

USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Low-Cost Hard-Label Adversarial Attack with Theoretical Foundations

Jun Liu, *University of Macau; National Institute of Informatics;*
Leo Yu Zhang, *Griffith University;* Fengpeng Li, *University of Macau;*
Isao Echizen, *University of Tokyo; National Institute of Informatics;*
Jiantao Zhou, *University of Macau*

https://www.usenix.org/conference/usenixsecurity26/presentation/liu-jun

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 35th USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 35th USENIX Security Symposium.

August 12–14, 2026 • Baltimore, MD, USA

ISBN 978-1-939133-58-8

Open access to the Artifact Appendices to the Proceedings of the 35th USENIX Security Symposium is sponsored by



جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology



USENIX Security '26 Artifact Appendix: Low-Cost Hard-Label Adversarial Attack with Theoretical Foundations

Jun Liu[†‡], Leo Yu Zhang[§], Fengpeng Li[†], Isao Echizen[¶‡], Jiantao Zhou[†*]

[†]University of Macau, [‡]National Institute of Informatics, [§]Griffith University, [¶]University of Tokyo
[*]Corresponding author

A Artifact Appendix

A.1 Abstract

This artifact contains the code, scripts, checkpoints, configuration files, and documentation for reproducing the main results of *Low-Cost Hard-Label Adversarial Attack with Theoretical Foundations*. It implements DPAttack, a query-efficient untargeted hard-label black-box adversarial attack with a zero-query frequency-based initialization and a Pattern-Driven Optimization (PDO) module. The artifact supports experiments on CIFAR-10, ImageNet, ObjectNet/CLIP, ImageNet-C, PathMNIST, adversarially trained models, real-world commercial APIs, Blacklight, randomized defenses, and dense prediction tasks. It also includes a lightweight CIFAR-10 Quick Start with Docker support, intended for rapid functional verification before running larger experiments.

This artifact evaluation appendix focuses on the experimental pipeline for the major evaluation settings, including CIFAR-10, ImageNet, ObjectNet/CLIP, adversarially trained models, and the Blacklight defense, to facilitate efficient artifact assessment.

A.2 Description & Requirements

The full artifact provides: (1) source code for the proposed DPAttack method, including fixed-block-size, dynamic-block-size, ℓ_2, Blacklight-aware, and randomized-defense variants; (2) scripts and wrappers for reproducing the main experiments and baselines; (3) analysis utilities for BFS, frequency statistics, gradient-alignment, cosine-similarity evolution, and curvature estimation; (4) included or referenced checkpoints and auxiliary files; and (5) README files describing dataset preparation, expected directory structure, hardware/runtime requirements, Quick Start usage, and commands for reproducing the paper's main tables and figures.

A.2.1 Security, privacy, and ethical concerns

This artifact contains adversarial-attack code and therefore could potentially be misused to study or evade model robustness in real-world systems. However, it does not include destructive functionality, malware, privilege escalation, persistence, or denial-of-service components. The code is released for robustness evaluation and defensive research only. Some experiments target commercial APIs; those require the evaluator's own credentials and should be conducted responsibly, with conservative query budgets and in compliance with the corresponding service policies.

A.2.2 How to access

The artifact is publicly available via the permanent Zenodo DOI https://doi.org/10.5281/zenodo.20322560, as provided in the Open Science section of the paper. It consists of three compressed packages: `DPAttack.tar.gz`, which contains the full artifact, including the code for E1–E3; `CIFAR10_QuickStart.tar.gz`, which provides a self-contained quick-start package with a Dockerfile; and `TestImages.tar.gz`, which includes 500 test images each from ObjectNet and ImageNet.

A.2.3 Hardware dependencies

Our experiments were primarily conducted on a single NVIDIA RTX A6000 GPU with 48GB VRAM. This is the hardware used in our experiments, not a strict minimum requirement.

Table 1: Observed hardware usage for representative settings.

Setting	Observed Usage
CIFAR-10 Quick Start	$<$ 1GB GPU VRAM, $<$ 1GB RAM
ImageNet classifiers	$<$ 2GB GPU VRAM, around 2GB RAM
CLIP-based classification	Up to around 14GB GPU VRAM, around 2GB RAM

For example, in our setup, ConvNeXt-Base used approximately 1.9GB system RAM. Among the full experiments, CLIP-based attacks had the largest GPU memory footprint. Running the full suite of experiments and baselines may take several days, depending on hardware, query budgets, dataset

availability, API rate limits, and whether all baselines are rerun. We therefore recommend starting from the Quick Start and then reproducing representative experiments.

