USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Revisiting Training-Inference Trigger Intensity in Backdoor Attacks

Chenhao Lin, Chenyang Zhao, Shiwei Wang, Longtian Wang, Chao Shen, and Zhengyu Zhao, *Xi'an Jiaotong University*

https://www.usenix.org/conference/usenixsecurity25/presentation/lin-chenhao

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 34th USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 34th USENIX Security Symposium.

August 13–15, 2025 • Seattle, WA, USA

978-1-939133-52-6

Open access to the Artifact Appendices to the Proceedings of the 34th USENIX Security Symposium is sponsored by USENIX.

USENIX Security '25 Artifact Appendix: Revisiting Training-Inference Trigger Intensity in Backdoor Attacks

Chenhao Lin, Chenyang Zhao, Shiwei Wang, Longtian Wang, Chao Shen, Zhengyu Zhao*
Xi'an Jiaotong University

A Artifact Appendix

A.1 Abstract

In our artifact, we provide the source code along with a dataset archive. Specifically, the codebase consists of implemented algorithms for backdoor attacks and defenses,shell scripts that facilitate the TITIMworkflow, and analysis scripts to reproduce experimental results (*e.g.*, the heatmaps presented in Figure 6). The dataset archive includes four clean datasets mentioned in our paper, with clearly divided training, test, and validation splits, as well as their associated metadata (*e.g.*, backdoor configuration).

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

We admit that the proposed attack strategy under train-inference intensity mismatched triggers can be used for malicious purposes, however, it is unlikely to cause severe harm to individuals or society, since selecting the optimal trigger for each setting would require exponentially more resources than training. Moreover, the phenomenon of train-inference intensity mismatch can help form a foundation for developing more robust defenses against backdoor attacks, which can be beneficial to the machine learning community in the long term.

A.2.2 How to access

The source code and datasets are available on GitHub (https://github.com/cv12ha0/TITIM) and Zenodo (https://zenodo.org/records/14729436).

A.2.3 Hardware dependencies

Experiment with ResNet18 on CIFAR-10 require at least 8GB of RAM and a 2GB NVIDIA GPU. For other models and datasets, we recommend a machine with Linux operating system with 128GB of RAM and a 24GB NVIDIA GPU. In our experiment setup, we use a server running Ubuntu 22.04.4 with 2× Intel Xeon Gold 6226R CPUs, 256GB of RAM, and 4× NVIDIA GeForce RTX 3090 GPUs.

*Corresponding Author

A.2.4 Software dependencies

Our experiments are primarily conducted using Python 3.8, PyTorch 1.12.1, and CUDA 11.3. Additional experiments confirmed compatibility with Python 3.9, PyTorch 2.5.1, and CUDA 12.2.

The Instagram filters used in the backdoor attack Styled require MagickWand, which can be installed by running the following command:

```
$ sudo apt-get install libmagickwand-dev
```

Please refer to the instructions in the *README.md* file in our repository to set up the environment.

A.2.5 Benchmarks

To reproduce the main results in our paper, we require four datasets: MNIST, CIFAR-10, GTSRB, and CelebA. You can either run the shell scripts to prepare the datasets automatically or manually download the preprocessed clean datasets from Zenodo (https://zenodo.org/records/14729436).

A.3 Set-up

A.3.1 Installation

Please follow these steps to install the software dependencies:

CUDA Installation: Please make sure a CUDA toolkit with a version later than 11.3 is properly installed on your machine. You can check the official website of CUDA Toolkit for instructions: https://developer.nvidia.com/

Anaconda Installation: You can use venv or Anaconda to manage your Python environments; however, we recommend Anaconda for its ease of use. Please check the official website of Anaconda for more details: https://www.anaconda.com/download

Environment Creation: You can either create the running environment based on the *requirements.txt* or manually run commands in *scripts/env.sh* which contains exactly the same commands we used to set up our environment. Please refer to the *README.md* for details.

A.3.2 Environment Test

Run the following commands in shell to validate the environment:

```
$ python
>>> import torch
>>> torch.cuda.is_available()
```

The dependencies are installed properly if the output is "True".

A.3.3 Dataset Preparation

Activate the Conda environment and run commands in *scripts/get_clean_datasets.sh* to prepare the clean datasets automatically. You can also manually download the preprocessed clean datasets from Zenodo (https://zenodo.org/records/14729436) if you encounter network issues while running the scripts.

If downloaded properly, there should be a directory named "data/", which contains a clean subset for each specified dataset.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): Models implanted with low training intensity triggers can generalize to higher-intensity inference triggers, while those with high training intensity triggers tend to overfit to higher-intensity inference triggers. This is proven by the experiments(E2) in Section 4.2 and Appendix D.

(C2): Attackers can improve the ASR by mixing triggers with varying intensities during data poisoning. For instance, one can improve the worst-case from approximately 10% (random guess) to at least 66.98% and exceeding 90% (92.77%) in extreme cases. This is proven by the experiments(E3) in Section 4.3 and Appendix C.

A.4.2 Experiments

(E1): [TITIM workflow] [5 human-minutes + 4 compute-hours + 6GB disk]: Run TITIM workflow to evaluate a single configuration (an attack with a specified intensity) and generate a heatmap as in Section 4.2.

Preparation: Download the clean datasets, create and activate the Conda environment as mentioned in Section A.3.

Execution: Run the full workflow in one script:

```
$ sh scripts/badnets_square.sh
```

Alternatively, you can run the scripts step by step.

Generate poisoned CIFAR-10 datasets by BadNets (Square):

```
$ sh scripts/inject/inject_badnets_square.sh
```

Train backdoored models:

```
$ sh scripts/train/train_badnets_square.sh 0
  cifar10 resnet18 0.05
```

Test backdoored models with varying intensities:

```
$ sh scripts/inference/crosstest_badnets\
  _square.sh 0 cifar10 resnet18
```

Draw a heatmap from the results:

```
$ python utils/scripts/tsv_reader.py
```

Scripts for other attacks are stored in the *scripts* folder, you can run them according to your specified attacks.

Results: The poisoned datasets are saved in "data/"; the backdoored models are stored in "res/"; the training and inference logs are saved as CSV files in "logs/"; the heatmap are generated and saved as PDF files in "heatmaps/". The trends in the heatmap should reflect the four regions described in Section 4.2 of the main text.

(E2): [Intensity mixture] [5 human-minutes + 7 compute-hour + 5GB disk]: Train backdoored models with two intensities by BadNets(Square) and generate a heatmap as in Figure 21.

Preparation: Download the clean datasets, create and activate the Conda environment.

Execution: Run the full workflow in one script:

```
$ sh scripts/badnets_square_mix.sh
```

Alternatively, you can run the scripts step by step.

Generate poisoned datasets by BadNets (Square) with two opacities:

```
$ sh scripts/mixtest/inject_mix.sh
```

Train backdoored models:

```
$ sh scripts/mixtest/train_mix.sh
```

Inference with varying intensities:

```
$ sh scripts/mixtest/crosstest_mix.sh
```

Draw the heatmap:

```
$ sh scripts/mixtest/draw_mix.sh
```

Results: The poisoned datasets are saved in "data/"; the backdoored models are stored in "res/"; the training and inference logs are saved as CSV files in "logs/"; the heatmap are generated and saved as PDF files in "heatmaps/".

A.5 Notes on Reusability

The modular approach of this artifact ensures compatibility and facilitates collaboration across different projects and implementations. For instance, as long as the dataset is packaged as a DataFrame with PNG files and labels, along with the model and backdoor algorithms adhering to the defined interface specifications, our work can be easily reused. Please refer to the repository for more details.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2025/.