
# USENIX Security '25 Artifact Appendix: Atkscopes: Multiresolution Adversarial Perturbation as a Unified Attack on Perceptual Hashing and Beyond

Yushu Zhang[1], Yuanyuan Sun[1], Shuren Qi[1,*], Zhongyun Hua[2], Wenying Wen[3], Yuming Fang[3]

[1]Nanjing University of Aeronautics and Astronautics
[2]Harbin Institute of Technology, Shenzhen
[3]Jiangxi University of Finance and Economics
[*]Corresponding author: Shuren Qi, Email: shurenqi@nuaa.edu.cn

## A Artifact Appendix

### A.1 Abstract

For USENIX Security '25 Artifact Evaluation, we provide the code, models, datasets, and execution scripts of our paper. In addition, we provide a detailed description in this document on how to set up the environment and reproduce the main statements of our paper.

### A.2 Description & Requirements

The attack against PhotoDNA has been tested on a 64-bit Windows machine. This limitation is due to the PhotoDNA binary, which is architecture-specific. The attacks on phash, PDQ, and NeuralHash have been tested on Linux. It is important to note that the attack on NeuralHash requires a CUDA-supported GPU.

To reproduce our major results, you can run our attack as described in the README.md.

#### A.2.1 Security, privacy, and ethical concerns

This artifact is used to perform adversarial attacks on perceptual hashing algorithms. Although these attacks are conducted in a controlled and compliant manner in the paper, evaluators should avoid applying these attacks to real-world systems or platforms, especially without authorization, as this may violate terms of service or cause unintended damage to commercial or personal systems.

#### A.2.2 How to access

Please download the artifact zip file from Zenodo . After extracting the file, please open the two subdirectories below as separate projects in PyCharm, as these projects need to be built in different environments.

#### A.2.3 Hardware dependencies

The only dependency is the need for a machine compatible with the corresponding .dll or .so file. This should work in a 64-bit Windows environment.

#### A.2.4 Software dependencies

The main dependencies are Docker, TensorFlow, and PyTorch. Other dependencies are listed in the requirements.txt.

#### A.2.5 Benchmarks

In the artifact, we provide executable Python files for escaping and triggering regulation attacks on pHash, PDQ, PhotoDNA, and NeuralHash. We evaluate our two attack scenarios using the ImageNet dataset from the ILSVRC 2012 challenge. For our experiments, we have randomly selected 50 pairs of images from the ImageNet dataset, which are available in the directories `./imagenet_50_resized/` and `./imagenet_tar_50_resized/`. Additionally, we have included the necessary models for the experiments in the artifact. The model for computing visual loss can be found in the `./lpips/` directory. The models for pHash, PDQ, PhotoDNA, and NeuralHash used in the attacks are located in the following files: `./imagehash.py`, `./python/pdqhashing`, `./PhotoDNAx64.dll`, and `./NeuralHash` respectively.

### A.3 Set-up

#### A.3.1 Installation

To conduct attacks on shallow hashes (pHash, PDQ, PhotoDNA) and deephash (NeuralHash), two different environments need to be configured. Below, we will first explain how to configure the environment to run the Python files under `AtkScope_ShallowHash`. It is important to note that the PhotoDNA model can only run on Windows systems. Therefore,

if you wish to conduct attacks on PhotoDNA, you will need to set up the environment on a Windows system.

First, you need to download the Anaconda installation package from the official website. Once the installation is complete, open the terminal and enter the following commands:

```
$ conda create -n <your_env_name> python=3.6

$ conda activate <your_env_name>

$ pip install -r requirements.txt
```

For running files under `AtkScope_NeuralHash`, we recommend configuring the environment on a Linux system. Please run the following command from the project's root

```
$ sudo docker build -t hashing_attacks --build-arg
     USER_ID=$(id -u) --build-arg GROUP_ID=$(id -g
    ) .

$ sudo docker build -t hashing_attacks -f rootless
    .Dockerfile .
```

### A.3.2 Basic Test

Run the following command in the project root directory to test if the shallow hashing environment is working properly.

```
$ conda activate <your_env_name>

$ python test_attack_rgb_target_PDQ.py  --
    untargeted -a black -d imagenet -c 10 -o 10000
     -m 10000  --reset_adam -n 50 --solver adam -b
     1 -p 1 --hash 92 --use_resize --htype "PDQ"
    --init_size 32 --init_dct_size 16 --method "
    tanh" --modifier_method "multiply" --batch 16
    --gpu 1 --lr 1 -s "
    RGB_results_imagenet_targetPDQ_lpips_dct8_lr0
    .1_c10_dist92" --start_idx=0 --dist_metrics "
    pdist" --save_ckpts "
    best_modifier_imagenet_target_PDQ"
```

The expected correct output is shown in Figure 1.

