Supplementary Material for Poison is Not Traceless: Fully-Agnostic Detection of Poisoning Attacks

To ensure reproducibility, we provide additional details for the paper in this section. We include details on FALFA (including its time complexity), the hardware and software configurations for our experiments, detailed dataset descriptions, and additional results.

Details of FALFA

In a poisoning attack, the attacker can directly manipulate the training data $\mathcal{D}_{\text{tr}} := \{(\mathcal{X}_{\text{tr}}, \mathcal{Y}_{\text{tr}})\}$ and interact with the training process of a model f. Based on this information, the attacker creates a poisoned training set $\mathcal{D}'_{\text{tr}} := \{(\mathcal{X}'_{\text{tr}}, \mathcal{Y}'_{\text{tr}})\}$, on which a poisoned model f' is trained, by modifying or injecting examples into $\mathcal{D}_{\text{tr}}$. To increase the test error, the attacker's goal is to maximize the loss on the clean test set $\mathcal{D}_{\text{te}} := \{(\mathcal{X}_{\text{te}}, \mathcal{Y}_{\text{te}})\}$.

The attack objective of a label-flipping attack is:

$$\min_{\mathcal{D}'_{\text{tr}}} \ell(\mathcal{D}'_{\text{tr}}, f') - \ell(\mathcal{D}_{\text{te}}, f'),$$

which can be interpreted as the adversary making the discrepancy between f and f' as large as possible. FALFA is derived from Equation 1, which we rewrite with the adversary's cost constants as:

$$\begin{aligned} \min_{\mathcal{Y}'_{\text{tr}}} \quad & \ell(f'(\mathcal{X}_{\text{tr}}), \mathcal{Y}'_{\text{tr}}) - \ell(f(\mathcal{X}_{\text{tr}}), \mathcal{Y}'_{\text{tr}}) \\ \text{s.t.} \quad & \sum_{i=1}^{n} |y'_i - y_i| \leq n\epsilon, \\ & y'_i \in \{0, 1\} \text{ for } i = 1, \ldots, n. \end{aligned} \tag{1}$$

A *Neural Network* (NN) optimizes the cross entropy loss $\ell(p_i, y_i) = y_i[-\log(p_i) + \log(1 - p_i)]$ and normalizes its outputs by the softmax function $p_i = \exp(x_i)/\sum_j \exp(x_j)$. In this case, we can transform Equation 1 into a linear programming problem in three steps:

1. We expand the objective function in Equation 1 using the cross entropy loss to $\ell(f'(\mathcal{X}_{\text{tr}}), \mathcal{Y}'_{\text{tr}}) - \ell(f(\mathcal{X}_{\text{tr}}), \mathcal{Y}'_{\text{tr}}) = (\alpha - \beta)\,\mathcal{Y}'_{\text{tr}}$ with $\alpha := -\log(p') + \log(1 - p')$ and $\beta := -\log(p) + \log(1 - p)$. β only depends on the original classifier f and clean data $\mathcal{D}_{\text{tr}}$, so it is a constant and can be computed beforehand.
2. Since the problem is a binary classification, we simplify the condition $\sum_{i=1}^{n} |y'_i - y_i| \leq n\epsilon$ to $\lambda \cdot \mathcal{Y}'_{\text{tr}} \leq n\epsilon + \lambda \cdot \mathcal{Y}_{\text{tr}}$ using a multiplier $\lambda = 1$ if $y_i = 0$ and -1 otherwise. λ is a constant vector as it only depends on $\mathcal{Y}_{\text{tr}}$. We obtain the multiplier λ by generalizing all the combinations between y_i and y'_i in a binary classification task, as shown in Table 1.

3. Using the simplex method, we relax the boundary condition $y'_i \in \{0, 1\}$ to $0 \leq y'_i \leq 1$, because the solution is guaranteed on the edges.

