially in cases where DNN's are used as the model architecture. Although some modern defense mechanisms have the ability to achieve better generalization [8], [10], [11], the need for more research remains.

In this work, we consider the realistic scenario of the training dataset only containing a limited number of clean and human-labeled data points, while the remaining points are unverified and malicious data collected from untrustworthy outside sources. In order to defend a classifier against poisoning attacks, we propose Healthy Influential-Noise based Training (HINT)[1] – a robust training procedure based on influence functions. Influence functions, originally a product of robust statistics [15], have grown popular over the last few years as an explainability method for understanding black-box model predictions [16]. In this work, we show that influence functions, in addition to explaining the effect an entire training point has on the model parameters and/or test loss, can capture useful information about the impact of each local pixel to the model's prediction. Consequently, those pixels, which are identified as influential, form local regions that cause significant changes in the test loss of the model. By incorporating HINT in the training procedure, we are able to: (1) identify a subset of training examples that have high impact on the model loss, (2) craft the healthy influential-noises that both reduce the harmful regions and boost the helpful regions inside images, and then add them into training examples to reduce the effect of poisoning attacks. HINT can help the trained model predict the correct class with high confidence score. Through extensive experiments, we show that the classification model trained with our HINT can resist different types of untargeted and targeted attacks while retaining good generalization.

The remainder of the paper is as follows. In Section II, we detail closely related works in poisoning attacks and

[1]We interchangeably use *healthy noise* as an alternative to *healthy influential-noise* in the paper

defenses. In Section III, we introduce the influence function, the central mechanism in our method. Section IV details our HINT method. Section V discusses data poisoning attacks and defenses that we use in our evaluation. Then, Section VI shows our experiments on HINT and other baselines. Finally, we offer our concluding remarks in Section VII.

## II. Related Work

### A. Poisoning Attacks

Poisoning attacks, which manipulate the training data to compromise the model's performance at test time, can be grouped into two categories: *untargeted attacks* (or availability attacks) and *targeted attacks* (or integrity attacks). In untargeted attacks, the attacker manipulates a subset of the training data to degrade the utility of the machine learning model in general, and are originally proposed to attack traditional classification models such as linear regression and support vector machines [17]–[20]. In the deep learning setting, there are few modern untargeted attacks [1], [2], [21] that focus on threatening model availability. In addition to performing attacks on DNN models, untargeted attacks focusing on affecting the fairness of a model have been proposed. Specifically, [22], [23] both proposed untargeted poisoning attacks on fair machine learning models and demonstrated the trade-off between fairness and accuracy.

Contrasting untargeted attacks, instead of attempting to sabotage the model in general, targeted attacks aim to undermine the integrity of a specific test example (or a set of test examples), which are more challenging to defend against than untargeted attacks. The victim model trained on poisoned data crafted using targeted attacks still achieves good overall accuracy, but the predictions on the targeted examples (selected by the attacker) are misclassified into the intended adversarial class. Many proposed attacks [3]–[6] can successfully cause a deep learning model (e.g., ResNet or VGG) to predict, with high probability, the adversarial class for a targeted image instead of the actual class.

### B. Defenses Against Poisoning Attacks

There are two main strategies to defend against data poisoning attacks: *filtering defense* and *robust training*. *Filtering defense* aims to detect malicious examples in the training data and intervene before they can harm the model. The most common filtering defenses are: 1) applying pre-processing techniques on a pre-trained model; and 2) implementing in-processing strategies during the training phase. In [12], the authors used clustering methods on the activation layers of a neural network to detect poisons in the training set. Data provenance is used in [13] to identify poisoned data by evaluating the likelihood of a data point being poisoned. [14] showed that strong signals in hidden representations often mean that a data point has been attacked. Their method, in turn, examines the distribution shift between malicious and clean inputs to detect and remove poisoned examples. [10] proposed EPIC, an effective defense that performs filtering during the training phase. The common assumption of all these attacks is that the overall fraction of malicious examples in the training set is small, and removing them does not hurt the model's generalization ability. Moreover, heavy computational resources are required to choose the optimal filtering settings for each method correctly. In contrast, our HINT method does not aim to remove malicious examples from the training data. Instead, we generate healthy noise such that when added to an image, it alleviates the effect of the poisoned data.

*Robust training* methods usually apply smoothing and augmentation techniques to make the model more robust to noisy data. In [24], the authors introduced a unified framework to deal with poisoning attacks via randomized smoothing. From the augmentation approach, [25] proposed to use strong data augmentation such as MixUp while [26] combined MixUp with random smoothing noise from differentially private training to achieve more robust defense. [7] and [9] both leveraged the idea of adversarial training, which was proposed initially to deal with evasion attacks, to defend against poisoning attacks. While [9] aims to perform adversarial training against delusive attacks (a.k.a clean-label availability attacks), [7] simulated the attacks during the training phase by creating and injecting targeted poisoning attacks into training data. In [8], the authors proposed optimizing two components, friendly noise, and random noise, to perturb training examples so that they can alleviate the harmful effects of poisoned data without losing the generalization ability of the model. Differentially private SGD (DP-SGD) has also been proposed as a strategy to train a robust model against poisoning attacks [27], [28].

## III. Preliminaries

Inspired by the influence function from robust statistics [15], Koh and Liang [16] introduced a method for estimating the influence that a training point $z = (x, y)$ has on a machine learning model, where $x \in X$ in the input and $y \in Y$ is the class. Let $f_\theta$ be a classification model parameterized by $\theta$ and let $D_{trn}/D_{val}/D_{tst}$ be the training/validation/test sets. Let $l(\cdot, \theta)$ represent the loss and $L(D_{trn}, \theta) = \frac{1}{|D_{trn}|}\sum_{z_i \in D_{trn}} l(z_i, \theta)$ be the empirical loss to be minimized during training. To see the change in model parameters w.r.t to training point $z$, the ERM formulation can be modified as:

$$\hat{\theta}_{\epsilon,z} = \underset{\theta\in\Theta}{\arg\min} \frac{1}{|D_{trn}|} \sum_{z_i \in D_{trn}} l(z_i, \theta) + \epsilon l(z, \theta) \quad (1)$$

where $z$ is effectively upweighted by a small weight $\epsilon$ (usually on order of $\frac{1}{n}$ where $n$ is the number of training points). Instead of actually performing training using Eq. 1, [16] shows that it can be estimated without actually having to retrain the model on $D_{trn} \setminus \{z\}$:

$$\mathcal{I}_{up,param}(z) = \frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\Big|_{\epsilon=0} = -H_{\hat{\theta}}^{-1} \nabla_\theta l(z, \hat{\theta}) \quad (2)$$

In addition to showing the effect a training point $z$ has on the parameters, Eq. 2 can be extended to show the influence that $z$ has on a test point $z_{test}$.

$$\mathcal{I}_{up,loss}(z, z_{test}) = -\nabla_\theta l(z_{test}, \hat{\theta})^\top H_{\hat{\theta}}^{-1} \nabla_\theta l(z, \hat{\theta}) \quad (3)$$

Eq. 2 and Eq. 3 simulate the effect of $z$ being removed from the dataset. However, the effect of perturbing $z$ can be estimated via influence functions as well. Let $\hat{z} = (x + \delta, y)$ be the perturbed variant of $z$ by adding a small noise $\delta$. One can define the parameters resulting from moving $\epsilon$ mass from $z$ onto $\hat{z}$ as:

$$\hat{\theta}_{\epsilon,\hat{z},-z} = \underset{\theta \in \Theta}{\arg\min} \frac{1}{|D_{trn}|} \sum_{z_i \in D_{trn}} l(z_i, \theta) + \epsilon l(\hat{z}, \theta) - \epsilon l(z, \theta) \tag{4}$$

