

USENIX Security '26 Artifact Appendix
XGUARDIAN: Towards Generalized, Explainable and More Effective Server-side Anti-cheat in First-Person Shooter Games

Jiayi Zhang
The University of Hong Kong
brucejiayi@connect.hku.hk

Chenxin Sun
The University of Hong Kong
roniny@connect.hku.hk

Chenxiong Qian
The University of Hong Kong
cqian@cs.hku.hk

A Artifact Appendix

This is the artifact evaluation document for XGUARDIAN, which is going to appear at USENIX Security '26.

A.1 Abstract

You may follow the instructions below to access, reuse, and reproduce XGUARDIAN as well as the experiments mentioned in our paper.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

Data Source and Acquisition. All data used in this study were provided by a third-party competitive gaming platform *5EPlay* that hosts CS2 servers independently of *Valve Corporation* (the game's developer). These servers operate under the platform's own user agreements and privacy policies, while authentication occurs via Steam. *5EPlay* directly provided the de-identified dataset to the research team, in compliance with their platform policies[1]. No private information or personally identifiable data was disclosed to the authors. Players' re-identifiable information was removed prior to data transfer.

Data Usage Agreement. The dataset is governed by the *5EPlay Personal Information Protection Policy*, which permits sharing de-identified information for academic research purposes conducted in the public interest. While the data usage agreement acknowledged by players was not specific to this research, it explicitly allows the platform to share anonymized gameplay data for research. All data were handled with strict confidentiality and in accordance with applicable legal and ethical standards.

EXPLAINER Potential Impact. EXPLAINER could theoretically influence reviewers' decisions, potentially increasing the risk of false decisions. To mitigate this risk, EXPLAINER is strictly a supportive interpretability tool intended to help experienced game masters (GMs) quickly identify anomalous gameplay segments. All decisions are made independently by trained GMs, and bans are only enforced when multiple reviewers reach the same conclusion, minimizing the influence of any single reviewer or algorithm. GMs should be trained to know that XGUARDIAN can make mistakes. Players affected by false positives retain the right to appeal, triggering review by a separate moderation team.

IRB Review. This research was reviewed and approved by the Human Research Ethics Committee (Reference Number: EA250702) at the author's affiliated institution. The study design, data acquisition, and handling procedures were deemed compliant with human subjects research standards.

A.2.2 How to access

We open-source XGUARDIAN's repository, datasets, and artifact at https://doi.org/10.5281/zenodo.17845613 (concept DOI, link to the latest version of the artifact), as well as its raw CS2 data listed in Table 1.

Table 1: XGUARDIAN's raw data.

Part	URL
1	https://doi.org/10.5281/zenodo.17838584
2	https://doi.org/10.5281/zenodo.17838763
3	https://doi.org/10.5281/zenodo.17838827
4	https://doi.org/10.5281/zenodo.17838865
5	https://doi.org/10.5281/zenodo.17839153
6	https://doi.org/10.5281/zenodo.17844922
7	https://doi.org/10.5281/zenodo.17844928
8	https://doi.org/10.5281/zenodo.17844938
9	https://doi.org/10.5281/zenodo.17844942
10	https://doi.org/10.5281/zenodo.17844950
11	https://doi.org/10.5281/zenodo.17844964
12	https://doi.org/10.5281/zenodo.17844969
13	https://doi.org/10.5281/zenodo.17844977

[1] *5EPlay Personal Information Protection Policy*, translated by the authors, original in Simplified Chinese: https://arena.5eplay.com/page/privacy.

A.2.3 Hardware dependencies

Technically, any PC with CPU is able to reuse XGUARDIAN directly, and any PC with NVIDIA GPU(s) can train and reproduce XGUARDIAN. In terms of the experimental configurations, XGUARDIAN is developed and tested on two different servers. XGUARDIAN's preprocessing is implemented on an Ubuntu server equipped with 128 cores AMD EPYC 7713P 64-Core Processor and 256 GB RAM. The rest of the XGUARDIAN's experiments including training, validation, testing, metric calculations, and other server-side experiments are conducted on an Ubuntu server equipped with a 24-core Intel Xeon Gold 5418Y CPUs, 503 GB RAM and eight NVIDIA GeForce RTX 4090 24GB (only used two of them for the training).

A.2.4 Software dependencies

XGUARDIAN uses go version `go1.22.5 linux/amd64` for preprocessing. The Go compiler can be downloaded from https://go.dev/dl/. Before parsing, use the following command to ensure that `go` is available in your environment.

```
$ export PATH=$PATH:/usr/local/go/bin
```

XGUARDIAN uses NVIDIA-SMI `570.124.06`, driver version `570.124.06`, CUDA `12.8`, TensorFlow `2.17.0`, and cuDNN `8` for training XGUARDIAN. Similar to all ML-based solutions, these are the basic prerequisites for an ML-based project. This project uses `conda` to manage packages and environments. This project is implemented in Python (https://www.python.org/downloads/) version `3.11.8`.

A.2.5 Benchmarks

Three datasets are required: CS2 (main dataset used for most of the experiments), FARLIGHT84 and HAWK (two sets are used for the comparision and generalizability experiments). All datasets are included in Appendix A.2.2.

A.3 Evaluation workflow

The evaluation can be proceed in the following sequence.

Step 1. Preprocessing. This step is optional if you decide to use our processed datasets and/or results. For the details, please refer to Appendix A.5.

Step 2. Deploy INSPECTOR. It is the implementation of XGUARDIAN's detection function. For the detailed usage, please refer to Appendix A.6.1.