Overall, Equation 1 becomes the following linear programming problem:

$$\begin{aligned} \min_{\mathcal{Y}'_{\text{tr}}} \quad & (\alpha - \beta)\, \mathcal{Y}'_{\text{tr}} \\ \text{s.t.} \quad & \lambda \cdot \mathcal{Y}'_{\text{tr}} \leq n\epsilon + \lambda \cdot \mathcal{Y}_{\text{tr}}, \\ & 0 \leq y'_i \leq 1 \text{ for } i = 1, \ldots, n. \end{aligned} \tag{2}$$

Table 1. All combinations for $|y'_i - y_i|$. By introducing a multiplier λ, $\lambda \cdot (y'_i - y_i)$ is equivalent to $|y'_i - y_i|$.

y_i	y'_i	$\lvert y'_i - y_i \rvert$	λ
0	0	0	1
0	1	1	1
1	0	-1	-1
1	1	0	-1

The full algorithm of FALFA is shown in Algorithm 1.

Algorithm 1 FALFA (Fast Adversarial Label Flipping Attack)

Require: Original training set $\mathcal{D}_{\text{tr}} := \{(\mathcal{X}_{\text{tr}}, \mathcal{Y}_{\text{tr}})\}$, budget parameter ϵ, classifier f trained on $\mathcal{D}_{\text{tr}}$
Ensure: Poisoned training labels $\mathcal{Y}'_{\text{tr}}$
$p \leftarrow \text{Softmax}(f(\mathcal{X}_{\text{tr}}))$;
$\beta \leftarrow -\log(p) + \log(1 - p)$;
$\lambda \leftarrow 1$ if $(y_i == 1)$ else $-1, \forall y_i \in \mathcal{Y}_{\text{tr}}$;
$\mathcal{Y}'_{\text{tr}} \leftarrow$ Randomly flip $n \cdot \epsilon$ examples on $\mathcal{Y}_{\text{tr}}$;
$f' \leftarrow f$;
while $\mathcal{Y}'_{\text{tr}}$ does not converge **do**
    Retrain classifier f' using $(\mathcal{X}_{\text{tr}}, \mathcal{Y}'_{\text{tr}})$;
    $p' \leftarrow \text{Softmax}(f'(\mathcal{X}_{\text{tr}}))$;
    $\alpha \leftarrow -\log(p') + \log(1 - p')$;
    Update $\mathcal{Y}'_{\text{tr}}$ by solving Eq. 2;
return $\mathcal{Y}'_{\text{tr}}$;

FALFA is a suitable poison generator for DIVA in the following ways:

1. It is efficient on models using cross-entropy loss. Apart from the poisoned classifier f' itself, only p' and α require updates. Anything else is constant and can be computed beforehand.

2. The computation time is invariant to poisoning rates. Because Equation 2 is solved by a linear programming solver, we update $\mathcal{Y}'_{\text{tr}}$ without looping through all permutations.
3. It is guaranteed solvable at any poisoning rate since we set the initial solution for $\mathcal{Y}'_{\text{tr}}$ by randomly flipping $n \cdot \epsilon$ examples.
4. FALFA is fast (we noticed a speed-up of at least factor 20 in our experiments; see our supplementary) which facilitates using a large meta-database.

Time Complexity of FALFA

FALFA is more computationally efficient than ALFA and PoisSVM by a substantial margin. Linear programming is an exponential-time algorithm, the time complexity is around $O(n^{2.5})$. Xiao *et al.*'s ALFA creates a copy of $\mathcal{Y}_{\text{tr}}$ in the linear programming step, so n is essentially doubled. Paudice *et al.*'s ALFA on NN is slower than Xiao *et al.*'s since it traverses all combinations of $\mathcal{Y}_{\text{tr}}$ instead of using linear programming. FALFA uses linear programming but without doubling $\mathcal{Y}_{\text{tr}}$, resulting in an approximately $2^{2.5} \approx 5.6$ times faster than ALFA on each iteration. Our test shows that FALFA converges at two iterations on average, but ALFA takes more than five iterations to converge. In the worst-case scenario, FALFA poisons the CMC dataset in 22.4 ± 8.6 secs, while ALFA requires 405.8 ± 348.4 secs, and PoisSVM took over 2 hours. We observe the minimal difference on Breastcancer, where FALFA completes the task at 5.3 ± 1.9 secs, and it takes ALFA 7.4 ± 5.6 secs.

