
# USENIX Security '25 Artifact Appendix: CAMP in the Odyssey: Provably Robust Reinforcement Learning with Certified Radius Maximization

Derui Wang[♡,♠], Kristen Moore[♡,♠], Diksha Goel[♡,♠], Minjune Kim[♡,♠], Gang Li[♣], Yang Li[♣], Robin Doss[♣], Minhui Xue[♡,♠], Bo Li[◊], Seyit Camtepe[♡,♠], and Liming Zhu[♡]

[♡]*CSIRO's Data61, Australia*
[♠]*Cyber Security Cooperative Research Centre, Australia*
[♣]*Deakin University, Australia*
[◊]*University of Chicago, USA*

## A Artifact Appendix

### A.1 Abstract

This artifact appendix focuses on two main claims (please refer to "Contributions" in Section 1 and "Main takeaways" at the end of Section 5) in our paper:

- **`CAMP` and *policy imitation* enhance the certified robustness of DRL agents:** We compare `CAMP` with two other baseline training methods, namely Gaussian and NoisyNet. `CAMP` outperforms these baselines in the trade-off between certified agent utility (*i.e.,* "Certified Expected Return") and certified robustness (*i.e.,* "Certified Radius"). In particular, significant performance gains are observed in both classic control and autonomous driving environments.
- **Agents trained by `CAMP` and *policy imitation* are more robust against empirical attacks:** `CAMP` effectively mitigates adversarial perturbations in observations, maintaining a higher average return across diverse attack scenarios. The `CAMP` agents exhibit superior robustness compared to Gaussian agents, with particularly significant improvements observed in classic control and autonomous driving environments.

We have provided the source code and scripts in this artifact to reproduce the results that support these two claims.

### A.2 Description & Requirements

The artifact enables the demonstration and reproduction of results for both `CAMP`, Gaussian, and NoisyNet agents in the Cartpole environment. Since the original code and scripts were developed and tested on our internal high-performance computer (HPC) cluster (CSIRO Virga), which restricts access from external users, we have done our best to ensure the results can be reproduced on non-HPC devices. We provided a list of requirements for running our code on a customer PC to ensure that evaluators can conduct the evaluation.

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

To the best of our knowledge, there are no security, privacy, or ethical concerns associated with our artifact. Our code does not collect data or fingerprints from evaluators. Moreover, these concerns are not applicable if the evaluator uses their own hardware and software platform for evaluation.

#### A.2.2 How to access

The code is available from our GitHub repository (https://github.com/NeuralSec/camp-robust-rl). Evaluators can simply clone the repository to their local PC and set up a virtual environment as instructed. Since training the DQN can be highly unstable, the quality of agents trained with different GPUs or in different system environments may vary. To ensure reproducibility, we set up containers with NVIDIA RTX-4090 on `runpod.io` , allowing evaluators to log in anonymously and run the corresponding experiments. The Zenodo release of our code can be accessed from `https://zenodo.org/records/14729675`.

#### A.2.3 Hardware dependencies

We recommend that evaluators use a PC equipped with a CUDA-capable GPU. While our code and scripts were developed and tested on an internal HPC cluster with NVIDIA H100 GPUs, the code is not memory-intensive. Consumer-grade GPUs supporting CUDA should handle execution without issues. For reference, training with `CAMP` and policy imitation on the Cartpole environment typically consumes less than 1GB VRAM.

Table 1: Environment for Experiments

| Software | Detail |
|---|---|
| Operating system | Linux (SUSE Linux Enterprise Server SLE15) |
| Python version | 3.11.4 |
| CUDA version | 12.4 |
| cuDNN version | 9.1.0 |
| Major libraries | Pytorch 2.5.1<br>Gymnasium 0.29.1<br>Highway-env 1.9.1<br>ale-py 0.8.1<br>autoattack 0.1<br>SciPy 1.11.0<br>Numpy 1.26.4<br>statsmodels 0.14.2<br>matplotlib 3.8.4<br>tensorboard 2.16.2<br>tqdm 4.66.4<br>mlconfig 0.1.8 |

#### A.2.4 Software dependencies

The code should be executable on various versions of Linux operating systems. A `requirements.txt` file is provided for quick setup of the environment. This file includes a comprehensive list of dependencies, rather than just the minimal required ones, to facilitate the recreation of the virtual environment used during our experiments. Table 1 lists the system environment and major libraries used during our code testing.

#### A.2.5 Benchmarks

We mainly produce results based on the Gymnasium [3] Cartpole environment in this artifact. Specifically, this artifact uses `"Cartpole-v0"` from Gymnasium. The Q-network of the agent is a multi-layer perception (MLP) whose architecture is described in Table 4 of our paper.

### A.3 Set-up

- Clone the repository from GitHub to your local PC or the provided container.
- A `readme.md` file included in the repository describes the commands for environment setup.
- The experiment will create new folders within the current directory to store its results. Please ensure that the directory where you run the experiment has the necessary write permissions to allow the creation of these folders.
- We provide four shell scripts to help you reproduce the results.

#### A.3.1 Installation

The installation steps are described in the `readme.md` file. First, navigate to the root directory of the cloned repository and create a virtual environment using `venv` . Next, update `pip` to the latest version and install dependencies from `requirements.txt` . After installation, you can type `deactivate` to exit the virtual environment. To this end, the setup is finished and the code should be ready for the following evaluation. We have observed that when installing dependencies on some Ubuntu distributions (tested on WSL2 Ubuntu 24.04.1 LTS), running `pip install -r requirements.txt --no-cache-dir` may result in errors like segmentation fault due to conflicting dependency versions. If this occurs, rerunning the same command immediately after the error often resolves the issue.

