USENIX'23 Artifact Appendix: "Meta-Sift: How to Sift Out a Clean Subset in the Presence of Data Poisoning?"

Yi Zeng[1,2], Minzhou Pan[1], Himanshu Jahagirdar[1], Ming Jin[1], Lingjuan Lyu[2], and Ruoxi Jia[1]

[1]Virginia Tech, Blacksburg, VA 24061, USA
[2]Sony AI, Tokyo, 108-0075, Japan

A Artifact Appendix

A.1 Abstract

This artifact appendix focuses on road-mapping the **three** main claims we developed in the "Meta-Sift" paper:

- **Defense performance is sensitive to the purity of the base set** (referring to Takeaway #1, **Section 1** and **Section 2.1**): Representative works of defense methods against data poisoning are only effective when they can access a small, clean-held-out dataset (base set). When infiltrated with poisoned samples, the defense effects of these methods are significantly impaired.

- **Both existing automated methods and human inspection fail to identify a clean subset with high enough precision** (referring to Takeaway #2, **Section 1** and **Section 2.3, 2.4**): To evaluate existing methods for identifying a clean base set from a poisoned dataset and conducting a human study, we found that these techniques cannot satisfy the necessary access to the base set required to initiate the defenses mentioned above.

- **Our proposed solution, Meta-Sift** (our main contribution, referring to Takeaway #3, **Section 1**, the implementation of the proposed method in **Section 3**, and results in **Section 4.2**), utilizes a new but intuitive idea that training a model on the clean portion of a (corrupted) dataset will perform poorly on the poisoned portion and vice versa, which is a splitting problem that helps to identify the clean samples and can be described as a bi-level optimization (Eqn. 11, 12). We have introduced a suite of techniques to build Meta-Sift to resolve this splitting problem, resulting in efficient and effective solutions.

A.2 Description & Requirements

The provided artifacts focus on reproducing results with the GTSRB dataset (smaller and easy to download). Our implementation has been tested on our server and can be accessed via SSH, along with the required software environments and dependencies. By running the implementation on our provided backend, you can reproduce examples of the experiments that support our claims. If you choose to re-implement everything on your hardware/software, it might require a machine with the following minimum requirements:
Hardware requirements: CPU: 1×AMD EPYC 7763 64-Core Processor; GPU: 1×NVIDIA RTX A6000 48 GB.
Software requirements: Operating system: Linux (Ubuntu 20.04); Python 3.9; CUDA 11.8; cuDNN 8.7.0; Required Python packages: h5py (version 3.6.0 or later); imageio (version 2.9.0 or later); numpy (version 1.21.5 or later); Pillow (version 9.4.0 or later); torch (version 1.13.0 or later); torchvision (version 0.14.0 or later); tqdm (version 4.64.0 or later); jupyter notebook (version 6.4.8 or later).

A.2.1 Security, privacy, and ethical concerns

We do not collect data or fingerprints while the evaluators use our provided backend for re-implementation. The concern is not applicable if the evaluator uses their own hardware/software platform.

A.2.2 How to access

We provide a stable released version of our implementation via the following stable reference of the GitHub link: https://github.com/ruoxi-jia-group/Meta-Sift/releases/tag/artifact. We have created an anonymous SSH account for evaluators to access our hardware platform. Please directly contact the authors for further instructions on using our provided backend.

A.2.3 Hardware dependencies

We highly recommend the evaluators **use our provided hardware backend** for reproducing the results.

A.2.4 Software dependencies

We have tested our artifact in Linux OS (Ubuntu 20.04), along with Python 3.9, CUDA 11.8, and cuDNN 8.7.0. Python packages required, including h5py (version 3.6.0 or later), imageio (version 2.9.0 or later), numpy (version 1.21.5 or later), Pillow (version 9.4.0 or later), torch (version 1.13.0 or later),

torchvision (version 0.14.0 or later), tqdm (version 4.64.0 or later), and jupyter notebook (version 6.4.8 or later). Before evaluating the artifact, please ensure all the necessary software components and packages are installed and configured correctly. For simplicity, we have provided the "**metasift.yml**" file so that one can easily install the required environment with Conda. Detailed instruction is **listed in the GitHub release**.