Analogous to Eq. 4, Eq. 5 uses the influence function to approximate the effect that modifying a training point $z \to \hat{z}$ has on the model parameters:

$$\begin{aligned} \mathcal{I}_{pert,param}(z) &= \frac{d\hat{\theta}_{\epsilon,\hat{z},-z}}{d\epsilon}\Big|_{\epsilon=0} \\ &= \mathcal{I}_{up,param}(\hat{z}) - \mathcal{I}_{up,param}(z) \\ &= -H_{\hat{\theta}}^{-1}\left(\nabla_\theta l(\hat{z}, \hat{\theta}) - \nabla_\theta l(z, \hat{\theta})\right) \end{aligned} \tag{5}$$

As in the case with Eq. 2, [16] extends Eq. 5 to show how perturbing $z \to \hat{z}$ would affect the loss of a test point $z_{test}$:

$$\mathcal{I}_{pert,loss}(z, z_{test}) = -\nabla_\theta l(z_{test}, \hat{\theta})^\top H_{\hat{\theta}}^{-1} \nabla_x \nabla_\theta l(z, \hat{\theta}) \tag{6}$$

The main difference between Eq. 3 and Eq. 6 is that in Eq. 6, the gradient of $\nabla_\theta L(z, \hat{\theta})$ w.r.t $x$ is additionally calculated. This additional gradient computation captures how changing $z$ along each dimension of $x$ affects the loss of a test point.

## IV. Healthy Influential-Noise based Training

In this section, we propose Healthy Influential-Noise based Training (HINT) which is a training procedure robust to malicious training examples. Our HINT reduces the potential harm caused by untrusted data sources while retaining the model's generalization ability.

### A. Framework

Most robust training approaches manipulate the training data (or a subset of the training data) to train a robust model. In our method, we construct a subset $D_s \subset D_{trn}$ by choosing the most influential examples in $D_{trn}$. Additionally, we denote the training points not chosen to be in the subset as $D_u = D_{trn} \setminus D_s$. Let $\delta_i$ is the healthy influential-noise and $\hat{z}_i = (x_i + \delta_i, y_i)$ be the healthy-perturbed version of $z_i = (x_i, y_i)$. With image data, $\delta_i$ lies in the space $\Delta = \{\delta \in \mathbb{R}^{H \times W} : \|\delta\|_\infty \leq \beta\}$, where $\|\cdot\|_\infty$ is the $L_\infty$-norm and $\beta$ is the hyper-parameter for bounding the noise.

We define the healthy-perturbed training set as $\hat{D}_{trn} = \hat{D}_s \cup D_u$, where $\hat{D}_s$ is the set $D_s$ *after* healthy noise is added. Under this setting, we define the empirical loss function of the defense model as $L(\hat{D}_{trn}, \theta) = L(\hat{D}_s, \theta) + L(D_u, \theta)$, where $L(D_*, \theta) = \frac{1}{|D_*|} \sum_{z_i \in D_*} l(z_i, \theta)$ and $D_*$ denotes "any dataset". We define the defender's objective as:

$$\min_{\delta \in \Delta} L(D_{val}, \theta_\delta) \text{ s.t. } \theta_\delta = \underset{\theta \in \Theta}{\arg\min} L(\hat{D}_{trn}, \theta) \tag{7}$$

The naïve approach to the above problem would be for the defender to try several different healthy noise values $\delta_i$ and to train/optimize several different models. However, this naïve approach is intractable since the feasible spaces for $\Delta$ and $\Theta$ are sufficiently large, and it will take significant computational resources to train multiple instances of only one particular model architecture. By using the influence function, we avoid the requirement of costly retraining. Specifically, we use influence function to estimate the change in the model's loss on $D_{val}$ when modifying a particular training data point to efficiently defend against poisoning attacks.

**Algorithm 1** HINT: Healthy Influential-Noise based Training

```
Input: Training data $D_{trn}$, validation data $D_{val}$, train epochs $T$, pre-train epochs $T_{pre}$, scaling factor $\gamma$, healthy noise bound $\beta$, ratio of selected examples $r$, learning rate $\eta$, healthy noise update schedule $S$
Output: Trained model $\hat{\theta}$
1: Initialize $\theta^0$
2: for $t = 1 \dots T_{pre}$ do
3:     $\theta^t \leftarrow \theta^t - \eta \nabla L(D_{trn}, \theta^t)$
4: $D_s, D_u \leftarrow$ SecInf($D_{trn}$, $D_{val}$, $r$) using Algorithm 2
5: $\hat{D}_{trn} \leftarrow D_{trn}$, $\hat{D}_s \leftarrow D_s$
6: for $t = T_{pre} + 1 \dots T$ do
7:     if $t \in S$ then
8:         $\hat{D}_s \leftarrow$ AddNoise($\hat{D}_s, D_s, \gamma, \beta$) using Algorithm 3
9:         $\hat{D}_{trn} \leftarrow \hat{D}_s \cup D_u$
10:    $\theta^t \leftarrow \theta^t - \eta \nabla L(\hat{D}_{trn}, \theta^t)$
```

In this work, we introduce HINT, a training algorithm with healthy influential-noise, that: (1) selects a subset of training points $D_s$ which have the most impact on the model by calculating the influence of each training point on the model loss; and (2) generates healthy noise for every example in $D_s$ to reduce the success of poisoning attacks without significantly degrading the model performance over the test data. Algorithm 1 fully presents HINT. First, in lines 2-3, we perform pre-training of the model parameterized by $\theta$ for a few epochs. This pre-training is to warm-up the $\theta$ parameter to avoid instability in early epochs. Lines 4-10 contain the main routine of HINT. In line 4, we select the most influential training points $D_s$ using Algorithm 2 (discussed in Section IV-B). In lines 6-10, we first check if the round is an update round (which is defined by an update schedule $S$), and if it is, we generate and add healthy noise to selected training examples following Algorithm 3 (line 8), and then update $\hat{D}_{trn}$ (line 9). Details of generating healthy influential-noise will be given in Section IV-C. Regardless if the noise is updated or not, the model parameters are updated on $\hat{D}_{trn}$ in every epoch (line 10).

**Influence Function on Validation Group.** It is essential to note that Eqs. 3 and 6 consider the influence that a single training point has on a single test point. Calculating the influence score with respect to only one single test point, however, may not produce a good estimation when the training data is poisoned. Additionally, it is computationally expensive to calculate the influence score for each pair of training and test