Step 3. Deploy EXPLAINER. It is the implementation of XGUARDIAN's interpretability. For the detailed usage, please refer to Appendix A.6.2.

Step 4. Reproduce the experiments. Please refer to Appendix A.7.

A.3.1 Major Claims

(C1): This artifact is able to be publicly available and permanently accessed.

(C2): This artifact is able to reuse XGUARDIAN, its datasets, trained models, codes and results.

(C3): This artifact is able to reproduce our paper experimental results.

A.4 Set-up

[*Installation*] Create a Conda environment for preprocessing. Replace the `prefix` field in `preprocessing_clean.yml` with your local `conda` path.

```
$ conda env create -f XGuardian/XGuardian/env/
    ↪ preprocessing_env/preprocessing_clean.yml
$ conda activate XGuardian-preprocessing
$ pip install -r XGuardian/XGuardian/env/
    ↪ preprocessing_env/
    ↪ preprocessing_pip_requirements.txt
```

Create a detector environment. Replace the `prefix` field in `detector_clean.yml` with your local `conda` path.

```
$ conda env create -f XGuardian/XGuardian/env/
    ↪ detector/detector_clean.yml
$ conda activate detector_clean
$ pip install -r XGuardian/XGuardian/env/detector/
    ↪ detector_pip_requirements.txt
```

Create an explainer environment. Replace the `prefix` field in `explainer_clean.yml` with your local `conda` path.

```
$ conda env create -f XGuardian/XGuardian/env/
    ↪ explainer/explainer_clean.yml
$ conda activate explainer_clean
$ pip install -r XGuardian/XGuardian/env/explainer
    ↪ /explainer_pip_requirements.txt
```

[*Troubleshooting*] If you are using an archived version of the artifact (v1), you might encounter the following issue. We use both `conda install` and `pip install`. When exporting the environment, `pip freeze` might capture the absolute path of the packages instead of their version. Might encounter an exception below.

```
ERROR: Could not install packages due to an
    ↪ OSError: [Errno 2] No such file or
    ↪ directory: /croot/xxx
```

To solve this problem, when a line is inside `requirement.txt` that contains `@ file:///croot`, discard everything starting from `@`. For example, below. Alternatively, you may download our revised version of the affected environments from: https://doi.org/10.5281/zenodo.18194986. This issue is fixed in the latest version of our artifact.

```
Bottleneck @file:///croot/bottleneck_1707864210935
    ↪ /work --> Bottleneck
```

A.5 Preprocessing

The raw data can be found at Zendo (please check Table 1 above). The raw data is **EXTREMELY** large, we strongly recommend you to use our processed data (can be found in `XGuardian/XGuardian/Design/dataset`). **If you would like to use our processed dataset directly, please skip this entire subsection**.

A.5.1 Parsing

Download the raw data and unzip the files.

```
$ cat XGuardian_raw.tar.zst.part.* > XGuardian_raw
    ↪ .tar.zst
$ tar -I zstd -xf XGuardian_raw.tar.zst
```

The initial file structure should be prepared as follows.

```
.
`-- demo_dir/
    |-- demo_0.zip
    |-- demo_1.zip
    |-- ...
    `-- demo_x.zip
```

We use the absolute path when zipping the artifact. You should rename the path and delete `home2/data1/J.Zhang/` ahead of the `XGuardian/XGuardian/`. Activate the conda environment `XGuradian-preprocessing`. To process concurrently, partition the demos first. You need to enter the demos' root directory `./demo_dir` in the command line.

```
$ python partition_creation.py
```

The file structure will then be partitioned as follows.

```
.
`-- demo_dir/
    |-- partition_0/
    |   |-- demo_0.zip
    |   |-- demo_1.zip
    |   |-- ...
    |   `-- demo_x.zip
    |-- ...
    `-- partition_y/
        |-- demo_x+n.zip
        |-- demo_x+n+1.zip
        |-- ...
        `-- demo_x+n+m.zip
```

To parse the demos concurrently, simply run `shell` script as follows. Note that you should manually modify the number of partitions in the `for i in {0..y}` loop, where `y` represents the total number of partitions, the `SCRIPT_PATH` where you store the source codes, and the `BASE_DIR` where it is the demos' root directory.

```
$ chmod +x run_preprocessing_abwh.sh
$ ./run_preprocessing_abwh.sh
```

If processing considerable demos, it will take a long time. After the parsing, the file structure should look like this. The `log.txt` under each partition folder contains error messages while parsing the demos.

```
.
`-- demo_dir/
    |-- partition_0/
    |   |-- demo_0/
    |   |   |-- demo_0_bomb.csv
    |   |   |-- demo_0_damages.csv
    |   |   |-- demo_0_header.txt
    |   |   |-- demo_0_infernos.csv
    |   |   |-- demo_0_kills.csv
    |   |   |-- demo_0_smokes.csv
    |   |   |-- demo_0_weapon_fires.csv
    |   |   |-- demo_0.dem
    |   |   `-- demo_0.zip
    |   |-- ...
    |   |-- demo_x/
    |   |   |-- demo_x_bomb.csv
    |   |   |-- demo_x_damages.csv
    |   |   |-- demo_x_header.txt
    |   |   |-- demo_x_infernos.csv
    |   |   |-- demo_x_kills.csv
    |   |   |-- demo_x_smokes.csv
    |   |   |-- demo_x_weapon_fires.csv
    |   |   |-- demo_x.dem
    |   |   `-- demo_x.zip
    |   `-- log.txt
    |-- ...
    `-- partition_y/
        |-- demo_x+n/
        |   |-- demo_x+n_bomb.csv
        |   |-- demo_x+n_damages.csv
        |   |-- demo_x+n_header.txt
        |   |-- demo_x+n_infernos.csv
        |   |-- demo_x+n_kills.csv
        |   |-- demo_x+n_smokes.csv
        |   |-- demo_x+n_weapon_fires.csv
        |   |-- demo_x+n.dem
        |   `-- demo_x+n.zip
        |-- ...
        |-- demo_x+n+m/
        |   |-- demo_x+n+m_bomb.csv
        |   |-- demo_x+n+m_damages.csv
        |   |-- demo_x+n+m_header.txt
        |   |-- demo_x+n+m_infernos.csv
        |   |-- demo_x+n+m_kills.csv
        |   |-- demo_x+n+m_smokes.csv
        |   |-- demo_x+n+m_weapon_fires.csv
        |   |-- demo_x+n+m.dem
        |   `-- demo_x+n+m.zip
        `-- log.txt
```

