# USENIX Security '24 Artifact Appendix: Machine Learning needs Better Randomness Standards: *Randomised Smoothing* and `PRNG`-based attacks

Pranav Dahiya
University of Cambridge

Ilia Shumailov
University of Oxford

Ross Anderson
University of Cambridge
& University of Edinburgh

## A Artifact Appendix

*This artifact appendix is meant to be a self-contained document which describes a roadmap for the evaluation of your artifact. It should include a clear description of the hardware, software, and configuration requirements. In case your artifact aims to receive the functional or results reproduced badge, it should also include the major claims made by your paper and instructions on how to reproduce each claim through your artifact. Linking the claims of your paper to the artifact is a necessary step that ultimately allows artifact evaluators to reproduce your results.*

*Please fill all the mandatory sections, keeping their titles and organization but removing the current illustrative content, and remove the optional sections where those do not apply to your artifact.*

## A.1 Abstract

This artifact is a github repository, with modifications made to the existing `numpy`[1] repository. The changes made in `numpy` enable the bit-flipping `PRNG` attack described in section 4.3 in the paper, allowing the user to tune the parameters $\alpha$, $\beta$ and $\gamma$, thereby introducing skewness or kurtosis into the sampled distribution.

## A.2 Description & Requirements

The experiments presented in the paper were run using Python 3.11 on a laptop running Arch Linux with the following specifications:
**CPU:** AMD Ryzen 7 6800HS
**RAM:** 32 GB
**GPU:** Nvidia RTX 3050 Mobile (4GB VRAM)

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

It is recommended that the version of `numpy` be installed inside a virtual environment to ensure that other projects on the evaluators' system are unaffected. Our version of `numpy` does print a message to stdout to let the user know that it has been modified. By default, any sampled distributions remain unaffected, unless specific parameters are passed to numpy, but you may see a performance hit when sampling random numbers.

### A.2.2 How to access

Our fork of `numpy` can be found at `https://github.com/pranav-dahiya/numpy/tree/USENIX_final`. The repository contains a `README` on how to integrate this code with `smoothing`[2].

### A.2.3 Hardware dependencies

A GPU capable of running CUDA is required to run the scripts in `smoothing`. With the default parameters around 2.0 GB of VRAM is used during certification.

### A.2.4 Software dependencies

`numpy` requires a version of Python newer than 3.9.x. A C compiler that complies with C99 and a C++ compiler that complies with the C++17 standard is also required to build `numpy` from scratch.

### A.2.5 Benchmarks

The CIFAR10 image dataset is required to reproduce the results presented in the paper. This is automatically downloaded by PyTorch the first time any script in `smoothing` is executed. Pretrained models made available by Cohen et al. were used to run certification. These can be downloaded from the Google Drive link included in the ReadMe file of the `smoothing` repository.

[1]`https://github.com/numpy/numpy`

[2]`https://github.com/locuslab/smoothing/tree/master`

## A.3 Set-up

The only set-up required is to have a recent version of Python, newer than version 3.9.x installed.

### A.3.1 Installation

1. Clone both `numpy` and `smoothing` from GitHub using the links provided in Appendix A.2.2.
2. Create a virtual environment within the directory that `smoothing` was cloned into and activate it.
3. Install dependencies for `smoothing` (requirements.txt) and `numpy` (build-requirements.txt) using pip.
4. Navigate to the `numpy` directory and build the project using `python setup.py build`.
5. Install `numpy` using `python setup.py install`.

### A.3.2 Basic Test

A basic test can be performed by executing pdf.py in the `smoothing` directory. The generated figures should match figure 3 in the paper.

## A.4 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/usenixsec2024/`.