Hardware and Software Configurations

All experiments are conducted on a workstation with the following configurations:

- CPU: AMD Ryzen 9 5900 24 threads @ 4.4GHz
- GPU: Nvidia GeForce RTX 3090 24GB
- Memory: 64GB
- Operating System: Ubuntu 20.04.3 LTS
- Software: Python 3.8.10, PyTorch 1.10.1+cu113, scikit-learn 1.0.2

The baseline data poisoning attacks are obtained from *Adversarial Robustness Toolbox* (ART) 1.9.1 [1] and *Secure and Explainable Machine Learning in Python* (SecML) 0.15 [2].

The execution time mentioned in the paper is evaluated using the environment above.

[1] Source: https://github.com/Trusted-AI/adversarial-robustness-toolbox
[2] Source: https://github.com/pralab/secml

Real-World Datasets. All datasets are obtained from the UCI Machine Learning Repository [3]. We apply standardization on all datasets during the preprocessing.

For multi-class classification tasks, we convert the dataset into binary based on the following:

- **Abalone:** If the 'Rings' attribute is less than 10, we assign the example to the negative class; Else, assign to the positive class. We exclude the categorical attribute – 'Sex' and the output label – 'Rings' from the inputs.
- **CMC:** has 3 output classes: 1. No-use, 2. Long-term, and 3. Short-term. If the class is 'No-use', assign it to the negative class; Else, to the positive class.
- **Texture:** It has 10 output classes. We use a subset which contains examples labeled as '3' and '9'. If the class is '3', assign it to the negative class; Else, to the positive class.
- **Yeast:** It has 10 output classes. We select ''0 and '7', the top two classes sorted by sample size. If the class is '0', assign it to the negative class; Else, to the positive class.

Synthetic Datasets. Table 2 shows the parameters we used to generate synthetic datasets.

Table 2. Hyper-parameters for generating synthetic data

Parameter	Range
Sample size	$\{1000, 1001, \ldots, 2000\}$
# of informative features	$\{4, 5, \ldots, 30\}$
# of redundant features	$\{0, 1, \ldots, 5\}$
Class ratio	$[0.4, 0.6]$

Additional Results

Classifiers' Performance. Table 3 shows the performance of classifiers trained on poison-free data.

Performance Loss. Fig. 1 shows the performance loss on all real datasets.

[3] Source: https://archive.ics.uci.edu/ml

Table 3. Summary of the training set size (n), number of features (m), Positive Label Rate (PLR), average accuracy (%) for training and test sets across all classifiers, and difficulty (Easy/Normal/Hard).

Dataset	n	m	PLR	Train Acc.	Test Acc.	Diff.
Abalone	1600	7	0.50	79.9 ± 0.7	76.5 ± 0.5	N
Australian	552	14	0.45	91.5 ± 3.1	81.9 ± 2.1	N
Banknote	1097	4	0.44	100.0 ± 0.0	100.0 ± 0.0	E
Breastcancer	455	30	0.63	99.3 ± 0.2	95.0 ± 2.5	E
CMC	1178	9	0.77	79.9 ± 2.8	77.5 ± 0.6	N
HTRU2	1600	8	0.50	94.8 ± 0.5	92.6 ± 0.9	E
Phoneme	1600	5	0.50	89.7 ± 6.3	85.6 ± 1.3	N
Ringnorm	1600	20	0.50	99.4 ± 0.4	97.8 ± 1.1	E
Texture	800	40	0.50	100.0 ± 0.0	99.8 ± 0.5	E
Yeast	713	8	0.48	73.5 ± 4.7	65.8 ± 1.6	H