Run the following script to check the integrity and filter the redundant data. The redundant data will be moved to a new directory under the root directory named `redundant`. If exceptions occurred, detailed logs will be presented in `integrity_check_log.txt`.

```
$ python after_parse_itegrity_check.py <
    ↪ DEMO_ROOT_DIR>
```

The file structure should then look like this.

```
.
`-- demo_dir/
    |-- partition_0/
    |   |-- demo_0/
    |   |   |-- demo_0_bomb.csv
    |   |   |-- demo_0_damages.csv
    |   |   |-- demo_0_header.txt
```

```
|   |   |-- demo_0_infernos.csv
|   |   |-- demo_0_kills.csv
|   |   |-- demo_0_smokes.csv
|   |   |-- demo_0_weapon_fires.csv
|   |   |-- demo_0.dem
|   |   `-- demo_0.zip
|   |-- ...
|   `-- log.txt
|-- ...
|-- partition_y/
|   |-- demo_x+n/
|   |   |-- demo_x+n_bomb.csv
|   |   |-- demo_x+n_damages.csv
|   |   |-- demo_x+n_header.txt
|   |   |-- demo_x+n_infernos.csv
|   |   |-- demo_x+n_kills.csv
|   |   |-- demo_x+n_smokes.csv
|   |   |-- demo_x+n_weapon_fires.csv
|   |   |-- demo_x+n.dem
|   |   `-- demo_x+n.zip
|   |-- ...
|   |-- demo_x+n+m/
|   |   |-- demo_x+n+m_bomb.csv
|   |   |-- demo_x+n+m_damages.csv
|   |   |-- demo_x+n+m_header.txt
|   |   |-- demo_x+n+m_infernos.csv
|   |   |-- demo_x+n+m_kills.csv
|   |   |-- demo_x+n+m_smokes.csv
|   |   |-- demo_x+n+m_weapon_fires.csv
|   |   |-- demo_x+n+m.dem
|   |   `-- demo_x+n+m.zip
|   `-- log.txt
`-- redundant/
    `-- .../
        `-- ...
```

Due to enginnering consideration, some redundant folders might be renamed with an addtional suffix `_0`, you can run the following scripts to rename them to the original form.

```
$ python redundant_folders_rename.py
> Enter the root directory path: <DEMO_ROOT_DIR>
> [printed info]
> Do you want to proceed with renaming? (y/n): y
> [printed info]
```

Next, label the cheaters within each match. The exceptions will be logged in `find_cheater_log.txt`.

```
$ python Find_Cheaters_in_Folder.py <DEMO_ROOT_DIR
    ↪ > <CHEATER_LIST_EXCEL_DIR>
```

Then your updated file structure should look like this. Each demo directory will now include a `cheaterlist.csv` file.

```
.
`-- demo_dir/
    |-- partition_0/
    |   |-- demo_0/
    |   |   |-- cheaterlist.csv
    |   |   |-- demo_0_bomb.csv
    |   |   |-- demo_0_damages.csv
    |   |   |-- demo_0_header.txt
    |   |   |-- demo_0_infernos.csv
    |   |   |-- demo_0_kills.csv
    |   |   |-- demo_0_smokes.csv
    |   |   |-- demo_0_weapon_fires.csv
    |   |   |-- demo_0.dem
    |   |   `-- demo_0.zip
    |   |-- ...
    |   `-- log.txt
    |-- ...
    |-- partition_y/
    |   |-- demo_x+n/
    |   |   |-- cheaterlist.csv
    |   |   |-- demo_x+n_bomb.csv
    |   |   |-- demo_x+n_damages.csv
    |   |   |-- demo_x+n_header.txt
    |   |   |-- demo_x+n_infernos.csv
    |   |   |-- demo_x+n_kills.csv
    |   |   |-- demo_x+n_smokes.csv
    |   |   |-- demo_x+n_weapon_fires.csv
    |   |   |-- demo_x+n.dem
    |   |   `-- demo_x+n.zip
    |   |-- ...
    |   |-- demo_x+n+m/
    |   |   |-- cheaterlist.csv
    |   |   |-- demo_x+n+m_bomb.csv
    |   |   |-- demo_x+n+m_damages.csv
    |   |   |-- demo_x+n+m_header.txt
    |   |   |-- demo_x+n+m_infernos.csv
    |   |   |-- demo_x+n+m_kills.csv
    |   |   |-- demo_x+n+m_smokes.csv
    |   |   |-- demo_x+n+m_weapon_fires.csv
    |   |   |-- demo_x+n+m.dem
    |   |   `-- demo_x+n+m.zip
    |   `-- log.txt
    `-- redundant/
        `-- .../
            `-- ...
```