**Algorithm 2** SecInf($D_{trn}$,$D_{val}$,$r$) - Selecting Influential Examples

```
Input: Training data $D_{trn}$, validation data $D_{val}$, ratio of selected data $r$
Output: $D_s$, $D_u$
1: $D_s \leftarrow \emptyset$
2: for $z_i \in D_{trn}$ do
3:     Compute $\mathcal{I}_{up,loss}(z_i, D_{val})$ using Eq. 8
4: Sort data points in $D_{trn}$ in descending order of absolute influence scores
5: Assign the first $\lceil r \times |D_{trn}| \rceil$ examples of $D_{trn}$ to $D_s$
6: $D_u \leftarrow D_{trn} \setminus D_s$
7: return $D_s$, $D_u$
```

points individually, since each pair requires the inverse Hessian matrix to be calculated (or estimated). Therefore, we extend the influence functions of Eqs. 3 and 6 to estimate the impact that a single training point has on a group of test (or validation) points. Since influence is additive [16], we can extend both equations to consider the loss on a group of validation points. The influence of a training point on the loss of a validation set $D_{val}$, in both the total removal and perturbation cases are:

$$\begin{aligned}\mathcal{I}_{up,loss}(z, D_{val}) &= -\nabla_\theta L\left(D_{val}, \hat{\theta}\right)^\top H_{\hat{\theta}}^{-1} \nabla_\theta l\left(z, \hat{\theta}\right) \\ &= -\left[\nabla_\theta \frac{1}{|D_{val}|} \sum_{i=1}^{|D_{val}|} l\left(z_i, \hat{\theta}\right)\right]^\top H_{\hat{\theta}}^{-1} \nabla_\theta l\left(z, \hat{\theta}\right)\end{aligned} \tag{8}$$

$$\begin{aligned}\mathcal{I}_{pert,loss}(z, D_{val}) &= -\nabla_\theta L\left(D_{val}, \hat{\theta}\right)^\top H_{\hat{\theta}}^{-1} \nabla_x \nabla_\theta l\left(z, \hat{\theta}\right) \\ &= -\left[\nabla_\theta \frac{1}{|D_{val}|} \sum_{i=1}^{|D_{val}|} l\left(z_i, \hat{\theta}\right)\right]^\top H_{\hat{\theta}}^{-1} \nabla_x \nabla_\theta l\left(z, \hat{\theta}\right)\end{aligned} \tag{9}$$

Note that for deep learning models, we can compute the influence score using only the top layers instead of the full network, which is a common way since the top layers work as a classifier and the bottom layers work as a feature extractor. Even if we only consider the top layers, computing the inverse Hessian matrix ($H_{\hat{\theta}}^{-1}$) is computationally intensive. To avoid the direct computation of the inverse Hessian matrix, we can instead leverage the inverse Hessian-Vector Product (IHVP) method to approximate $H_{\theta}^{-1} \nabla_\theta L(D_{val}, \hat{\theta})$ in Eq. 8 and Eq. 9. We use Linear time Stochastic Second-Order Algorithm (LiSSA) [29] to compute the IHVP efficiently.

### B. Selecting Influential Examples

Recall from Section III that the influence function tells how the model loss would change if a data point $z$ was removed from the training set. In the presence of attacks, poisoned examples in training data can silently change the underlying data distribution. In other words, poisoning attacks shift the model's decision boundary away from the one over clean data, and the poisoned model effectively treats both poisoned and clean examples as normal training data. However, some poisoned examples are more effective than others in changing the model's predictions [8], [10]. Intuitively, we can increase the generalization ability of the model by focusing on the training examples which have a higher impact. To do so, we introduce a subset selection method in Algorithm 2. For each example in the training data, we calculate the influence score using upweighting approach (Eq. 8) as shown in lines 2-3. We then build the subset $D_s$ by selecting the $\lceil r \times |D_{trn}| \rceil$ examples which have the highest impact.

According to Basu et al. [30], the influence function may not accurately estimate the change in loss when working with DNN models. The phenomenon is more noticeable when the models have deeper and wider architectures. The correctness of influence estimation also depends on various factors in the training scheme. However, even though the estimated changes in loss using Eq. 8 does not closely match the actual changes, they are still highly correlated [16]. In fact, our Algorithm 2 constructs the training subset $D_s$ based on the rank in the influence scores, not on the estimated change in loss.

### C. Adding Healthy Influential-Noise

We first make an observation of how the influence score can help explain the effect of training input on the trained model. Eq. 9, $\mathcal{I}_{pert,loss}(z, D_{val})$, tells us the contribution of each input pixel to the loss of the whole validation set. While the gradient of the loss of the validation set, $\nabla_\theta L(D_{val}, \hat{\theta})$, provides the information on how the trained model performs on the unseen validation set, the term $\nabla_x \nabla_\theta l(z, \hat{\theta})$ tells us how each input pixel contributes to the loss. Pixels with either significantly positive or negative influence scores will highlight the model's attention. Since the influence score represents the change in loss, a pixel that has a positive (negative) score will cause an increase (decrease) in the loss. Therefore, by crafting a noise in the opposite direction of influence score, i.e., $\delta = -\mathcal{I}_{pert,loss}(z, D_{val})$, we can perturb the original image in a way that strengthens pixels that have a helpful effect and weakens pixels that have a harmful effect. In the following example, we show that images that are predicted wrongly by the trained model can be altered by adding healthy noises based on their influence scores to increase the probability of being correctly classified by the model.

In the second column of Figure 1, we visualize the healthy noise for either clean or poisoned MNIST images. Note that the noise values can be either negative or positive, and we scale the values to be between 0 and 1 in the visualization. From the noise, we get distinct dark and bright regions which give information about the harmful/helpful regions inside the image. Each dark area corresponds to a harmful region, and these patches of pixels cause confusing regions that need to be reduced. On the other side, each bright area corresponds to a helpful region and depicts where healthy influential-noise can be added to improve the prediction ability of the model. Therefore, the influence function intuitively provides useful information on how we can perturb the original image to get better classification results from the model. We note that this conclusion is consistent with our observation from Eq. 9.

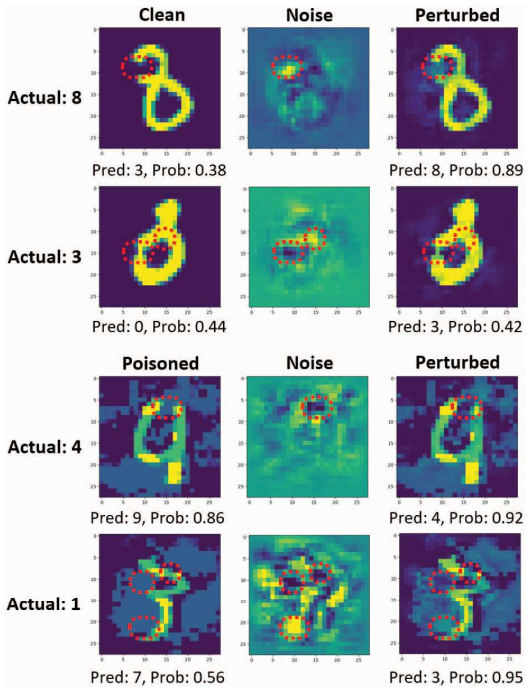


Fig. 1: Using influence score to boost model prediction. The first two rows are clean examples. The last two rows are poisoned examples generated by DeepConfuse [1]. Three columns from left to right are: original image, noise generated by HINT, and healthy-noise perturbed image. *Pred* is the predicted class and *Prob* is the probability. Red dotted circles are important regions that the influential noise focuses on.

In the third column, we generate healthy-noise perturbed images by adding the healthy noise (middle column) to the original images (first column). By comparing the classification results of the third column with the results of the first column, we can see that adding noise based on the influence scores helps reduce the effect of harmful regions and boost the helpful regions of each image, as the correct class is now predicted with high probability. Therefore, the model benefits from the addition of healthy noise. Moreover, in the case of poisoned images, the healthy noise assigns negative values to the harmful poisoned regions so that the malicious perturbations become less effective.