A.2.5 Benchmarks

Throughout our artifacts, we mainly reproduce the results on the GTSRB dataset [1], which features traffic sign images scaled to 32×32 pixels. This dataset includes 39,209 training samples and 12,630 testing samples, both with class-imbalanced distributions, providing a real-life set of data for our analysis. In the "quick_start.ipynb," we utilized a VGG-16 model that was poisoned with BadNets backdoor attack in "class 38" as the poison model (model structure and poisoned parameters obtained from [2]) that will be using I-BAU [2] for purification. In the subsequent Meta-Sift processes, we used a ResNet-18 [3] model as the feature extractor θ for sifting.

A.3 Set-up

To prepare the environment for evaluating our artifact, one needs to first log in to our server (**contact us!**) or a server with minimum hardware configuration required. After that, simply follow the Conda command provided in the GitHub release will be able to set up the required environment.

A.3.1 Installation

To install the required software environment and the artifacts, first, download all the code from the GitHub release. Once the artifacts are downloaded, one needs to navigate to the directory using the command line and install the required software dependencies. Meanwhile, the GTSRB dataset can be found here. Please **download the required GTSRB dataset and put it under the "./dataset" folder**. By following these steps, one should have a properly configured environment ready for evaluation.

A.3.2 Basic Test

Please **go to the "quick_start.ipynb," and try to run the first block.** If no error pops up, it indicates that the prerequisite environment has been successfully installed.

A.4 Evaluation workflow

This artifact consists of three functional parts that accounts for three experiments:

- "./quick_start.ipynb" contains three example experiments supports each claim (Takeaway #1, #2 and #3);
- "./human_exp" folder provides the tool and a Narcissus [4] poisoned image dataset we built for human study;
- "main.py" implements our proposed method, which accounts for the core contribution, and one can thoroughly evaluate Meta-Sift with different poison settings.

A.4.1 Major Claims

Please refer to Section A.1 for a detailed recap of our claims. Mainly, we have claimed that:

(C1): Defense effects are sensitive to the purity of the base set (Takeaway #1 in **Section 1**, results in **Section 2.1**).

(C2): Both existing automated methods and human inspection fail to identify a clean subset with high enough precision (Takeaway #2 in **Section 1**, results and analysis in **Section 2.3** and **2.4**).

(C3): Our proposed solution, Meta-Sift, can obtain a clean subset with the required budget in many poison situations (Takeaway #3 in **Section 1**, results in **Section 4.2** or **Table 15** for the GTSRB).

A.4.2 Experiments

(E1): *[1 human-minutes + 10 compute-minutes]:* ***C1****, part of* ***C2*** *(Automated methods fail to identify a pure enough clean subset), and one experiment for* ***C3****.*

How to:

- *First, we reproduce one experiment in Table 1 to support* ***C1****. The code first loads a poisoned small VGG-16 [2] that has been poisoned with BadNets [5] targeting class 38. The code then executes I-BAU [2] for backdoor removal with a randomly selected 1000-size clean base set. We can observe that the ASR of the model is successfully mitigated. Then, we consider a poisoned base set with* ***8*** *poisoned samples mixed in, and we can observe the efficacy of the defense is largely impaired.*
- *Moving forward, we reproduce one experiment in Table 2 to support* ***C2*** *on automatic methods. The notebook implements an automated sifting method which we termed Distance to the Class-Means (DCM), to sift out a base set and evaluate the Normalized Corruption Rate (NCR) of the selected base set. The resulting value is much larger than 0, indicating that automated methods fail to identify a clean subset with sufficient precision within the given 1000 selection budget.*
- *Finally, we reproduce Meta-Sift's result on BadNets poisoned GTSRB to confirm* ***C3****. Upon running the code in the notebook, we can observe that the NCR is 0, indicating that we successfully identified a pure-clean base set within the same 1000 budget.*

Preparation: *Please complete Section A.3 first.*
Execution: *Run "quick_start.ipynb."*
Results: *Expected results are in the current notebook.*