[***Optional***] Some raw zip files might be corrupted during the download, you may use the following script to remove them.

```
$ python remove_corrupt_zips.py <DEMO_ROOT_DIR>
```

A.5.2 Trajectories Extraction

Run the following `shell` scripts to concurrently process the trajectory data. You should modify the directory and number of partitions based on your own data.

```
$ chmod +x run_3D_conversion_modified.sh
$ chmod +x run_2D_conversion_modified.sh
$ ./run_3D_conversion_modified.sh
$ ./run_2D_conversion_modified.sh
```

The logs (if any) for 2D/3D trajectory converters will be stored as `2D_trajectory_modified.txt` and `3D_trajectory_modified.txt`, respectively. The file structure should then look like this.

```
.
`-- demo_dir/
    |-- partition_0/
    |   |-- demo_0/
    |   |   |-- 2D_trajectory_modified.csv
    |   |   |-- 3D_trajectory_modified.csv
    |   |   |-- cheaterlist.csv
    |   |   |-- demo_0_bomb.csv
    |   |   |-- demo_0_damages.csv
    |   |   |-- demo_0_header.txt
    |   |   |-- demo_0_infernos.csv
    |   |   |-- demo_0_kills.csv
    |   |   |-- demo_0_smokes.csv
    |   |   |-- demo_0_weapon_fires.csv
```

```
|   |   |-- demo_0.dem
|   |   `-- demo_0.zip
|   |-- ...
|   `-- log.txt
|-- ...
|-- partition_y/
|   |-- demo_x+n/
|   |   |-- 2D_trajectory_modified.csv
|   |   |-- 3D_trajectory_modified.csv
|   |   |-- cheaterlist.csv
|   |   |-- demo_x+n_bomb.csv
|   |   |-- demo_x+n_damages.csv
|   |   |-- demo_x+n_header.txt
|   |   |-- demo_x+n_infernos.csv
|   |   |-- demo_x+n_kills.csv
|   |   |-- demo_x+n_smokes.csv
|   |   |-- demo_x+n_weapon_fires.csv
|   |   |-- demo_x+n.dem
|   |   `-- demo_x+n.zip
|   |-- ...
|   |-- demo_x+n+m/
|   |   |-- 2D_trajectory_modified.csv
|   |   |-- 3D_trajectory_modified.csv
|   |   |-- cheaterlist.csv
|   |   |-- demo_x+n+m_bomb.csv
|   |   |-- demo_x+n+m_damages.csv
|   |   |-- demo_x+n+m_header.txt
|   |   |-- demo_x+n+m_infernos.csv
|   |   |-- demo_x+n+m_kills.csv
|   |   |-- demo_x+n+m_smokes.csv
|   |   |-- demo_x+n+m_weapon_fires.csv
|   |   |-- demo_x+n+m.dem
|   |   `-- demo_x+n+m.zip
|   `-- log.txt
`-- redundant/
    `-- .../
        `-- ...
```

A.6 XGUARDIAN General Usage

This section describes the general usage of XGUARDIAN's core detection and explanation components.

A.6.1 INSPECTOR

Activate the conda environment `detector_clean`. Run `XGuardian-Inspector.py` with the following command, where `<train_path>`, `<val_path>`, `<test_path>` are the directories of the data splits. According to our paper findings, `<window_size>` should be `6`, `<stride>` should be `1`. `<drop_columns>` is the parameter to drop some redundant columns in the dataset. This should be set as `"isCheater,steamid,kill_num,match_id,in_crouch,is_walking,is_strafing,weapon,angle,x,y,distance"`. Note that at Line 82-83, the GPUs are designated; please adjust according to your device configurations.

```
$ python XGuardian-Inspector.py <train_path> <
    ↪ val_path> <test_path> <output_dir> <
    ↪ window_size> <stride> <drop_columns>
```

[*Directories of the datasets*] Except the detection performance (Appendix A.7.1) and elimination window size comparison (Appendix A.7.3), the `<train_path>`, `<val_path>`, `<test_path>` are as follows. The directories of the above two sections are explicitly stated in the relevant sections.

```
<train_path>:
/home2/data1/J.Zhang/XGuardian/XGuardian/Design/
    ↪ dataset/Train_cleaned.csv
<val_path>:
/home2/data1/J.Zhang/XGuardian/XGuardian/Design/
    ↪ dataset/Validation_cleaned.csv
<test_path>:
/home2/data1/J.Zhang/XGuardian/XGuardian/Design/
    ↪ dataset/Test_cleaned.csv
```

After execution, three CSVs will be generated named as `aggregated_preditions_x.csv`, where x is corresponding to training, validation or testing set, respectively. These CSVs should be first processed with the following commands for some of the following experiments. `<input_csv_path>` should be the directories of the `aggregated_preditions_x.csv`. `aggregated_preditions_x.csv` is the raw output for each elimination event, we use as an intermediate data it for processing the match predictions

```
$ python XGuardian-Inspector-Match-Preprocessing.
    ↪ py <input_csv_path> <output_csv_path>
```