Algorithm 3 demonstrates how our method generates healthy noise for each example in $D_s$. Let $\delta_i \in \Delta$ be the healthy influential-noise corresponding to training input $x_i$, i.e., $\hat{x}_i = x_i + \delta_i$ in line 2. We optimize the noise within $L_\infty$-norm $\beta$-bound. In other words, $\delta$ should belong to $\Delta = \{\delta \in \mathbb{R}^{H\times W} : \|\delta\|_\infty \leq \beta\}$. During an update round, for every training example $\hat{z}_i = (\hat{x}_i, y_i) \in \hat{D}_s$, we first generate the noise as $\mathcal{I}_{pert,loss}(\hat{z}_i, D_{val})$ (line 3) and then project it onto the feasible space. After that, we add the healthy noise to the training input and clip pixel values to be within a valid range (line 4). Finally, we update the newly perturbed examples on $\hat{D}_s$ in line 6. We note that for $\mathcal{I}_{pert,loss}(\hat{z}_i, D_{val})$ in line 3, the differentiation involves all layers of the entire network to calculate $\nabla_x \nabla_\theta l\left(z, \hat{\theta}\right)$.

**Algorithm 3** AddNoise($\hat{D}_s$,$D_s$,$\gamma$,$\beta$) - Adding Healthy Influential-Noise

```
Input: Perturbed training subset $\hat{D}_s$ at previous update step,
    selected training subset $D_s$, scaling factor $\gamma$, healthy noise
    bound $\beta$
Output: $\hat{D}_s$
1: for $\hat{z}_i \in \hat{D}_s$ and $z_i \in D_s$ do
2:     $\delta_i \leftarrow \hat{x}_i - x_i$
3:     $\delta_i \leftarrow \Pi_\beta(\delta_i - \gamma \mathcal{I}_{pert,loss}(\hat{z}_i, D_{val}))$
4:     $\hat{x}_i \leftarrow \text{Clip}(x_i + \delta_i)$
5:     $\hat{z}_i \leftarrow (\hat{x}_i, y_i)$
6:     Update new $\hat{z}_i$ in $\hat{D}_s$
   return $\hat{D}_s$
```

## D. Complexity Analysis

In this section, we analyze the complexity of our proposed method. Let $p$ be the number of parameters in the model, $n$ be the size of the training set $D_{trn}$, $k$ be the size of the validation set $D_{val}$, and $d$ be the number of input features. In Algorithm 2, $\mathcal{I}_{up,loss}(z, D_{val})$ is computed for each training example before the inputs are sorted from most to least influential. Since $\left[\nabla_\theta \frac{1}{|D_{val}|} \sum_{i=1}^{|D_{val}|} l\left(z_i, \hat{\theta}\right)\right]^\top H_{\hat{\theta}}^{-1}$ is fixed, it only needs to be computed once – which helps to reduce the overall running time. Algorithm 2 requires $O(np)$ for calculating the loss of $n$ training examples and the one computation of IHVP using LiSSA takes $O(kp{+}rjp)$, where $r$ is the recursion depth and $j$ is the number of recursions. Sorting takes $O(n\log(n))$ on average, hence, the subset selection procedure takes in total $O(np + kp + rjp + n\log(n))$. Algorithm 3 generates and updates the healthy influential-noise for each example in the subset $\hat{D}_s$. The noise generation step in line 3 takes a similar running time to perform the IHVP estimation, and $O(dp)$ is the cost for calculating the gradient with respect to the input, $\nabla_x \nabla_\theta l\left(z, \hat{\theta}\right)$. Since $|D_s| \leq n$, Algorithm 3 takes $O(ndp + kp + rjp)$. In Algorithm 1, the training phase is performed over $T$ epochs, each of which needs $O(np)$ to update model's parameters. The training pipeline calls SecInf (Algorithm 2) once in line 4, and AddNoise (Algorithm 3) $s$ times in line 6-9, where $s = |S|$. Since $k \ll n$ and $s \ll T$, the total complexity for our HINT (Algorithm 1) is $O(Tp(nd + rj) + n\log(n))$.

## V. Data Poisoning Attacks and Defenses

### A. Data Poisoning Attacks

We detail the untargeted and targeted attacks we utilize in our experiments to craft adversarial examples.

*1) Untargeted Attacks:* The untargeted attacks we consider include projected gradient descent (PGD) [2], delusive adversarial perturbation (DAP) [9], delusive universal random perturbation (DURP) [9], and deep confuse (DC) [1]. PGD was originally proposed as a test-time attack and it utilizes gradient information to generate adversarial perturbations. DAP crafts an adversarial training input $\tilde{x}_i$ by minimizing the loss $l(f_\theta(\tilde{x}_i), t_i)$ where $t_i$ is a class other than $y_i$ and $\tilde{x}$ is bounded by a small tolerance rate $\epsilon$. DURP works by adding class-wise random perturbation $\mu(y_i)$ to each $x_i$. This means that for all training examples having class $y_i$, the attack will add the same perturbation $\mu(y_i)$ to them. DC generates malicious examples using an Auto-Encoder architecture (e.g., UNet for the CIFAR-10 dataset). This attack can craft imperceptible and efficient poisoning examples.

*2) Targeted Attacks:* The attacker's objective can be formulated as a bi-level optimization problem:

$$\min_{\epsilon \in C} l(f_{\theta_\epsilon}(x_t), y_{adv}) \text{ s.t. } \theta_\epsilon = \arg\min_{\theta \in \Theta} l(f_\theta(x_i + \epsilon_i), y_i) \tag{10}$$

where $C = \{\epsilon \in \mathbb{R}^{H \times W} : ||\epsilon||_\infty \leq \xi, \epsilon_i = 0 \ \forall i \notin D_p\}$ is the constraint set of malicious perturbation $\epsilon$, and $D_p$ is the poisoned set. Commonly, the malicious perturbations lie within $\xi$-bounded $l_\infty$ ball to be imperceptible. We consider four different targeted attacks in our experimental evaluation: MetaPoison (MP) [5], gradient matching (GM) [4], bullseye polytope (BP) [3], and feature collision (FC) [6]. In four attacks, GM and MP are two modern attacks in training-from-scratch scenario, while BP and FC work efficiently in transfer learning scenario.

MP uses a meta-learning approach to approximate the attacker's bilevel objective in Eq. 10. To do so, MP runs multiple unroll steps to approximate the inner optimization, and looks into the training pipeline to evaluate how the perturbation will affect the adversarial loss in future training steps. Then, the method uses the Adam optimizer to update the perturbation. GM optimizes the malicious perturbation via aligning the gradients of targeted and poisoned examples. Therefore, the method only needs one unroll step to compute the gradient. To optimize the malicious perturbation, GM attempts to minimize the negative cosine similarity between adversarial loss and natural loss, where the adversarial (natural) loss is the loss defined in the outer (inner) objective function in Eq. 10. In BP, the attacker crafts poisoned examples such that their representation in feature space is close to the targeted image. BP significantly improves the scalability and transferability of Convex Polytope attack [31]. FC, also known as Poison Frogs attack, has a similar idea to BP that it explores the feature space of images. The method aims to optimize the malicious examples such that they collide with the targeted example in the feacture space.

## B. Defenses against Data Poisoning Attacks

In this section, we detail the defense mechanisms that we compare with HINT in Section VI. Specifically, we consider the FRIENDS [8], adversarial training against delusive adversaries (ATDA) [9], and EPIC [10] algorithms.