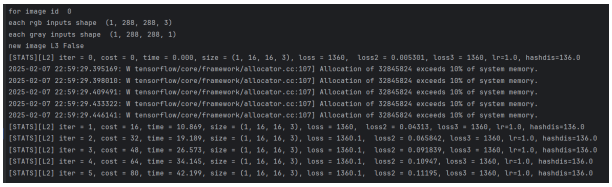


Figure 1: Expected output of testing the shallow hashing environment.

Run the following command in the project root directory to test if the deep hashing environment is working properly.
To start the docker container run the following command from the project's root:

```
$ sudo docker run --rm --shm-size 16G --name
    my_hashing_attacks --gpus '"device=0"' -v $(
    pwd):/code -it hashing_attacks bash
```

To run the triggering regulation attack against NeuralHash, enter the following command in the project root :

```
$ python adv1_target_attack.py --source=
    imagenet_50_resized --target_hashset=
    dataset_hashes/imagenet_tar_50_resized_hashes.
    csv --output_folder '
    output_target_imagenet_l2_dist17_each10_lr0.01
    '
```

The expected correct output is shown in Figure 2.

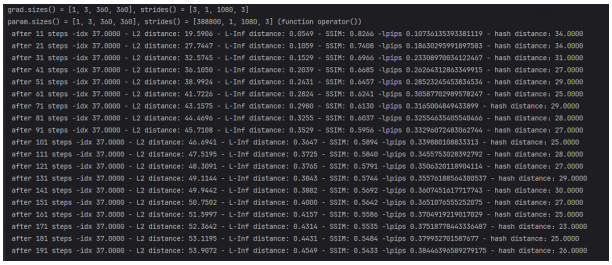


Figure 2: Expected output of testing the deep hashing environment.

## A.4 Evaluation workflow

We will include the operational steps and experiments that must be performed to evaluate if our artifact is functional and to validate our paper's key results and claims in this section.

### A.4.1 Major Claims

**(C1):** Atkscopes achieve highly efficient and effective adversarial attacks on four commercial hashing algorithms—pHash, PDQ, PhotoDNA, and NeuralHash, in both the escaping regulation and triggering regulation attack scenarios. This is proven by the experiments (E1) and described in sections 5.3, with results reported in tables 2 and 3.

### A.4.2 Experiments

**(E1):** [Escaping and Triggering Regulation Attacks] [10 human-minutes + 32 compute-hour + 20GB disk]:
**Preparation:** Follow the configuration instructions for the two environments as described in section A.3. For the PhotoDNA attack experiment, the setup must be done on a Windows system. All instructions can be found in the README file, and we recommend copying the commands directly from the README to avoid formatting errors.

**Execution:** Open the README file, copy the attack commands into the terminal in the project root directory to run the Python scripts.
**Results:** For the Python files under `AtkScope_ShallowHash`, after the program finishes running, the terminal prints `success_rate`, which corresponds to the Success Rate in the table, `overall average l2`, which corresponds to the $L_2$ distance in the table, `overall average perceptual distance`, which corresponds to the LPIPS in the table, `overall average iterations`, which corresponds to the Rounds in the table, and `overall average hash_ori` and `overall average hash_tar`, which correspond to the HashDistance in the table.
For the Python files under `AtkScope_NeuralHash`, after the program finishes running, the terminal prints `Success rate`, which corresponds to the *Success Rate* in the table, `Average L2 distance`, which corresponds to the $L_2$ *Distance* in the table, `Average LPIPS distance`, which corresponds to the *LPIPS Distance* in the table, `Average iterations`, which corresponds to the *Rounds* in the table, and `Average target loss`, which corresponds to the *HashDistance* in the table.

## 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/usenixsec2025/.