Then run `XGuardian-Inspector-Match.py` for the final detection result. In which, `<feature_selection>` can be set as `'o'` (original, please use this one normally), `'b'` (binary), and `'a'` (all, original+binary). For details, please refer to Section 5.4.5. Note that at Line 46-47, the GPUs are designated; please adjust according to your device configurations.

```
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match.py --
    ↪ train_path <training_path> --val_path <
    ↪ validation_path> --test_path <test_path> --
    ↪ feature_set <feature_selection> --
    ↪ output_dir <output_directory>
```

In the output directory, `round_integ_x_predictions_n.csv` will be generated, where x is corresponding to training, validation or testing set; n is corresponding to original, binary and all (feature selection). The above CSV is the final detection results of XGUARDIAN, you may use `round_eval_metrics_stats.py` to evaluate the metrics. `round_integ_x_predictions_n.csv` is the final prediction results of XGUARDIAN for each set, `original` means we are applying `'o'` settings for predictions stage embedding style comparison. **[*Important note*]** You should always manually change the directory of `round_integ_x_predictions_n.csv` in `round_eval_metrics_stats.py` at Line 5 before using it.

```
$ python round_eval_metrics_stats.py
```

A.6.2 EXPLAINER

Activate the conda environment `explainer_clean`. Run `XGuardian-Explainer-Elimination.py` to generate raw

SHAP values per elimination. This script's usage is very similar to XGuardian-Inspector.py. <window_size> should be 6, <stride> should be 1. <drop_columns> is the parameter to drop some redundant columns in the dataset. This should be set as "isCheater,steamid,kill_num, match_id,in_crouch,is_walking,is_strafing, weapon,angle,x,y,distance". This script will not train the detection model again, but will use the trained models (check Line 270), so please make sure the model directories are correct.

```
$ python XGuardian-Explainer.py <train_path> <
    ↪ val_path> <test_path> <output_dir> <
    ↪ window_size> <stride> <drop_columns>
```

The above execution will generate shap_values _with_ids.npz. Then run XGuardian-Explainer-Elimination-xguardian_value_avg.py to generate the XGUARDIAN values per elimination. You should manually change the input directories: csv_file_path (Line 5, your test set directory, i.e., Test_cleaned.csv) and shap_values (Line 10, your shap_values_with_ids.npz directory), and output_csv_file_path (Line 81).

```
$ python XGuardian-Explainer-Elimination-
    ↪ xguardian_value_avg.py
```

For elimination visualization, please check /home2 /data1/J.Zhang/XGuardian/XGuardian/Design/src /Explainer_vis folder. There are some preprocessing needs to be done. Run split_shap_csv_to_subCSVs.py. At Line 38-39, you should change the input (the XGuardian-Explainer-Elimination-xguardian_value_avg.py generated CSV) and output directories.

```
$ python split_shap_csv_to_subCSVs.py
```

Then it will generate a subfolder test_set_data. Run 2D _vis.sh to concurrently visualize the trajectory. In which, at Line 4-5, change the input and output directories. Also, at Line 22, make sure the Python script directory is correct (using XGuardian/XGuardian/Design/src/Explainer_vis /2D_vis_batch_concurrent.py). At Line 30-31, you should adjust the total number of the cheaters and normal players (check the numbering in test_set_data, no need to replace if using our dataset). Due to a different coordination system, the final visualizations (PNGs and MP4s) need to be center-rotated 180 degrees to match the user's aiming trajectories.

```
$ ./2D_vis.sh
```

For match EXPLAINER visualization, run XGuardian-Explainer-Match.py. <model_path> is the directory of logistic_regression_model_original.joblib (this is the model of Match-Inspector, just ignore the naming, we are actually using random forest).

```
$ python XGuardian-Explainer-Match.py --
    ↪ dataset_path <test_set_directory> --
    ↪ train_path <training_set_directory> --
    ↪ model_path <model_path> --feature_set <
    ↪ feature_selection> --output_dir <
    ↪ output_directory>
```

A.7 Experiments

This subsection contains the instructions to run and check the raw results of each experiment in our paper. There are some redundant files for debugging and testing, which can be ignored for reusing/reproducing XGUARDIAN. Please follow the instructions below to reuse/reproduce XGUARDIAN. After v2, each experiment folder contains its own round_eval _metrics_stats.py for evaluation, please ignore all firectory modification instructions in the following section.

A.7.1 Detection Performance

[*To check the raw results*] Check each round_integ _test_predictions_original.csv under each subfolder of /home2/data1/J.Zhang/XGuardian/XGuardian /Experiments/5.2_Detection_Performace. Use round _eval_metrics_stats.py to calculate the metrics.

```
$ python round_eval_metrics_stats.py
```

[*To reproduce*] Run the following command by replacing the order of the train, validation, and test sets in the dataset thoroughly with the conda environment detector_clean activated.

```
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector.py <
    ↪ train_path> <val_path> <test_path> <
    ↪ output_dir> 6 1 "isCheater,steamid,kill_num
    ↪ ,match_id,in_crouch,is_walking,is_strafing,
    ↪ weapon,angle,x,y,distance"
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match-
    ↪ Preprocessing.py <path_of_each_subfolder>/
    ↪ aggregated_predictions_train.csv <
    ↪ path_of_each_subfolder>/
    ↪ aggregated_predictions_train_cleaned.csv
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match-
    ↪ Preprocessing.py <path_of_each_subfolder>/
    ↪ aggregated_predictions_val.csv <
    ↪ path_of_each_subfolder>/
    ↪ aggregated_predictions_val_cleaned.csv
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match-
    ↪ Preprocessing.py <path_of_each_subfolder>/
    ↪ aggregated_predictions_test.csv <
    ↪ path_of_each_subfolder>/
    ↪ aggregated_predictions_test_cleaned.csv
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match.py --
    ↪ train_path <path_of_each_subfolder>/
    ↪ aggregated_predictions_train_cleaned.csv --
    ↪ val_path <path_of_each_subfolder>/
    ↪ aggregated_predictions_val_cleaned.csv --
    ↪ test_path <path_of_each_subfolder>/
    ↪ aggregated_predictions_test_cleaned.csv --
    ↪ feature_set 'o' --output_dir <
    ↪ output_directory>
```