(E2): *[30 human-minutes]:* ***C2*** *on humans' inability to identify the poisons with enough precision.*

How to: *We suggest the evaluator should not check the visual examples in our paper before completing this experiment for non-necessary bias. Please first go to the "./exp_human" folder, unzip the "img.zip", and open "huamn_label_interface.html" with your web browser. The interface allows human labelers to assess whether an image is poisoned and output the results with the "yes" or "no" button. Once one successfully goes through all 1000 samples, the browser will automatically download the "result.csv" file. Reviewers can compare these results with the "gound_trouth.xlsx" to calculate the false-negative rate (FNR).*

Preparation: *A web browser is required.*

Execution: *After inspection, the browser will automatically download a "result.csv" file. Please copy the first column from "result.csv" and paste it over column B, "ground_trouth.xlsx," to get the final FNR. One can compare the results with the results in Figure 3.*

Results: *The results should be quite similar to our results in Figure 3, i.e., end up with high FNR.*

(E3): *[1 human-minutes + 20 compute-minutes]: Further evaluation of **C3** (main contribution) with representative poisoning attack under each category.*

How to: *Please refer to the commands in the GitHub release and run them in a terminal with the required Conda environment.*

Preparation: *Please complete Section A.3 first.*

Execution: *Run commands one by one in a terminal.*

Results: *The sifting results over these three poison settings should all end up with an NCR equal to 0 with our proposed method at a selection budget of 1000 (default).*

A.5 Notes on Reusability

In the current release, we provide a plug-in (optional) function that allows for adopting Meta-Sift to identify a clean base set with a specific selection budget from any dataset. When using Meta-Sift on a new dataset, choosing hyperparameters with care is essential. We suggest using a pseudo-poisoned dataset by applying the Narcissus attack [4] with a 10% poison ratio on the top of the provided dataset (i.e., despite what poison the original dataset is poisoned with, we manually introduce the Narcissus attack over the whole dataset). Narcissus is the most stealthy but effective attack we have found in our empirical study. Once you have the Narcissus poisoned dataset on top of the given dataset, input it into Meta-Sift and fine-tune the hyperparameters to reduce the output NCR. When you have a set of hyperparameters that can help you achieve 0 NCR on this pseudo-poisoned dataset, it should perform well in sifting out a 0 NCR base set for your provided dataset.

To adjust the main hyperparameters:

"**-warmup_epochs**" determines the number of rounds the model should be pre-trained on the dataset before starting the sift. For example, in GTSRB, the model's accuracy should be kept around 50% after warmup, while the accuracy of a well-trained model is over 90%.

"**-batch_size**" is an important parameter influencing performance. Keeping the "batch_size" small allows the model to be updated more frequently, which might lead to better NCR over low-resolution datasets.

"**-v_lr**" determines the learning rate of the weight-assigning network, and it should be adjusted for the dataset size. This parameter should be as small as possible for large datasets to prevent overfitting.

"**-top_k**" determines the last few layers of the model that will be selected to compute the gradient for the virtual update. This parameter should be adjusted according to the depth of the model, and the larger it is, the better it is for a model with a deep structure. In RestNet-18, this parameter is set to 15, covering the last residual block.

"**-num_sifter**" controls how many sifters will be trained. Increasing this setting improves the filtering effect but adds to the time/memory overhead. Starting from five and gradually growing, this parameter is recommended.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20220926. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2023/.

References

[1] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The german traffic sign recognition benchmark: a multi-class classification competition," in *The 2011 international joint conference on neural networks*. IEEE, 2011, pp. 1453–1460.

[2] Y. Zeng, S. Chen, W. Park, Z. Mao, M. Jin, and R. Jia, "Adversarial unlearning of backdoors via implicit hypergradient," in *ICLR*, 2022.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[4] Y. Zeng, M. Pan, H. A. Just, L. Lyu, M. Qiu, and R. Jia, "Narcissus: A practical clean-label backdoor attack with limited information," *arXiv:2204.05255*, 2022.

[5] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47 230–47 244, 2019.