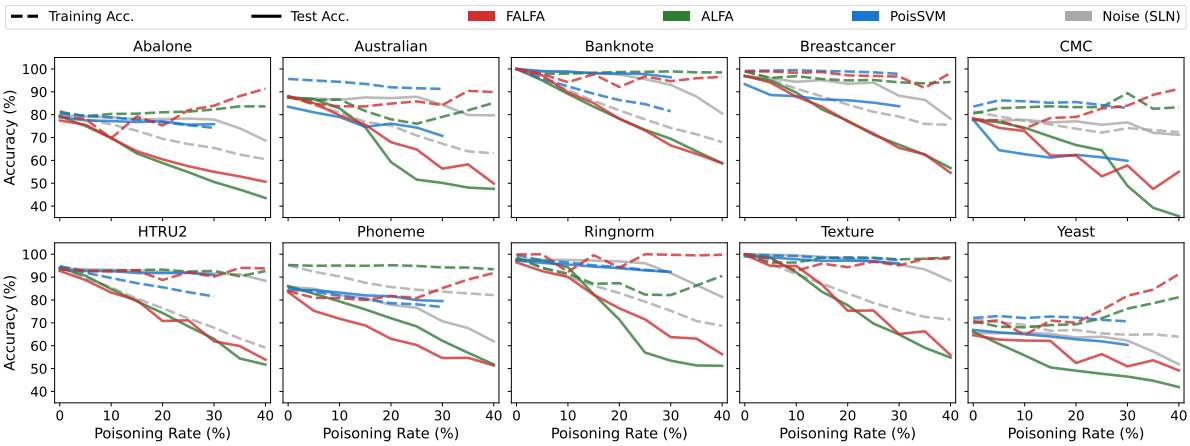

Fig. 1. Train and test accuracy at various poisoning rates when classifiers under SLN, PoisSVM, and FALFA attacks.

C-Measures on Clean and Poisoned Data. When no poisoning attack is present, the C-Measures strongly correlate to the classifier's test accuracy as can be seen in Fig. 2a. When the dataset has been poisoned, the C-Measures react to it. Despite a performance drop on the training accuracy of $1.0 \pm 5.6\%$, we observe a correlation drop across all measures when measuring on poisoned data, as shown in Fig. 2b. This matches the test accuracy drop, which indicates the data becomes more complex, despite only minor changes in the training accuracy.

Performance Loss at a Low Poisoning Rate. Here, we present the performance loss at a low poisoning rate (10%) in Table 4. This is the test accu-

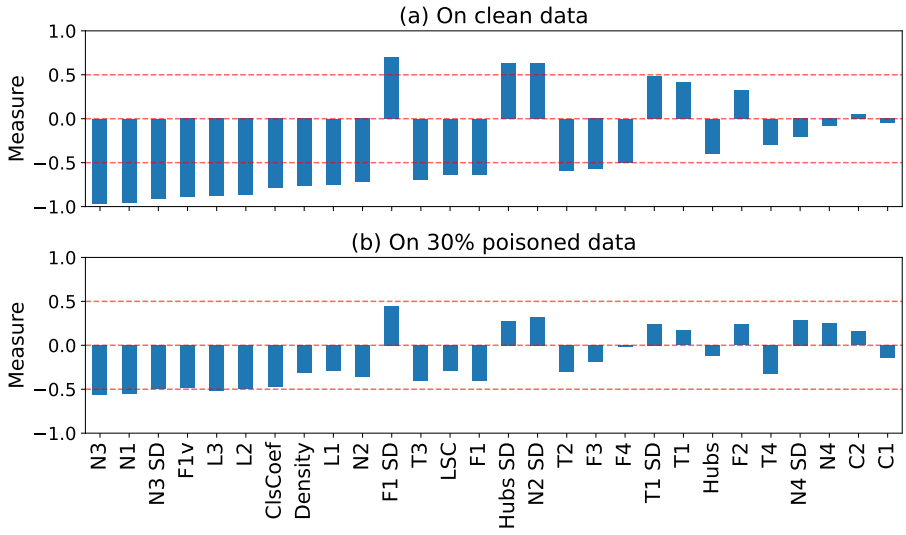

Fig. 2. (a): Correlation of each measure in C-Measures to the test accuracy on synthetic datasets without poisoning attacks. (b): Same correlation but measured on the datasets containing 30% poisoning examples.

racy difference before and after the attack. PoisSVM has no meaningful impact ($< 2\%$) on the classifiers' performance in 7 out of 10 datasets. Meanwhile, SLN leads to minor performance improvement on CMC and Yeast.

Table 4. Performance loss (%) after attacked by a poisoning attack with 10% poisoning rate.