[***Directories***] In this experiments, the directories of would be depending on which set combination is applied to verify the results. For example if now testing the combination of `TestTrainVal`, the directories of `<train_path>`, `<val_path>`, `<test_path>` should be:

```
<train_path>:
/home2/data1/J.Zhang/XGuardian/XGuardian/Design/
    ↪ dataset/Test_cleaned.csv
<val_path>:
/home2/data1/J.Zhang/XGuardian/XGuardian/Design/
    ↪ dataset/Train_cleaned.csv
<test_path>:
/home2/data1/J.Zhang/XGuardian/XGuardian/Design/
    ↪ dataset/Validation_cleaned.csv
```

All of the results can be found at `/home2/data1/J.Zhang/XGuardian/XGuardian/Experiments/5.2_Detection_Performace/overall_performance/xxx/round_integ_test_predictions_original.csv`, where `xxx` are `TrainValTest`, `TrainTestVal`, `ValTrainTest`, `ValTestTrain`, `TestTrainVal`, `TestValTrain`. Please then use `/home2/data1/J.Zhang/XGuardian/XGuardian/Design/src/round_eval_metrics_stats.py` to evaluate each set combination results (remember to change different directory at Line 5 in `round_eval_metrics_stats.py`). The returned results from `round_eval_metrics_stats.py` would be the results of Section 5.2.

A.7.2 Feature Ablation Study

[***To check the raw results***] Check each `round_integ_test_predictions_original.csv` under each subfolder of `/home2/data1/J.Zhang/XGuardian/XGuardian/Experiments/5.3_Feature_Ablation_Study/feature_selection`. Use `round_eval_metrics_stats.py` to calculate the metrics.

```
$ python round_eval_metrics_stats.py
```

[***To reproduce***] Run the following command by modifying the removed columns parameter from the dataset with the conda environment `detector_clean` activated. As default, `<drop_columns>` would be `"isCheater,steamid,kill_num,match_id,in_crouch,is_walking,is_strafing,weapon,angle,x,y,distance"`. This is the case `all`, where all of the features are included. When doing the ablation study, one might just add the feature that would like to be ablated in `<drop_columns>`. For example, when removing angular change, add `angle_change` to the default parameter. Other valid options to be added are `velocity_x,velocity_y,acceleration_x,acceleration_y`.

```
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector.py XGuardian
    ↪ /XGuardian/Design/dataset/Train_cleaned.csv
    ↪  XGuardian/XGuardian/Design/dataset/
    ↪ Validation_cleaned.csv XGuardian/XGuardian/
    ↪ Design/dataset/Test_cleaned.csv <output_dir
    ↪ > 6 1 <drop_columns>
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match-
    ↪ Preprocessing.py <path_of_each_subfolder>/
    ↪ aggregated_predictions_train.csv <
    ↪ path_of_each_subfolder>/
    ↪ aggregated_predictions_train_cleaned.csv
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match-
    ↪ Preprocessing.py <path_of_each_subfolder>/
    ↪ aggregated_predictions_val.csv <
    ↪ path_of_each_subfolder>/
    ↪ aggregated_predictions_val_cleaned.csv
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match-
    ↪ Preprocessing.py <path_of_each_subfolder>/
    ↪ aggregated_predictions_test.csv <
    ↪ path_of_each_subfolder>/
    ↪ aggregated_predictions_test_cleaned.csv
$ python /home2/data1/J.Zhang/XGuardian-Inspector-
    ↪ Match.py --train_path <
    ↪ path_of_each_subfolder>/
    ↪ aggregated_predictions_train_cleaned.csv --
    ↪ val_path <path_of_each_subfolder>/
    ↪ aggregated_predictions_val_cleaned.csv --
    ↪ test_path <path_of_each_subfolder>/
    ↪ aggregated_predictions_test_cleaned.csv --
    ↪ feature_set 'o' --output_dir <
    ↪ output_directory>
```

A.7.3 Elimination Window Comparison

[***To check the raw results***] Check each `round_integ_test_predictions_original.csv` under each subfolder of `/home2/data1/J.Zhang/XGuardian/XGuardian/Experiments/5.4.1_Different_Elimination_Window_Sizes/diff_window_comp/results`. Use `round_eval_metrics_stats.py` to calculate the metrics.

```
$ python round_eval_metrics_stats.py
```

