A Artifact Appendix

A.1 Abstract

This artifact contains the code for paper "Adversarial Detection Avoidance Attacks: Evaluating the robustness of perceptual hashing-based client-side scanning". We provide instructions for reproducing the results and running the attack on any other dataset of images. To reproduce the results in a timely fashion, we recommend using a Linux machine with at least 30 cores and 64G of RAM. A smaller machine will work but will require more time to run the experiments. The code is written to be modular such that it can be extended to run the attack on any other dataset. Furthermore, the experiments can be modified easily by changing a few parameters in the provided configuration files. This allows, for instance, to run the attack on a small number of images.

A.2 Artifact check-list (meta-information)

- **Algorithm:** The code for the attack algorithm is provided.
- **Data set:** ImageNet, the downloading and setting up instructions are provided.
- **Run-time environment:** Python
- **Metrics:** All metrics are provided in the code.
- **Output:** NPZ and PKL files containing the intermediate results. The final outputs are plots in PDF format.
- **Experiments:** All experiments are part of the code, and configuration files can be tuned to run them on a small scale.
- **How much disk space required (approximately)?:** 1TB.
- **How much time is needed to prepare workflow (approximately)?:** 2 hours, includes the setting up of the Python environment and the downloading of the dataset.
- **How much time is needed to complete experiments (approximately)?:** 2 weeks.
- **Publicly available?:** No.

A.3 Description

A.3.1 How to access

The artifact will only be provided by the authors to the reviewers. We will not release it publicly for ethical concerns and sensitivity of the topic.

A.3.2 Hardware dependencies

We recommend using a Linux machine with at least 30 cores and 64G of RAM to be able to run the attacks and reproduce the results. A smaller non-Windows machine will also work but with significantly longer time required (e.g. 2 weeks) to run all the large-scale experiments included in the paper. Small-scale experiments (i.e., attacking a few images) can be run in a shorter time.

A.3.3 Software dependencies

We use Miniconda [1] to setup the Python environment. The libraries and instructions to setup the environment are provided in the README. The code is tested on linux machines (specifically Ubuntu 16.04 and Ubuntu 20.04). But we expect the code should run on any other OS without any change as no OS dependent code is used to the best of our knowledge.

A.3.4 Datasets

We use ImageNet for all the experiments. For the updated results with duplicates removed from the ImageNet dataset, we refer the reader to our extended arXiv version[2].

A.4 Installation

Installation is only required to setup the conda environment for Python. We provide all the instructions in the README along with environment yaml file.

A.5 Experiment workflow

The workflow involves

1. Setup the Python environment;
2. Downloading the ImageNet dataset;
3. Running the Python code (preferably in something like tmux) to compute the image hashes;
4. Running the experiments to generate pickle files with results;
5. Running a Jupyter notebook to generate plots from the results.

A.6 Evaluation and expected results

To reproduce the results we need to run the experiments with the provided configuration files. This in turn would generate pickle files with the results of each experiment. Then, the notebooks are used to generate plots from the paper and they also print the reported values. One can expect to reproduce all the results from the paper. For reproducibility We also provide the seed used for all the experiments. All the results and plots from our paper can be generated using this evaluation.

A.7 Experiment customization

The configuration files can be modified to reconfigure the experiments, e.g. running the attack on fewer images. Similarly, the modularity in the code enables us to extend the experiments to other hashing algorithms and datasets by simply inheriting the generic classes provided for each module.

[1] https://docs.conda.io/en/latest/miniconda.html
[2] https://arxiv.org/pdf/2106.09820.pdf