A.2.4 Software dependencies

We use Ubuntu with Conda for environment management, and recommend Python 3.11. Unless otherwise specified in the corresponding sections, the test experiments described below can be executed directly in the environment provided by the Quick Start image. For the complete set of experiments reported in the paper, we provide the full pinned Conda environment file: `tools/py311_20260517.yaml`.

For the Quick Start setup, we additionally provide `Dockerfile.quickstart`, `requirements_quickstart.txt`, and `requirements_quickstart_docker.txt`. Depending on the local Docker configuration, Docker commands may require `sudo`. Some experiments further require external public datasets, pretrained checkpoints, third-party repositories, or commercial API credentials; these requirements are documented in the repository README or explicitly noted in the corresponding experiment sections.

A.2.5 Benchmarks

The artifact evaluation supports experiments on CIFAR-10, ImageNet-1K and ObjectNet/CLIP. We provide small subsets of the public datasets for efficient evaluation, while the full large-scale public datasets and some third-party repositories are not redistributed directly. The following sections and README explain how to obtain these resources and place them in the expected directory structure.

A.3 Set-up

A.3.1 Installation

Quick start.

1. Download the artifact `CIFAR10_QuickStart.tar.gz`.
2. Extract this package:
   ```
   tar -xzf CIFAR10_QuickStart.tar.gz
   cd CIFAR10_QuickStart
   ```
3. Build the Docker image:
   ```
   docker build -f Dockerfile.quickstart \
   -t dpattack-cifar10-quickstart:latest .
   ```

If Docker is not available in your environment, please follow the instructions in `CIFAR10_QuickStart/README_quickstart.md` to install the required dependencies with `pip`.

Full artifact. The following steps 2-3 are optional for artifact evaluation. The Conda environment built in the Quick Start above can be reused for the following evaluation experiments, requiring only a small number of additional package installations when necessary.

1. Extract the full artifact[1] `DPAttack.tar.gz`.
2. Create the Conda environment:
   ```
   conda env create -f tools/py311_20260517.yaml
   conda activate py311
   ```
3. Prepare datasets / checkpoints / third-party repositories according to the README.

A.3.2 Basic Test

The basic functionality test is the CIFAR-10 Quick Start. It is designed to finish in about 5 minutes and serves to verify that the software environment, checkpoint loading, and attack pipeline are functioning correctly.

Docker command:

```
docker run --rm --gpus all dpattack-cifar10- \
quickstart:latest
python OursCifar10DBS.py --victimmodel vgg
```

If the system uses the NVIDIA runtime in CDI mode, use:

```
docker run --rm --runtime=nvidia \
-e NVIDIA_VISIBLE_DEVICES=all \
-e NVIDIA_DRIVER_CAPABILITIES=compute,utility \
dpattack-cifar10-quickstart:latest
python OursCifar10DBS.py --victimmodel vgg
```

Expected result: The script should successfully load the CIFAR-10 victim model VGG-16-BN, attack 200 test samples, and produce summary statistics consistent with those reported in Table 10 of the manuscript, including an Attack Success Rate (ASR) of approximately 30% under a query budget of 50. It reports the ASR and query statistics (Avg.Q and Median.Q). By default, the result log is saved under `CIFAR10_QuickStart/results/` with an automatically generated file name. Users can set `--results_format` to either `json` or `csv`, and use `--results_log` to specify a custom output path.

A.4 Evaluation workflow

A.4.1 Major Claims

This artifact is designed to reproduce the following major claims of the paper:

(C1): *On standard classifiers, especially ImageNet, DPAttack achieves higher ASR and better query efficiency than prior hard-label attacks under realistic query budgets. This is supported by Sec.5.2 and Table 3.*

(C2): *DPAttack generalizes beyond standard ImageNet models to CLIP/ObjectNet. This is supported by Secs.5.3 and Tables 4.*

(C3): *DPAttack remains effective under Blacklight. This is supported by Sec.5.4 and Table 7.*

[1] Please note that `CIFAR10_QuickStart` contain utility files whose names overlap with those in the full `DPAttack` artifact. To avoid ambiguity, please run the corresponding commands within their respective workspaces when invoking these overlapping files.

A.4.2 Experiments

The results reported below are expected to be close to those in the paper. Small numerical differences may still arise from hardware/software differences, such as CUDA/cuDNN versions, dependency versions, and GPU computation details.