**FRIENDS.** From the observation that each effective poison causes a local increase in the training loss and the whole poisoning set forms local regions in the loss space, FRIENDS aims to optimize the maximum perturbation without changing the model prediction. For each training example $x_i$, the friendly noise $\epsilon_i$ is generated as:

$$\epsilon_i = \arg\min_{\epsilon:||\epsilon||_\infty \leq \beta} D_{KL}(f_\theta(x_i + \epsilon)||f_\theta(x_i)) - \lambda||\epsilon||_2,$$

where $\lambda$ is the scaling factor. Besides the friendly noise, FRIENDS adds random noise from Gaussian, Uniform, or Bernoulli distribution to smooth the training loss.

**ATDA.** Following the theoretical proof that the adversarial risk can be the upper bound of natural risk, ATDA adapts the adversarial training technique to defend against untargeted poisoning attacks. By using FAT [32], the adversarial example $\tilde{x}_i$ can be generated as:

$$\tilde{x}_i = \arg\min_{\tilde{x}:||\tilde{x}-x||_p \leq \beta} l(f_\theta(\tilde{x}_i), y_i)$$
$$\text{s.t. } l(f_\theta(\tilde{x}_i), y_i) - \min_{y \in Y} l(f_\theta(\tilde{x}_i), y) \geq \tau,$$

where $\tau > 0$ is the margin such that an adversarial example would be misclassified.

**EPIC.** Different from the previously discussed methods, EPIC finds and drops malicious training points. From the observation that effective poisoned examples are often isolated from others of the same class in the gradient space, EPIC builds a set of medoids of each class, assigns other data points to its closest medoid, and drops isolated medoids during the training. The objective to find the set of medoids can be formulated as:

$$S \in \arg\min_{S \subseteq V, |S| < m} \sum_{i \in V} \min_{j \in S} ||\nabla l(f_\theta(x_i), y_i) - \nabla l(f_\theta(x_j), y_j)||_2,$$

where $m$ is the maximum number of medoids, $S$ is the index set of medoids, and $V$ is the index set of training data.

**Remark.** Our HINT is similar to FRIENDS and ATDA in principle as all three methods perturb training examples to defend against poisoning attacks. But HINT uses a different defense mechanism as aforementioned in Section IV. The healthy noise can capture local harmful/helpful regions formed by influential pixels and, by focusing on those important regions, the trained model is more resilient to attacks. Furthermore, HINT leverages the advantage of influence function, which can estimate the impact of each training example to the model loss, to choose a subset of training examples for perturbation.

## VI. Experiments

### A. Evaluation Setup

*1) Dataset and Model:* We focus on the image classification tasks and use MNIST and CIFAR-10 as our primary evaluation datasets. Each dataset is divided into three subsets (train, validation, and test), and a classification model is trained over the training set. Table I gives the specific details of our dataset construction. For MNIST, we train the CNN model using the SGD method with a learning rate of 0.01. For CIFAR-10, we train the ResNet-18 model using the SGD method with a Nesterov momentum of 0.9 and weight decay of $5 \times 10^{-4}$. Data augmentation techniques such as random crop and horizontal flip are also applied to the training images.

Additionally, the initial learning rate is set to 0.1, which is then decreased by a factor of 10 at epochs 30, 50, and 70.

TABLE I: Description of datasets and corresponding models.

| Dataset | $\|D_{trn}\|$ | $\|D_{val}\|$ | $\|D_{tst}\|$ | Model | Batch | Epoch |
|---|---|---|---|---|---|---|
| MNIST | 59000 | 1000 | 10000 | CNN | 128 | 30 |
| CIFAR-10 | 49000 | 1000 | 10000 | Resnet-18 | 128 | 80 |

*2) Poisoning Training Data:* We use the attack algorithms aforementioned in Section V-A to generate adversarial examples and inject them into the clean training data to produce our poisoned training set.

**Poisoning training data with untargeted attacks.** In this experiment, we consider the difficult scenario where the poisoned training data contains poisoned examples generated by four different malicious attacks: PGD, DAP, DURP, and DC. We train a victim model over the clean data for each attack type and generate the same amount of poisoned data per attack type based on a poison ratio $\rho$. For example, using MNIST with $\rho = 0.4$, we create a poisoned training dataset (with 59,000 images), which has 29,400 clean images and 24,500 poisoned images (4,900 images per attack type). For each attack type, the attacker's budget for perturbation is $\xi = 0.031$ (or 8/255) when attacking CIFAR-10 and $\xi = 0.3$ (or 76/255) when attacking MNIST. The setting is practical since the attacker tries to use a combination of multiple powerful attacks to efficiently generate malicious data.

**Poisoning training data with targeted attacks.** When analyzing the ability of defenses against targeted attacks, we only consider the CIFAR-10 as this dataset is commonly used for targeted attack evaluation. In this experiment, we evaluate the effectiveness of defense methods in both training-from-scratch (with GM and MP) and transfer learning (with BP and FC) settings as aforementioned in Section V-A. In both scenarios, we assume that the attacker knows the dataset, model architecture, and training scheme. However, they do not know the model weights. Similar to other papers aimed at defending against targeted attacks [8], [10], and the benchmarks in [33], we randomly choose the target and source classes, and then generate 490 poisoned training images ($\rho = 1\%$).

We run the attacks with the same setup, except for some specific hyper-parameters for each attack that we mention in this section. For GM, BP and FC, we train the victim model for 80 epochs and choose $\xi = 0.062$ (or 16/255) as the bound for malicious perturbation. The number of attack iterations is set at 250, 4,000, and 1,000, respectively. Other hyper-parameters follow the default values from the implementation of GM[2]. For MP, the poisoned data is downloaded from MetaPoison[3]. We use an equal mix of poison-dog target-bird and poison-frog target-plane settings in the evaluation. The bound for perturbation used in MetaPoison is $\xi = 0.031$.

For transfer learning scenario, we use a similar setup as in the from-scratch scenario to pre-train the victim model. Then, we randomly re-initialize the top layers while freezing the feature extraction layers. The victim model is then optimized on the transfer set. We construct the clean training set (for pre-training the victim model) with 44,100 clean images (90%) selected uniformly from each class. The transfer set consists of the remaining 4,900 images (10%) of the training set, in which 490 examples have been poisoned. We note that transfer learning setting used in our evaluation is not a real transfer learning setting as the clean training and the transfer sets come from the same original dataset. This, however, is the worst-case scenario that a defense has to consider to show its effectiveness, as the setting makes it easier for the attacker to succeed [4], [8].

[2]https://github.com/JonasGeiping/poisoning-gradient-matching

[3]https://github.com/wronnyhuang/metapoison

*3) HINT and Defense Baselines:* In this section, we briefly discuss the hyper-parameters for running each defense used in our experiment.

**Defense Baselines.** We choose FRIENDS+Bernoulli in our evaluation as it has the best performance according to [8]. Similar to the default setting of FRIENDS, we set $\beta = 0.062$. For ATDA, we choose FAT as the representation and set the default defender's budget $\beta$ as 0.25 based on the findings presented in [9]. When conducting experiments of ATDA against targeted attacks, we also use $\beta = 0.062$ for a fair comparison with other baselines. EPIC-0.1 is the default representation for EPIC with the poison drop interval $T$ and warm-up epochs $K$ as: $T = 4$, $K = 10$ for CIFAR-10; and $T = 2$, $K = 5$ for MNIST. We note that our evaluation uses default values as presented in the FRIENDS, ATDA, and EPIC papers [8]–[10] for all other hyper-parameters. Besides the above defenses, we additionally run experiments for a naïve training method (W/o Defense) using the default architecture of each dataset.