Dataset	SLN	PoisSVM	ALFA	FALFA
Abalone	0.8 ± 0.7	1.8 ± 0.8	9.5 ± 1.9	7.7 ± 1.7
Australian	0.7 ± 0.5	4.5 ± 3.9	4.9 ± 4.0	8.3 ± 3.8
Banknote	1.4 ± 2.3	1.1 ± 1.1	10.9 ± 2.5	10.3 ± 2.9
Breastcancer	2.5 ± 0.7	5.3 ± 4.6	7.2 ± 2.0	9.1 ± 2.7
CMC	-0.2 ± 0.7	15.1 ± 4.7	3.5 ± 3.0	5.7 ± 3.3
HTRU2	0.7 ± 0.3	0.7 ± 1.3	9.2 ± 3.1	9.4 ± 2.4
Phoneme	3.5 ± 2.9	0.9 ± 2.1	6.8 ± 0.7	11.6 ± 2.1
Ringnorm	0.1 ± 0.3	1.7 ± 0.5	3.2 ± 2.5	6.4 ± 2.9
Texture	0.5 ± 1.1	1.2 ± 0.8	7.9 ± 4.6	4.9 ± 3.9
Yeast	-0.2 ± 1.6	1.9 ± 3.8	10.4 ± 4.9	2.3 ± 4.6

Additional Results When Varying Thresholds. The results for DIVA with a single threshold set to 98% TNR are given in the paper. Here we provide additional results by varying the threshold. The first row in heatmaps represents the detectors' performance when there is no poison labels.

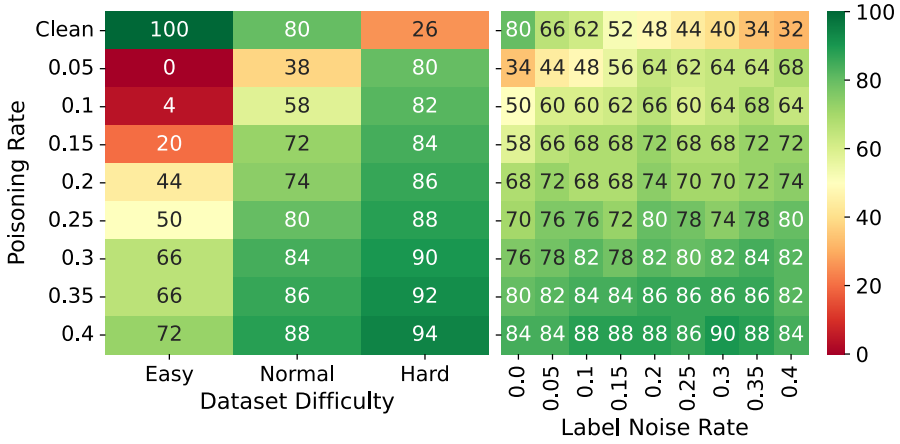

Fig. 3. DIVA's performance with a fixed threshold set to 80% TNR.

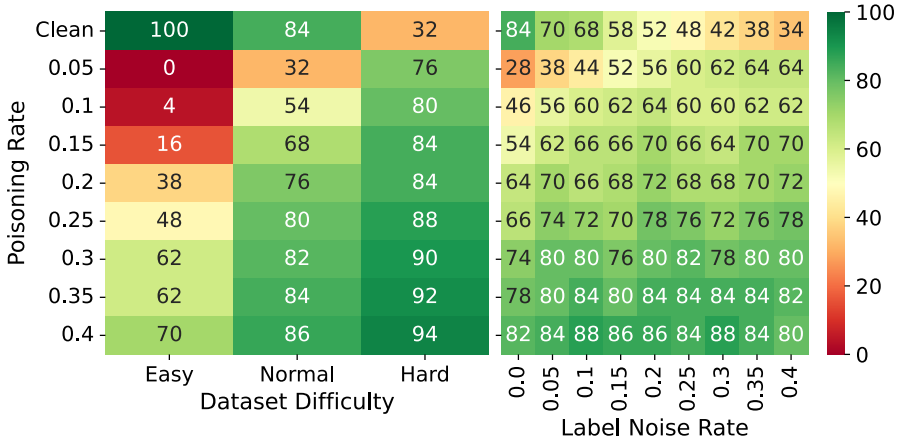

Fig. 4. DIVA's performance with a fixed threshold set to 85% TNR.