#### A.3.2 Basic Test

In the root directory of the cloned repository, execute `bash ae_step1.sh` . If the Python version and GPU information are displayed without errors, the setup is successful. To monitor training with TensorBoard, open a new terminal, navigate to the root directory of the repository with the virtual environment activated, and run `tensorboard --logdir="exp"` . This will allow you to visualize the training progress.

### A.4 Evaluation workflow

This artifact includes four shell scripts located in the root directory, which generate results for two major experiments designed to verify our two claims. In this evaluation, we will focus on reproducing results in the Cartpole environment under the setting of $\sigma = 0.2$.

- `ae_step1.sh` trains agents using `CAMP`, Gaussian, and NoisyNet on `"Cartpole-v0"` with single-frame observations (*i.e.,* "Cartpole-1" in the paper) and $\sigma = 0.2$.
- `ae_step2.sh` tests the three agents trained by `ae_step1.sh` and certifies their robustness. The results are plotted as a `.pdf` figure and saved in the directory `./exp_cert/comparisons/` .
- `ae_step3.sh` attacks the `CAMP` and Gaussian agents trained by `ae_step1.sh` using projected gradient (PGD) attack [2] and plots the result in a `.pdf` figure in the `./attacks/` folder.
- `ae_step4.sh` attacks the `CAMP` and Gaussian agents trained by `ae_step1.sh` using Auto PGD (APGD) attack [1] and plots the result in a `.pdf` figure in the `./attacks/` folder.

The results from `ae_step1.sh` and `ae_step2.sh` support our first claim, while those from `ae_step3.sh` and `ae_step4.sh` support our second claim.

#### A.4.1 Major Claims

**(C1):** *`CAMP` and policy imitation enhance the certified robustness of DRL agents. This is demonstrated by the experimental results in Figures 2 and 8 of the paper.*

**(C2):** *`CAMP` and policy imitation effectively improves the robustness of agents against empirical attacks perturbing*

*observations. This is demonstrated by the experimental results in Figures 3 and 4 of the paper.*

### A.4.2 Experiments

**(E1):** *[Train agents with different methods on Cartpole-1 and certify the robustness of trained agents] [1 human-minute + 2.5 compute-hour]: Train agents with different methods and certify their robustness to support **(C1)**.*
**How to:**
**Preparation:** *Complete installation described in Section A.3 first.*
**Execution:** *1) In the root directory of the cloned repository, run* `bash ae_step1.sh` *. The script trains* `CAMP`*, Gaussian, and NoisyNet agents in the Cartpole-1 environment with the noise scale* $\sigma = 0.2$*. The trained agents are saved under the* `./exp/` *folder. 2) When finished, run* `bash ae_step2.sh` *. This script loads the trained agents and tests each agent in Cartpole-1 for* $10,000$ *independent episodes. The test rewards will be saved under* `./eval_exp/` *. Then the test rewards will be loaded and used for robustness certification. The plotted certification results will be saved in the directory* `./exp_cert/comparisons/` *.*
**Results:** *The plotted results should closely resemble those in Figure 2 (sub-figure "Cartpole-1" when* $\sigma = 0.2$*) of the paper. The certified curve for* `CAMP` *should be higher than those of the Gaussian and NoisyNet baselines.*

**(E2):** *[Apply PGD and APGD attacks to evaluate empirical robustness.] [1 human-minute + 2 compute-hour]: Compare the robustness of* `CAMP` *agent and Gaussian agent to support **(C2)**.*
**How to:**
**Preparation:** *Complete the installation described in Section A.3 and **(E1)** to ensure that trained agents are in place.*
**Execution:** *1) In the root directory of the cloned repository, run* `bash ae_step3.sh` *. This script attacks previously trained* `CAMP` *and Gaussian agents using PGD across different perturbation budgets. The attack results will be saved under* `./exp/` *. The comparison between* `CAMP` *and Gaussian agents will be plotted in* `attacks/cartpole_simple_pgd_threshold=0.pdf` *. 2) Run* `bash ae_step4.sh` *. This script attacks previously trained* `CAMP` *and Gaussian agents using APGD across different perturbation budgets. The attack results will again be saved under* `./exp/` *. The comparison between* `CAMP` *and Gaussian agents will be plotted in* `cartpole_simple_apgd_threshold=0.pdf` *.*
**Results:** *To reduce time costs for evaluators, we have decreased the number of evaluations from 1,000 to 100 when computing the average return in this artifact evaluation. The expected results should reflect the trends in Figures 3 and 4 (sub-figure "Cartpole-1" when* $\sigma = 0.2$*). The average return of the* `CAMP` *agent should be significantly higher than that of the Gaussian agent.*

## A.5 Notes on Reusability

We have included four scripts in our code repository to enable a fuss-free reproduction of the training-testing-certification pipeline and empirical robustness evaluation tasks in "Cartpole-1" under a specific hyper-parameter setting ($\sigma = 0.2$). This configuration represents one of the major improvement cases (*i.e.,* "Cartpole" and "Highway"), and other configurations can be evaluated by simply setting the `env_id` and `env_sigma` arguments in the scripts. Beyond these scripts, detailed instructions for running experiments in different environments with various hyper-parameter settings are available in the `readme.md` file. This also enables others to adapt `CAMP` and policy imitation to agents and environments beyond those tested in our paper. For comprehensive information and access to the source code, please visit our GitHub repository: https://github.com/NeuralSec/camp-robust-rl.

## 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/usenixsec2025/`.

## References


[1] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, 2020.

[2] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

[3] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulao, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.