SoK: All You Need to Know About On-Device ML Model Extraction - The Gap Between Research and Practice

Tushar Nayan, Qiming Guo, and Mohammed Al Duniawi, *Florida International University;* Marcus Botacin, *Texas A&M University;* Selcuk Uluagac and Ruimin Sun, *Florida International University*

https://www.usenix.org/conference/usenixsecurity24/presentation/nayan

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 33rd USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE | ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL | ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

USENIX Security '24 Artifact Appendix: SoK: All You Need to Know About On-Device ML Model Extraction - The Gap Between Research and Practice

Tushar Nayan[*]
Florida Intl. University

Qiming Guo[*]
Florida Intl. University

Mohammed Al Duniawi
Florida Intl. University

Marcus Botacin
Texas A&M University

Selcuk Uluagac
Florida Intl. University

Ruimin Sun
Florida Intl. University

A Artifact Appendix

A.1 Abstract

The appendix will introduce the roadmap for evaluating the artifact and reproducing the major results discussed in the paper "SoK: All You Need to Know About On-Device ML Model Extraction - The Gap Between Research and Practice." Specifically, this appendix will explain how to use the source code available.

To facilitate the result, we evaluated the representative projects in laboratory settings and then further on large-scale ML applications. Our artifact comprises open-source projects, models, and evaluation strategies. It further includes script files with instructions for creating the environment for each open-source project, preparing the dataset, executing, and observing results claimed in our SoK paper.

A.2 Description & Requirements

This section includes the necessary information for recreating the experimental setups used in the paper, including how to access the source code, hardware and software requirements, and how to prepare the required datasets and model files for the evaluation. In most cases, it is recommended to run all experiments inside the virtual environment. Ideally, this environment should be set up on a Linux machine (e.g., Ubuntu) with GPU capabilities and administrative access.

A.2.1 Security, privacy, and ethical concerns

As far as we know, the artifact does not raise any security, privacy, or ethical concerns.

*The first two authors contributed equally to this work.

A.2.2 How to access

The artifact is available on GitHub[1]. Please start by cloning or downloading the repository on a commodity computer running on Linux. We have provided some sample model files in this repository that can be used for evaluation. If you require more real-world Machine Learning (ML) models, we can provide them upon request.

A.2.3 Hardware dependencies

We recommend evaluating our findings with a machine running Linux OS, equipped with at least 8GB RAM and a GPU that supports CUDA. We used a machine running Ubuntu 20.04.6 with Intel® Core™ i7-8700 and NVIDIA A100 GPU (CUDA 10.2) to get all the claims discussed in the paper. We further recommend that users must have administrative access, as for some projects, you may require sudo access to install and run the experiment.

A.2.4 Software dependencies

We recommend creating virtual environments for each open-source project using Anaconda. You should first check the `README.md` and `evaluate.md` file provided in the root directory to reproduce each project, we further recommend you to have a look at `README` files provided in the project's directory, where you may be asked to install some standard libraries such as PyTorch, NumPy, TensorFlow, etc. For most of the projects, you will find the `requirement.txt` or `environment.yml` files to quickly create the environments. Later, to evaluate the power consumption of each experiment you need to install Intel Performance Counter Monitor (PCM). The documentation for this tool is provided in the `Evaluation` folder.

[1] https://github.com/sys-ris3/ML_Extraction_Sok/tree/0d19edab5b5bd4bad4562543f4c1457be3c30852

A.2.5 Benchmarks

The experiments in our paper need ML models from different frameworks. Due to the large size of the models and potential intellectual property concerns, we can only provide access to these models upon request. Nonetheless, some projects use off-the-shelf datasets and models that can be automatically loaded by their script files. Detailed information is provided in each project's README file.

A.3 Set-up

This section will introduce how to set up the environment for running experiments quickly.

A.3.1 Installation

After cloning or downloading the repository, we need to have installed Python, Anaconda, and PCM as preliminaries for our evaluation. Then for each project, we need to create a separate virtual environment using Anaconda (conda) and install other necessary requirements. The steps are the following.

1. install PCM by following the steps in `Evaluation/README.md`.
2. install python with this command `sudo apt install python` and for conda check the documentation attached here.
3. for each project in folders `ModayXray`, `DeepSniffer`, `ML-Doctor`, `Prediction-Poison`, `Adaptive_misinformation`, repeat steps (4-7).
4. create a virtual environment with a short project name `conda create -name project_name python=v.0.0`, check the project's `README` file for the python version required.
5. activate the virtual environment and enter the project folder.
6. install requirements through `pip install -r requirements.txt` if file `requirements.txt` exists, or `conda env update -n project_name -file environment.yml.` if `environment.yml` exists.
   If you want to create virtual environments for all representative projects simply run the `env-setup.sh` file provided in the root directory to create the virtual environments followed by step 7.
7. follow the `README` steps for other installation requirements.