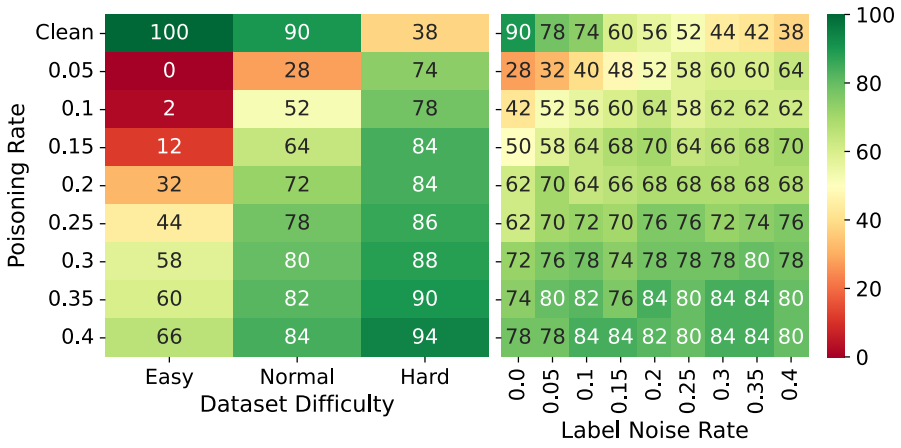

Fig. 5. DIVA's performance with a fixed threshold set to 90% TNR.

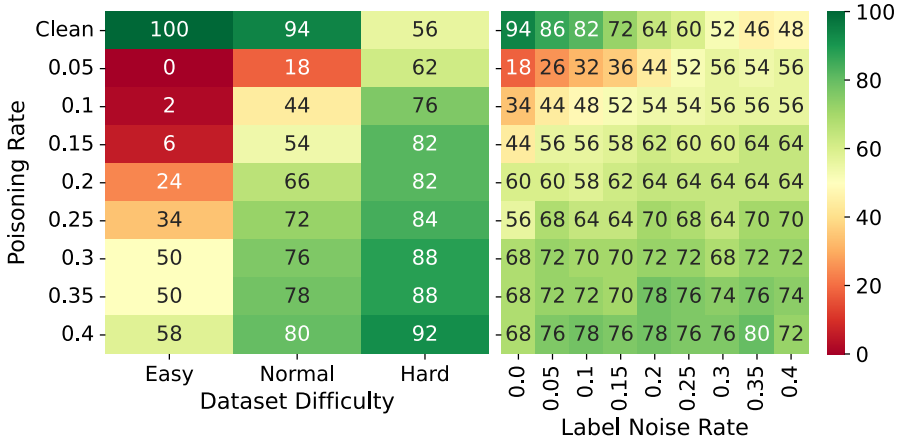

Fig. 6. DIVA's performance with a fixed threshold set to 95% TNR.

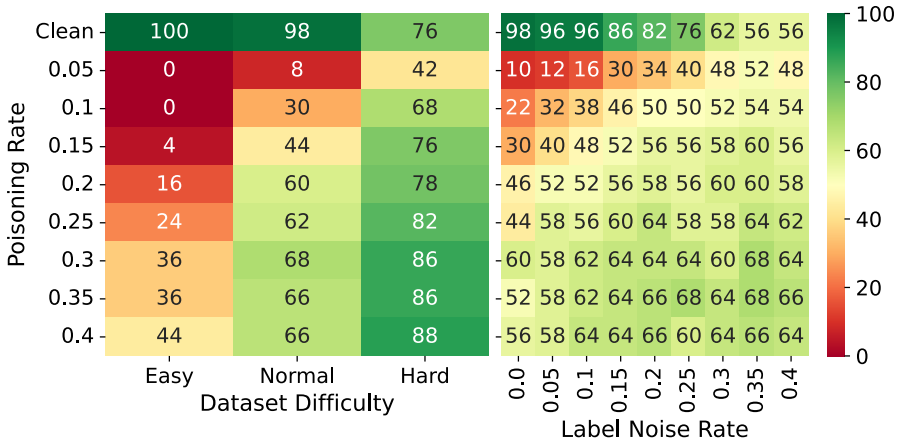

Fig. 7. DIVA's performance with a fixed threshold set to 99% TNR.