[***To reproduce***] Run the following commands with the conda environment `detector_clean` activated.

```
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Experiments/5.4.1
    ↪ _Different_Elimination_Window_Sizes/
    ↪ diff_window_comp/src/XGuardian_128.py
    ↪ XGuardian/XGuardian/Experiments/5.4.1
    ↪ _Different_Elimination_Window_Sizes/
    ↪ diff_window_comp/ready_for_eval_data/128
    ↪ _train.csv XGuardian/XGuardian/Experiments/
    ↪ 5.4.1_Different_Elimination_Window_Sizes/
    ↪ diff_window_comp/ready_for_eval_data/128
    ↪ _val.csv XGuardian/XGuardian/Experiments/
    ↪ 5.4.1_Different_Elimination_Window_Sizes/
    ↪ diff_window_comp/ready_for_eval_data/128
    ↪ _test.csv <output_dir> 6 1 "isCheater,
    ↪ steamid,kill_num,match_id,in_crouch,
    ↪ is_walking,is_strafing,weapon,angle,x,y,
    ↪ distance"
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Experiments/5.4.1
    ↪ _Different_Elimination_Window_Sizes/
    ↪ diff_window_comp/src/XGuardian_256.py
    ↪ XGuardian/XGuardian/Experiments/5.4.1
    ↪ _Different_Elimination_Window_Sizes/
```

```
    ↪ diff_window_comp/ready_for_eval_data/256
    ↪ _train.csv XGuardian/XGuardian/Experiments/
    ↪ 5.4.1_Different_Elimination_Window_Sizes/
    ↪ diff_window_comp/ready_for_eval_data/256
    ↪ _val.csv XGuardian/XGuardian/Experiments/
    ↪ 5.4.1_Different_Elimination_Window_Sizes/
    ↪ diff_window_comp/ready_for_eval_data/256
    ↪ _test.csv <output_dir> 6 1 "isCheater,
    ↪ steamid,kill_num,match_id,in_crouch,
    ↪ is_walking,is_strafing,weapon,angle,x,y,
    ↪ distance"
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match-
    ↪ Preprocessing.py <path_of_each_subfolder>/
    ↪ aggregated_predictions_train.csv <
    ↪ path_of_each_subfolder>/
    ↪ aggregated_predictions_train_cleaned.csv
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match-
    ↪ Preprocessing.py <path_of_each_subfolder>/
    ↪ aggregated_predictions_val.csv <
    ↪ path_of_each_subfolder>/
    ↪ aggregated_predictions_val_cleaned.csv
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match-
    ↪ Preprocessing.py <path_of_each_subfolder>/
    ↪ aggregated_predictions_test.csv <
    ↪ path_of_each_subfolder>/
    ↪ aggregated_predictions_test_cleaned.csv
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match.py --
    ↪ train_path <path_of_each_subfolder>/
    ↪ aggregated_predictions_train_cleaned.csv --
    ↪ val_path <path_of_each_subfolder>/
    ↪ aggregated_predictions_val_cleaned.csv --
    ↪ test_path <path_of_each_subfolder>/
    ↪ aggregated_predictions_test_cleaned.csv --
    ↪ feature_set 'o' --output_dir <
    ↪ output_directory>
```

[*Directories*] The directories of <train_path>, <val_path>, <test_path> for elimination window size = 384(-256/+128) should be:

```
<train_path>:
/home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Experiments/5.4.1
    ↪ _Different_Elimination_Window_Sizes/
    ↪ diff_window_comp/original_data/2
    ↪ D_tra_vel_acc_train.csv
<val_path>:
/home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Experiments/5.4.1
    ↪ _Different_Elimination_Window_Sizes/
    ↪ diff_window_comp/original_data/2
    ↪ D_tra_vel_acc_val.csv
<test_path>:
/home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Experiments/5.4.1
    ↪ _Different_Elimination_Window_Sizes/
    ↪ diff_window_comp/original_data/2
    ↪ D_tra_vel_acc_test.csv
```

The directories of <train_path>, <val_path>, <test_path> for elimination window size = 192(-128/+64) should be:

```
<train_path>:
/home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Experiments/5.4.1
    ↪ _Different_Elimination_Window_Sizes/
    ↪ diff_window_comp/original_data/128_train.
    ↪ csv
<val_path>:
/home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Experiments/5.4.1
    ↪ _Different_Elimination_Window_Sizes/
    ↪ diff_window_comp/original_data/128_val.csv,
<test_path>:
/home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Experiments/5.4.1
    ↪ _Different_Elimination_Window_Sizes/
    ↪ diff_window_comp/original_data/128_test.csv
    ↪ ,
```

[*Naming*] You may have noticed that the naming for the above datasets are different from the main datasets. The file with _cleaned suffix means it was generated by the INSPECTOR. It filters out some edge cases mentioned in Section 5.1.1 - Data Cleansing of our paper and also removes some redundant columns.

A.7.4 Sliding Window Comparison

[*To check the raw results*] Check each round_integ_test_predictions_original.csv under each subfolder of /home2/data1/J.Zhang/XGuardian/XGuardian/Experiments/5.4.2_Different_Sliding_Window_Sizes/diff_sliding_window_comp. Use round_eval_metrics_stats.py to calculate the metrics.

```
$ python round_eval_metrics_stats.py
```