**HINT.** When computing the healthy noise, we only use the weights from the top layers of the model and discard all other weights (i.e., all weights from feature extraction layers). We set the update schedule for healthy noise at epochs 5, 15, and 40 for CIFAR-10 and at epochs 5 and 15 for MNIST. We choose the budget for the healthy noise to be $\beta = 0.062$ and the scaling factor to be $\gamma = 0.1$.

**Metrics.** For each experiment, we report the average and standard deviation of the test accuracy over five trials. When defending against targeted attacks, we additionally report the Attack Success Rate (ASR), which evaluates the success of an attack in changing the prediction of targeted examples. Note that an attack is considered successful only if it can change the predicted class to the intended class, following the evaluation in [4], [8], [10]. We use GPU Tesla V100 (32GB RAM) and CPU Xeon 6258R 2.7 GHz to conduct all experiments.

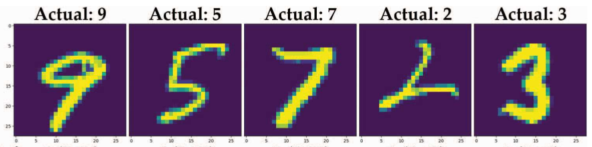
## *B. Results*

*1) Defending against Untargeted Attacks:* In this experiment, we evaluate the ability of different defense mechanisms to defend against poisoning attacks under different poison ratios. In Table II, we report the average and standard deviation of the test accuracy on CIFAR-10 and MNIST datasets. Each row shows the result for one poisoning set constructed from a particular poison ratio $\rho$. In most rows, HINT outperforms all other defense baselines, demonstrating our defense's effectiveness against poisoning attacks. For HINT and other methods, the test accuracy decreases when $\rho$ increases, which is not surprising since the attacker is able to inject more poisoning examples into the training set. With a large poison ratio ($p \geq 0.8$), our HINT method significantly reduces the effect of poisoned samples when we compared to naïve training. It also clearly outperforms other three defense baselines.

When we look at the results on the CIFAR-10 with a small poison ratio ($\rho \leq 0.4$), HINT is the only defense method that achieves better performance than naïve training, while other defenses have significant gaps below (more than $1.5\%$). A similar pattern happens to the results on the MNIST when $\rho \leq 0.6$. In this case, the model still learns both clean and malicious patterns but is able to focus more on the clean data. With FRIENDS and ATDA, when friendly or adversarial noises are added to clean training examples, the decision boundary may move further away from the original one. Recall from Section V-B where we explain that FRIENDS and ATDA add noise to the whole training dataset while our HINT method perturbs only a selected subset. Especially in the case that there is no poisoning attack performed ($\rho = 0.0$), our HINT method only loses $0.22\%$ test accuracy compared to naïve training on MNIST and even has better accuracy on CIFAR-10. These results match our previous observation in Section IV-C that the model trained with healthy noise does not lose its generalization ability. On both datasets, EPIC is the worst performer in the four defenses. This is because EPIC continuously detects and drops malicious examples, which works more efficiently when poisoned examples are far from their class in the gradient space. However, it is hard for EPIC to detect poisoned examples by untargeted attacks since untargeted attacks significantly perturb multiple training examples of every class to shatter the decision boundary, which also tampers the representations of training examples in the gradient space. It is even more challenging when we mix four different attack types with clean data in our setting.

In Figure 2, we show the predicted class, along with the probability of the prediction, of HINT and other baselines for five test examples from MNIST. We use red to denote each defense's success in preventing untargeted attacks. Each result shows the predicted class with the confidence score from the trained model using the corresponding defense method. Some pairs of classes have a high chance of confusion in prediction, which shows that untargeted attacks can successfully mislead the victim model and shift the decision boundary. For example, a trained model easily gets confusing an image of digit 7 for digit 2, or an image of digit 5 for digit 6. The results show that our method is more effective than other baselines in helping the trained model avoid the effect of untargeted attacks and give correct prediction results with a high confidence score.

*2) Defending against Targeted Attacks:* Table III shows a comparison of our HINT method with all baselines in terms of ASR and test accuracy on CIFAR-10. We compare the results based on ASR (lower is better) and test accuracy (higher is better). In this setting, our HINT method also significantly outperforms all other baselines. With all attacks, the naïve model fails all five trials, except for MP with 4/5. In particular, HINT, FRIENDS, and EPIC succeed in keeping the targeted example safe under the GM attack. However, our method achieves the highest test accuracy of the three defenses. Under other attack methods, HINT only fails once, while most other baselines fail multiple times. In most experiments, our HINT method also consistently has the best test accuracy compared to the other baselines. ATDA is the worst performer in terms of ASR compared to HINT, FRIENDS, and EPIC. This is unsurprising since ATDA uses adversarial perturbation to break the malicious examples by untargeted attacks, while the training data is poisoned by targeted attacks in this experiment.



| | | | | | |
|---|---|---|---|---|---|
| **W/o Def.** | 4 (0.87) | 6 (0.52) | 2 (0.38) | 1 (0.70) | 2 (0.78) |
| **HINT** | 9 (0.87) | 5 (0.94) | 7 (0.96) | 2 (0.48) | 3 (0.57) |
| **ATDA** | 8 (0.82) | 5 (0.73) | 2 (0.79) | 2 (0.66) | 2 (0.47) |
| **FRIENDS** | 8 (0.49) | 6 (0.63) | 2 (0.49) | 1 (0.72) | 2 (0.37) |
| **EPIC** | 4 (0.54) | 5 (0.87) | 7 (0.42) | 1 (0.60) | 8 (0.40) |

Fig. 2: Prediction of defense methods on MNIST test images under multiple untargeted attacks.

From the results, we can see that all the defense methods work well when defending against GM and MP. In training-from-scratch scenario, when training the model from scratch, defense methods will have a better chance to capture the malicious pattern from the poisoned examples and thereby be able to reduce their effect. Conversely, FRIENDS and EPIC have noticeably dropped in efficiency when defending against BP and FC in transfer learning scenario.

Figure 3 and 4 illustrate our results under MP and BP attacks, respectively. Similar to Figure 2, we use red color to denote the defense's success in preventing attacks from falsifying the target's class, and each result shows the predicted class with the confidence score. The model trained with HINT predicts the correct classes for most of the targeted examples with a high confidence score, consistent with the results of defending against untargeted attacks. Especially under the BP attack, HINT is the only method to successfully defend against the attack trials in the last two columns.

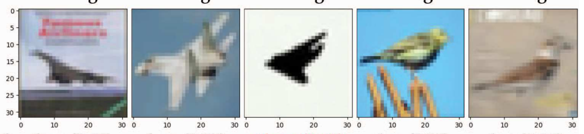
*3) Sensitivity Analysis:* Table IV shows the sensitivity analysis of HINT on the ratio of selected examples $r$ and the defender's budget $\beta$. We run our HINT method with different $r$ values to show how the size of the selected subset affects the model's performance. Note that when $r = 1.0$, the result is equivalent to the case of removing the subset selection module. The result shows that as $r$ increases, the test accuracy decreases. This observed trend highlights the contribution of our subset selection module in the whole training method. The trend also matches our previous observation that adding noise into clean examples may cause a drop in test accuracy.

