PatchCleanser: Certifiably Robust Defense against Adversarial Patches for Any Image Classifier

Chong Xiang, Saeed Mahloujifar, and Prateek Mittal, *Princeton University*

https://www.usenix.org/conference/usenixsecurity22/presentation/xiang

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 31st USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 31st USENIX Security Symposium.

August 10–12, 2022 • Boston, MA, USA

978-1-939133-31-1

Open access to the Artifact Appendices to the Proceedings of the 31st USENIX Security Symposium is sponsored by USENIX.

A Artifact Appendix

A.1 Abstract

This artifact is based on PyTorch and requires GPU support. We implemented Algorithm 1 (double-masking algorithm) and Algorithm 2 (robustness certification procedure) of our PatchCleanser paper. The artifact can reproduce all experimental results (*clean accuracy and certified robust accuracy*) reported in the main body of the paper.

Our source code is available at https://github.com/inspire-group/PatchCleanser/tree/2370c78da15ccec08b7a05145c92cafb9b0f73a9. We further provide a detailed guide for evaluating our artifact at https://github.com/inspire-group/PatchCleanser/blob/2370c78da15ccec08b7a05145c92cafb9b0f73a9/misc/reproducibility.md.

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

- **Algorithm:** We implement Algorithm 1 (double-masking algorithm) and Algorithm 2 (robustness certification procedure) of our paper.
- **Program:** N/A.
- **Compilation:** N/A.
- **Transformations:** N/A.
- **Binary:** N/A.
- **Model:** ResNet, Vision Transformer, ResMLP. We provide downloading links to pretrained weights.
- **Data set:** ImageNet, ImageNette, CIFAR-10. They are publicly available benchmark datasets.
- **Run-time environment:** We test our artifact using anaconda virtual environment on Linux.
- **Hardware:** Requires one GPU.
- **Run-time state:** N/A.
- **Execution:** N/A.
- **Security, privacy, and ethical concerns:** N/A.
- **Metrics:** (top1) clean accuracy and (top1) certified robust accuracy. They are defined in the paper (at the end of Section 4.1).
- **Output:** We output results to console; they are numerical accuracy values.
- **Experiments:** We provide scripts for running experiments.
- **How much disk space required (approximately)?:** Datasets takes around 7GB. Each model weight take 100-300MB. The mid-product of the experiments takes less 300 MB.
- **How much time is needed to prepare workflow (approximately)?:** 1 hour.
- **How much time is needed to complete experiments (approximately)?:** Each experiment takes 3 hours to 2 days (from scratch).
- **Publicly available (explicitly provide evolving version reference)?:** https://github.com/inspire-group/PatchCleanser.
- **Code licenses (if publicly available)?:** MIT License.
- **Data licenses (if publicly available)?:** N/A.
- **Workflow frameworks used?:** N/A.
- **Archived (explicitly provide DOI or stable reference)?:** https://github.com/inspire-group/PatchCleanser/tree/2370c78da15ccec08b7a05145c92cafb9b0f73a9.

A.3 Description

A.3.1 How to access

We host our source code on GitHub at https://github.com/inspire-group/PatchCleanser.

Specifically, we use this commit for the artifact evaluation: https://github.com/inspire-group/PatchCleanser/tree/2370c78da15ccec08b7a05145c92cafb9b0f73a9.

A.3.2 Hardware dependencies

The artifact requires 2 CPU cores and 1 GPU.

A.3.3 Software dependencies

The artifact is based on Python, PyTorch, `timm`, and other Python packages. All packages can be easily installed with pip; we provide a list of required packages in `requirement.txt`.

A.3.4 Data sets

We use three publicly available datasets in our evaluation: ImageNet, ImageNette, CIFAR-10. See our reproducing instructions for more details.

A.3.5 Models

We use three representative image classifier models: ResNet, ResMLP, and Vision Transformer. We build models using `timm`; we provide download links to our pretrained weights. See our reproducing instructions for more details.

A.3.6 Security, privacy, and ethical concerns

N/A.

A.4 Installation

1. Install Python. [help link]
2. Install GPU-compatible PyTorch. [help link]
3. Install other Python dependencies. [help link]

4. Clone the source code from https://github.com/inspire-group/PatchCleanser/tree/2370c78da15ccec08b7a05145c92cafb9b0f73a9

5. Download datasets. [help link]

6. Download pretrained weights. [help link]

A.5 Experiment workflow

Our experiment is mostly based on the script `pc_certification.py`, in which we implemented our double-masking algorithm (Algorithm 1) and certification procedure (Algorithm 2). By running this script with proper command (we provide all necessary shell commands to run `pc_certification.py` in our reproducing instructions), we can read clean accuracy and certified robust accuracy from the console. We can compare the obtained results with the results reported in the paper to validate the reproducibility of our paper.

A.6 Evaluation and expected results

Our main claim is that our PatchCleanser defense achieves state-of-the-art defense performance, in terms of certified robust accuracy and clean accuracy, against adversarial patch attacks. This claim is supported by Table 2 of our paper. We can use commands listed in this section of our reproducing instructions to generate our key results, which should match the results reported in Table 2 of our paper.

In addition to our key results, our artifact also supports other experimental analyses presented in Section 4 and Section 5 of our paper. We provide detailed instructions and shell commands in our reproducing instructions.

Our algorithms are deterministic. Therefore, we do not expect any large variation in results. However, it is possible to have tiny mismatches ($< 0.1\%$) due to the imprecise float point computation on different hardware (e.g., GPUs).

A.7 Experiment customization

Our source code provides an easy way to customize the experiment.

First, we already support three datasets (ImageNet. ImageNette, CIFAR-10) in this artifact. In our GitHub repository, we further support three additional datasets (CIFAR-100, SVHN, Flowers-102). We can also add other datasets if their pretrained weights are available; we only need to register the new datasets in `utils/setup.py`.

Second, we can also support other image classification models (other than ResNet, ResMLP, ViT). We only need to have pretrained weights for these models and register them in `utils/setup.py`.

Third, we can easily change the parameters of our system. In our source code, we can change the underlying classification model using the flag `-model`, change the number of masks using `-num_mask`, change the mask stride using `-mask_stride`, change the estimated patch sizes using flags `-patch_size`, `-pa`, and `-pb`. We have already evaluated the effect of different parameters in our paper and this artifact.

A.8 Version

Based on the LaTeX template for Artifact Evaluation V20220119.