A.3.2 Basic Test

Use the following commands in the shell to check if Conda and Python is properly installed.

- `conda --version`
- `python --version`

If no error is reported in the final output, then all the dependencies are installed properly.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): The open-source projects that we selected for evaluation can work on sample models provided by authors.

(C2): The evaluated projects may not work on real-world ML models.

(C3): The power consumption of the evaluated projects is consistent with those reported in Section 4.4.2 of our paper.

A.4.2 Experiments

To conduct the model extraction attacks on a large scale with different model files, we recommend you refer to the README files provided in the project directory of - (project names)

(E1): EVALUATION - POWER CONSUMPTION [5 human-minutes + 5 compute-minutes]: Intel Performance Counter Monitor is used to estimate the power consumption before and after the attacks.

How to: Run the PCM tool in parallel to your model extraction attacks/defense projects to observe the power consumption in real time. You can find the documentation in the `Evaluation` folder.

Preparation: Steps 1 and 2 of the README file provided in the `Evaluation` folder will help set up the PCM tool and configure the specific environment variables required for execution.

Execution: Step 3 `bin/pcm` will run the PCM tool and display the PPO energy in real-time.

Results: The power consumption is the difference of the PPO energy before and after executing the attack/defense projects.

(E2): [Open-Source Projects] [3 human-hour + 5 compute-hour]: We have added all the open-source projects with the folder names `ModelXRay, ML-DOCTOR DeepSniffer, Adaptive Misinformation, Prediction-Poisoning` to our repository for large-scale evaluation on ML model extraction attacks/defenses.

How to: We highly recommend following the README file within each project directory for instructions on setting up the environment.

Preparation: We recommend to start testing with our model files provided in the folder `Models`. However, for some projects, e.g., `DeepSniffer`, model checkpoints are available here. Similarly, for `ML-DOCTOR`, one can use the model files in the `Models` folder. For `Adaptive Misinformation` and `Prediction-Poisoning`, the files are generated at execution. Later, one can replace the files with other model files. Last, please refer to the README file of each project for any additional preparation.

Execution: We recommend to first double check that you have created and activated the virtual environments corresponding to the projects you intend to test. Please follow the instructions provided in the README file of each project for execution. For most of the projects, we have provided a bash file that can be used as an alternative for evaluation. Additionally, for projects such as `DeepSniffer`, `ModelXRay`, `ML-DOCTOR`, `Adaptive Misinformation`, and `Prediction-Poisoning`, we recommend referring to the `orig_README` file for more detailed information regarding the datasets, models, and attack/defense strategies employed while evaluating. Finally, for projects `ModelXRay and ML-Doctor`, it is necessary to manually specify the path to your model file. This step may require some customization based on the user's specific model configurations.

We have also provided an option to quickly verify the execution for all projects using the script file - `env-setup.sh` (creates the environment) and `quick-test.sh` (run each project for a 20-second time frame) provided in the root directory of the repository. Once the virtual environment is created using the file `env-setup.sh`, you can verify the execution using two parallel terminals. In one terminal, we have a script file `quick-test.sh` that automates the execution of all projects, one by one, for claims 1 and 2. In the other terminal, PCM can run to monitor the power consumption of each experiment.

Results: The results for the projects can be observed while running the provided script files for evaluation of the experiments. Whereas, the results for projects such as Prediction Poisoning and Adaptive Misinformation can be found in subfolders of the models folder located within their respective project directories. These initial results produced by projects `DeepSniffer`, `ModelXRay`, `ML-Doctor`, `Adaptive Misinformation`, and `Prediction-Poisoning` will aid in the analysis of the findings presented in the original paper. Upon further testing with our real-world models, you will find the results and challenges that we have discussed in sections 4.2 and 4.3 of our original paper.

A.5 Notes on Reusability

The projects featured in our repository are specifically tailored for execution within a laboratory environment. We strongly advise readers to refer to the README file, as it serves as a comprehensive guide for applying model extraction attacks/defenses across various model files. Additionally, some attacks may not be practical or cannot be performed on a large scale, and the corresponding defense solutions are limited in deployment as well.

A.6 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/`.