TABLE II: Test accuracy (%) of our method and baselines when defending against multiple untargeted attacks (PGD+DAP+DURP+DC) on CIFAR-10 and MNIST datasets. In this evaluation, $r = 0.5$ and $\rho$ is the poison ratio.

| Dataset | $\rho$ | W/o Defense | HINT | ATDA | FRIENDS | EPIC |
|---|---|---|---|---|---|---|
| CIFAR-10 | 0.0 | **93.86 ± 0.26** | 93.64 ± 0.12 | 91.82 ± 0.17 | 89.25 ± 0.53 | 87.28 ± 0.71 |
| | 0.2 | 92.54 ± 0.05 | **92.67 ± 0.04** | 90.97 ± 0.17 | 89.37 ± 0.52 | 86.98 ± 0.79 |
| | 0.4 | 91.67 ± 0.12 | **92.08 ± 0.17** | 89.92 ± 0.25 | 88.53 ± 0.38 | 87.37 ± 0.40 |
| | 0.6 | 83.51 ± 0.17 | **91.94 ± 0.21** | 90.00 ± 0.38 | 88.90 ± 0.53 | 86.70 ± 0.11 |
| | 0.8 | 80.02 ± 0.36 | **91.20 ± 0.21** | 89.22 ± 0.57 | 87.75 ± 0.18 | 85.38 ± 0.73 |
| | 1.0 | 48.62 ± 0.74 | **90.79 ± 0.13** | 89.41 ± 0.58 | 86.76 ± 0.55 | 84.73 ± 0.32 |
| MNIST | 0.0 | 98.44 ± 0.02 | **98.87 ± 0.01** | 98.45 ± 0.11 | 98.33 ± 0.09 | 98.06 ± 0.05 |
| | 0.2 | 97.73 ± 0.01 | **98.42 ± 0.02** | 97.39 ± 0.29 | 97.58 ± 0.08 | 97.31 ± 0.16 |
| | 0.4 | 97.15 ± 0.03 | **98.01 ± 0.09** | 97.01 ± 0.09 | 97.14 ± 0.12 | 96.76 ± 0.19 |
| | 0.6 | 96.64 ± 0.12 | **96.87 ± 0.02** | 96.46 ± 0.14 | 96.58 ± 0.09 | 96.19 ± 0.19 |
| | 0.8 | 95.55 ± 0.12 | **95.88 ± 0.04** | 95.59 ± 0.09 | 95.49 ± 0.21 | 95.03 ± 0.20 |
| | 1.0 | 74.62 ± 0.94 | **88.92 ± 0.19** | 80.60 ± 1.01 | 77.08 ± 1.70 | 77.84 ± 1.54 |

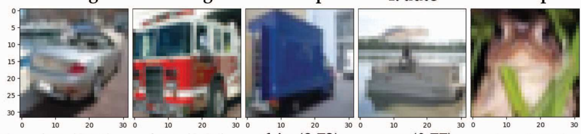
TABLE III: Attack Success Rate and test accuracy (%) of defense mechanisms against different targeted poisoning attacks. Experiments with MetaPoison is run without augmentation, following the setting in [5], [8].

| | | GM | MP | BP | FC |
|---|---|---|---|---|---|
| W/o Defense | ASR | 5/5 | 4/5 | 5/5 | 5/5 |
| | Test acc. | 93.69 ± 0.19 | 87.48 ± 0.41 | 91.41 ± 1.34 | 89.40 ± 1.39 |
| HINT | ASR | **0/5** | **1/5** | **1/5** | **1/5** |
| | Test acc. | **92.99 ± 0.26** | **87.15 ± 0.46** | **92.41 ± 0.54** | **92.21 ± 0.31** |
| ATDA | ASR | 3/5 | 3/5 | 4/5 | 4/5 |
| | Test acc. | 93.64 ± 0.27 | 87.45 ± 0.61 | 89.50 ± 1.51 | 88.37 ± 1.73 |
| FRIENDS | ASR | 0/5 | 1/5 | 3/5 | 2/5 |
| | Test acc. | 89.17 ± 0.41 | 78.26 ± 0.63 | 89.53 ± 0.66 | 88.84 ± 0.94 |
| EPIC | ASR | 0/5 | 2/5 | 4/5 | 3/5 |
| | Test acc. | 90.36 ± 0.43 | 86.68 ± 0.23 | 89.65 ± 2.34 | 89.19 ± 1.61 |



| | A: plane I: frog | A: plane I: frog | A: plane I: frog | A: bird I: dog | A: bird I: dog |
|---|---|---|---|---|---|
| **W/o Def.** | frog (0.66) | frog (0.99) | plane (0.78) | dog (0.92) | dog (0.99) |
| **HINT** | plane (0.97) | frog (0.99) | plane (0.72) | bird (0.73) | bird (0.57) |
| **ATDA** | plane (0.97) | frog (0.99) | bird (0.79) | dog (0.65) | dog (0.99) |
| **FRIENDS** | plane (0.99) | plane (0.51) | plane (0.99) | dog (0.97) | bird (0.92) |
| **EPIC** | plane (0.99) | frog (0.99) | plane (0.99) | bird (0.99) | dog (0.99) |

Fig. 3: Prediction of defense methods on CIFAR-10 test images under MP attack. "A" and"I" stand for actual and intended classes, respectively.



| | A: auto I: dog | A: truck I: frog | A: truck I: ship | A: ship I: auto | A: frog I: ship |
|---|---|---|---|---|---|
| **W/o Def.** | dog (0.57) | frog (0.96) | ship (0.72) | auto (0.77) | ship (0.93) |
| **HINT** | auto (0.72) | frog (0.89) | truck (0.78) | ship (0.65) | frog (0.52) |
| **ATDA** | auto (0.57) | frog (0.97) | ship (0.65) | auto (0.77) | ship (0.93) |
| **FRIENDS** | auto (0.69) | frog (0.76) | truck (0.78) | auto (0.69) | ship (0.85) |
| **EPIC** | auto (0.52) | frog (0.98) | ship (0.74) | auto (0.71) | ship (0.94) |

Fig. 4: Prediction of defense methods on CIFAR-10 test images under BP attack. "A" and"I" stand for actual and intended classes, respectively.

For hyper-parameter $\beta$, the test accuracy increases when moving from 0.031 to 0.062, and decreases when $\beta > 0.062$. This phenomenon is understandable since the healthy noise is not large enough to reduce the effect of malicious perturbation when we choose small $\beta$. However, when the healthy noise is sufficiently large, it makes the values of the pixels move too far from their original value and breaks the spatial relationship in the images.

*4) Execution Time:* We report the execution time of all defense methods used in our experiments in Table V. For HINT, we report the time for four variants with different $r$ values. When $r$ increases, the running time increases since the method needs to compute healthy noise for more training examples. When $r = 1.0$, the running time does not significantly

TABLE IV: Sensitivity analysis of HINT on $r$ and $\beta$ with untargeted attacks ($\rho = 0.6$).

| | Value | CIFAR-10 | MNIST |
|---|---|---|---|
| $r$ | 0.25 | 92.06 ± 0.08 | 97.13 ± 0.03 |
| | 0.5 | 91.94 ± 0.21 | 96.87 ± 0.02 |
| | 0.75 | 91.83 ± 0.11 | 96.62 ± 0.03 |
| | 1.0 | 91.72 ± 0.15 | 96.46 ± 0.05 |
| $\beta$ | 0.031 | 91.80 ± 0.18 | 96.76 ± 0.01 |
| | 0.062 | 91.94 ± 0.21 | 96.87 ± 0.02 |
| | 0.125 | 91.77 ± 0.13 | 96.74 ± 0.05 |
| | 0.251 | 91.40 ± 0.20 | 96.43 ± 0.01 |

increase compared to when $r = 0.75$ since HINT disables the subset selection module. For other defenses, the running time of FRIENDS is close to the default setting of HINT ($r = 0.5$), while ATDA and EPIC need more time to execute. ATDA generates and adds the noise for every training example in every training step. EPIC executes poison identification, which requires extensive resources to find medoids for each class in gradient space, to drop isolated points in every $T$ epoch.