(E1): *Representative ImageNet experiments for C1 [20 human-minutes + 9 compute-minutes].*

How to: Reproduce a representative attack experiment on ImageNet dataset against ResNet-50 with a fixed query budget of 50, as reported in Table 3.

Preparation: Activate the Conda environment built by the Dockerfile in the Quick Start. Extract `TestImages.tar.gz` and move the extracted `val` directory to `DPAttack/data/imagenet/`.

Execution: Run the command `python OursDBS.py`.

Results: The expected trend is that `Ours_dyn` achieves nearly the same ASR ($\approx 55\%$) and query efficiency as reported in Table 3. After execution, the key summary metrics, including `ASR`, `Avg.Q`, and `Median.Q`, will be saved under `DPAttack/results/`.

Note: The results for `Ours_opt` and the other baselines in Table 3 can also be reproduced using:

```
python Ours.py
python baselines/ADBA.py
python baselines/attack_imagenet_others.py \
--attack_method rays
```

These additional experiments require approximately 6 compute-minutes, 2 compute-minutes, and 1/2/2/2/2 compute-minutes for HRayS/TtBA/BouT/BouR/HSJA.

(E2): *Representative ObjectNet experiments for C2 [20 human-minutes + 1 compute-hour].*

How to: Reproduce attack experiments on ObjectNet dataset against the CLIP model with a fixed query budget of 50, as reported by Table 4.

Preparation: Activate the Conda environment built by the Dockerfile in the Quick Start. Install the CLIP model dependencies using:

```
pip install open_clip_torch transformers
```

Extract `TestImages.tar.gz` and move the extracted `objectnet_sample` directory to `DPAttack/data/objectnet/`.

Execution: Run the command

```
python OursDBS.py --victimmodel clip \
--datasource objectnet
```

to perform `Ours_dyn` on ObjectNet dataset.

Results: The expected trend on ObjectNet/CLIP is that `Ours_dyn` outperforms ADBA and the other baselines in terms of ASR ($\approx 39.5\%$) and query efficiency under realistic query budgets, consistent with Table 4. After execution, the key summary metrics, including `ASR`, `Avg.Q`, and `Median.Q`, will be saved under `DPAttack/results/`. This experiment also illustrates the most memory-demanding classification setting in the artifact.

Note: The results for `Ours_opt` and the other baselines on ObjectNet/CLIP in Table 4 can also be reproduced using the following scripts:

```
python Ours.py --victimmodel clip \
--datasource objectnet --blocksize 4
python baselines/ADBAObjectNetClip.py
python baselines/attack_objectnet_others.py\
--attack_method rays
```

These additional experiments require approximately 30 compute-minutes, 30 compute-minutes, and 15/30/15/15/15 compute-minutes for HRayS/TtBA/BouT/BouR/HSJA, respectively.

(E3): *Representative Blacklight experiments for C3 [20 human-minutes + 2 compute-hours].*

How to: Reproduce the ImageNet attack experiments against ViT-B-32 under the stateful detection defense Blacklight, with a fixed query budget of 100, as reported in Table 7.

Preparation: Activate the Conda environment built by the Dockerfile in the Quick Start.

Execution: Run `python OursBlacklight.py` to perform `Ours_dyn` against ViT-B-32 on ImageNet.

Results: The expected trend is that `Ours_dyn` remains effective under Blacklight and achieves stronger attack and evasion performance than competing methods, consistent with Table 7. Specifically, DPAttack is expected to achieve a 0% Blacklight detection rate and an ASR of nearly 43%. In this setting, once an input sample is detected by the defense, the attack on that sample is terminated and counted as a failure. After execution, the attack metrics together with Blacklight-related defense statistics, including the numbers of successful and failed detections, the Blacklight detection ratio, and the average and median query indices at which Blacklight first detects an attack, will be saved under `DPAttack/results/`.

Note: The results for `Ours_opt` and the other baselines in Table 7 can also be reproduced using the following scripts:

```
python OursBlacklightOpt.py --blocksize 4
python baselines/ADBA.py --victimmodel vit \
--budget 100 --defense 1
python baselines/attack_imagenet_others.py \
--attack_method rays --defense 1 \
--targetn vit --maxquery 100
```

These additional experiments require approximately 2 compute-hours, 2.5 compute-hours, and 2/5/5/5/2 compute-minutes for HRayS/TtBA/BouT/BouR/HSJA, respectively.

A.5 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/usenixsec2026/.