TABLE V: Execution time of defenses with CIFAR-10.

| Method | Time (s) | Method | Time (s) |
|---|---|---|---|
| HINT ($r = 0.25$) | 2113 | W/o Defense | 1600 |
| HINT ($r = 0.5$) | 2427 | ATDA | 5760 |
| HINT ($r = 0.75$) | 2745 | FRIENDS | 2426 |
| HINT ($r = 1.0$) | 2875 | EPIC | 3147 |

## VII. Conclusion

In this work, we presented an effective robust training framework, HINT, that hardens the model with healthy influential-noise to protect machine learning models from poisoning attacks. Our method uses the influence function as a central mechanism to select examples with the highest impact on the model test loss and crafts the healthy influential-noise. Deep learning models trained with our HINT method are more resilient to the effect of malicious examples. Through comprehensive empirical evaluations, we demonstrate the effectiveness and stability of HINT in defending against powerful untargeted and targeted attacks (e.g., Deep Confuse, Gradient Matching, and Bulleyes Polytope) and its superiority over state-of-the-art defense baselines. These evaluations were conducted in a realistic scenario, highlighting the suitability of our defense mechanism for deployment in sensitive security settings. In future work, we will extend our approach to defend against other attack types, such as back door attacks, which involve injecting specific backdoor patterns into selected training data, manipulating test data to embed the triggers, and causing intentional misclassification.

**Reproducibility.** Our source code is available at https://github.com/minhhao97vn/HINT.

## Acknowledgements

This work was supported in part by National Science Foundation under awards 1946391, the National Institute of General Medical Sciences of National Institutes of Health under award P20GM139768, and the Arkansas Integrative Metabolic Research Center at University of Arkansas.

## References

[1] J. Feng, Q. Cai, and Z. Zhou, "Learning to confuse: Generating training time adversarial data with auto-encoder," in *NeurIPS*, 2019.
[2] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *ICLR (Poster)*. OpenReview.net, 2018.
[3] H. Aghakhani, D. Meng, Y. Wang, C. Kruegel, and G. Vigna, "Bullseye polytope: A scalable clean-label poisoning attack with improved transferability," in *EuroS&P*. IEEE, 2021.
[4] J. Geiping, L. H. Fowl, W. R. Huang, W. Czaja, G. Taylor, M. Moeller, and T. Goldstein, "Witches' brew: Industrial scale data poisoning via gradient matching," in *ICLR*. OpenReview.net, 2021.
[5] W. R. Huang, J. Geiping, L. Fowl, G. Taylor, and T. Goldstein, "Metapoison: Practical general-purpose clean-label data poisoning," in *NeurIPS*, 2020.
[6] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," in *NeurIPS*, 2018.
[7] J. Geiping, L. H. Fowl, G. Somepalli, M. Goldblum, M. Moeller, and T. Goldstein, "What doesn't kill you makes you robust (er): How to adversarially train against data poisoning," 2021.
[8] T. Y. Liu, Y. Yang, and B. Mirzasoleiman, "Friendly noise against adversarial noise: A powerful defense against data poisoning attack," in *NeurIPS*, 2022.
[9] L. Tao, L. Feng, J. Yi, S. Huang, and S. Chen, "Better safe than sorry: Preventing delusive adversaries with adversarial training," in *NeurIPS*, 2021.
[10] Y. Yang, T. Y. Liu, and B. Mirzasoleiman, "Not all poisons are created equal: Robust training against data poisoning," in *ICML*. PMLR, 2022.
[11] M. Van, W. Du, X. Wu, F. Chen, and A. Lu, "Defending evasion attacks via adversarially adaptive training," in *IEEE International Conference on Big Data*. IEEE, 2022.
[12] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. M. Molloy, and B. Srivastava, "Detecting backdoor attacks on deep neural networks by activation clustering," in *SafeAI@AAAI*, 2019.
[13] N. Baracaldo, B. Chen, H. Ludwig, J. A. Safavi, and R. Zhang, "Detecting poisoning attacks on machine learning in iot environments," in *ICIOT*. IEEE Computer Society, 2018.
[14] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," in *NeurIPS*, 2018.
[15] R. D. Cook and S. Weisberg, *Residuals and influence in regression*. New York: Chapman and Hall, 1982.
[16] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *ICML*, ser. Proceedings of Machine Learning Research. PMLR, 2017.
[17] B. Biggio, B. Nelson, and P. Laskov, "Support vector machines under adversarial label noise," in *ACML*. JMLR, 2011.
[18] ——, "Poisoning attacks against support vector machines," in *ICML*, 2012.
[19] A. Newell, R. Potharaju, L. Xiang, and C. Nita-Rotaru, "On the practicality of integrity attacks on document-level sentiment analysis," in *AISec@CCS*. ACM, 2014.
[20] M. Zhao, B. An, W. Gao, and T. Zhang, "Efficient label contamination attacks against black-box learning models," in *IJCAI*, 2017.
[21] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *AISec@CCS*. ACM, 2017.
[22] H. Chang, T. D. Nguyen, S. K. Murakonda, E. Kazemi, and R. Shokri, "On adversarial bias and the robustness of fair machine learning," *arXiv preprint arXiv:2006.08669*, 2020.
[23] M. Van, W. Du, X. Wu, and A. Lu, "Poisoning attacks on fair machine learning," in *DASFAA*. Springer, 2022.
[24] M. Weber, X. Xu, B. Karlas, C. Zhang, and B. Li, "RAB: provable robustness against backdoor attacks," in *SP*. IEEE, 2023.
[25] E. Borgnia, V. Cherepanova, L. Fowl, A. Ghiasi, J. Geiping, M. Goldblum, T. Goldstein, and A. Gupta, "Strong data augmentation sanitizes poisoning and backdoor attacks without an accuracy tradeoff," in *ICASSP*. IEEE, 2021.
[26] E. Borgnia, J. Geiping, V. Cherepanova, L. Fowl, A. Gupta, A. Ghiasi, F. Huang, M. Goldblum, and T. Goldstein, "Dp-instahide: Provably defusing poisoning and backdoor attacks with differentially private data augmentations," *arXiv preprint arXiv:2103.02079*, 2021.
[27] M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *CCS*. ACM, 2016.
[28] B. Jayaraman and D. Evans, "Evaluating differentially private machine learning in practice," in *USENIX Security Symposium*, 2019.
[29] N. Agarwal, B. Bullins, and E. Hazan, "Second-order stochastic optimization for machine learning in linear time," *J. Mach. Learn. Res.*, 2017.
[30] S. Basu, P. Pope, and S. Feizi, "Influence functions in deep learning are fragile," in *ICLR*. OpenReview.net, 2021.
[31] C. Zhu, W. R. Huang, H. Li, G. Taylor, C. Studer, and T. Goldstein, "Transferable clean-label poisoning attacks on deep neural nets," in *ICML*. PMLR, 2019.
[32] J. Zhang, X. Xu, B. Han, G. Niu, L. Cui, M. Sugiyama, and M. S. Kankanhalli, "Attacks which do not kill training make adversarial learning stronger," in *ICML*. PMLR, 2020.
[33] A. Schwarzschild, M. Goldblum, A. Gupta, J. P. Dickerson, and T. Goldstein, "Just how toxic is data poisoning? A unified benchmark for backdoor and data poisoning attacks," in *ICML*. PMLR, 2021.