[*To reproduce*] Run the following commands with the conda environment detector_clean activated. Modify sliding window parameter <window_size> accordingly.

```
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector.py XGuardian
    ↪ /XGuardian/Design/dataset/Train_cleaned.csv
    ↪  XGuardian/XGuardian/Design/dataset/
    ↪ Validation_cleaned.csv XGuardian/XGuardian/
    ↪ Design/dataset/Test_cleaned.csv <output_dir
    ↪ > <window_size> 1 "isCheater,steamid,
    ↪ kill_num,match_id,in_crouch,is_walking,
    ↪ is_strafing,weapon,angle,x,y,distance"
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match-
    ↪ Preprocessing.py <path_of_each_subfolder>/
    ↪ aggregated_predictions_train.csv <
    ↪ path_of_each_subfolder>/
    ↪ aggregated_predictions_train_cleaned.csv
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match-
    ↪ Preprocessing.py <path_of_each_subfolder>/
    ↪ aggregated_predictions_val.csv <
    ↪ path_of_each_subfolder>/
    ↪ aggregated_predictions_val_cleaned.csv
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match-
    ↪ Preprocessing.py <path_of_each_subfolder>/
    ↪ aggregated_predictions_test.csv <
    ↪ path_of_each_subfolder>/
    ↪ aggregated_predictions_test_cleaned.csv
```

```
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match.py --
    ↪ train_path <path_of_each_subfolder>/
    ↪ aggregated_predictions_train_cleaned.csv --
    ↪ val_path <path_of_each_subfolder>/
    ↪ aggregated_predictions_val_cleaned.csv --
    ↪ test_path <path_of_each_subfolder>/
    ↪ aggregated_predictions_test_cleaned.csv --
    ↪ feature_set 'o' --output_dir <
    ↪ output_directory>
```

A.7.5 Different Model Comparison in Embedding Generation

[*To check the raw results*] Check each `round_integ_test_predictions_original.csv` under each subfolder of `/home2/data1/J.Zhang/XGuardian/XGuardian/Experiments/5.4.3_Different_Models_in_Embedding_Generation/embedding`. Use `round_eval_metrics_stats.py` to calculate the metrics.

```
$ python round_eval_metrics_stats.py
```

[*To reproduce*] BiLSTM and Transformer can be directly executed under `detector_clean` environment with the following commands, respectively.

```
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Experiments/5.4.3
    ↪ _Different_Models_in_Embedding_Generation/
    ↪ embedding/BiLSTM.py XGuardian/XGuardian/
    ↪ Design/dataset/Train_cleaned.csv XGuardian/
    ↪ XGuardian/Design/dataset/Validation_cleaned
    ↪ .csv XGuardian/XGuardian/Design/dataset/
    ↪ Test_cleaned.csv <output_dir> 6 1 "
    ↪ isCheater,steamid,kill_num,match_id,
    ↪ in_crouch,is_walking,is_strafing,weapon,
    ↪ angle,x,y,distance"
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Experiments/5.4.3
    ↪ _Different_Models_in_Embedding_Generation/
    ↪ embedding/Transformer.py XGuardian/
    ↪ XGuardian/Design/dataset/Train_cleaned.csv
    ↪ XGuardian/XGuardian/Design/dataset/
    ↪ Validation_cleaned.csv XGuardian/XGuardian/
    ↪ Design/dataset/Test_cleaned.csv <output_dir
    ↪ > 6 1 "isCheater,steamid,kill_num,match_id,
    ↪ in_crouch,is_walking,is_strafing,weapon,
    ↪ angle,x,y,distance"
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match-
    ↪ Preprocessing.py <path_of_each_subfolder>/
    ↪ aggregated_predictions_train.csv <
    ↪ path_of_each_subfolder>/
    ↪ aggregated_predictions_train_cleaned.csv
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match-
    ↪ Preprocessing.py <path_of_each_subfolder>/
    ↪ aggregated_predictions_val.csv <
    ↪ path_of_each_subfolder>/
    ↪ aggregated_predictions_val_cleaned.csv
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match-
    ↪ Preprocessing.py <path_of_each_subfolder>/
    ↪ aggregated_predictions_test.csv <
    ↪ path_of_each_subfolder>/
    ↪ aggregated_predictions_test_cleaned.csv
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match.py --
    ↪ train_path <path_of_each_subfolder>/
    ↪ aggregated_predictions_train_cleaned.csv --
    ↪ val_path <path_of_each_subfolder>/
    ↪ aggregated_predictions_val_cleaned.csv --
    ↪ test_path <path_of_each_subfolder>/
    ↪ aggregated_predictions_test_cleaned.csv --
    ↪ feature_set 'o' --output_dir <
    ↪ output_directory>
```

Clone the environment `detector_clean` and install `keras-tcn` (https://github.com/philipperemy/keras-tcn). A segement of the code (from Line 183 to Line 221 was made as comments during the debugging and testing phases, please cancel the comment hashtags while training).

```
$ conda create --name tcn --clone detector_clean
$ conda activate tcn
$ pip install keras-tcn
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Experiments/5.4.3
    ↪ _Different_Models_in_Embedding_Generation/
    ↪ embedding/TCN.py XGuardian/XGuardian/Design
    ↪ /dataset/Train_cleaned.csv XGuardian/
    ↪ XGuardian/Design/dataset/Validation_cleaned
    ↪ .csv XGuardian/XGuardian/Design/dataset/
    ↪ Test_cleaned.csv <output_dir> 6 1 "
    ↪ isCheater,steamid,kill_num,match_id,
    ↪ in_crouch,is_walking,is_strafing,weapon,
    ↪ angle,x,y,distance"
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match-
    ↪ Preprocessing.py <path_of_each_subfolder>/
    ↪ aggregated_predictions_train.csv <
    ↪ path_of_each_subfolder>/
    ↪ aggregated_predictions_train_cleaned.csv
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match-
    ↪ Preprocessing.py <path_of_each_subfolder>/
    ↪ aggregated_predictions_val.csv <
    ↪ path_of_each_subfolder>/
    ↪ aggregated_predictions_val_cleaned.csv
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match-
    ↪ Preprocessing.py <path_of_each_subfolder>/
    ↪ aggregated_predictions_test.csv <
    ↪ path_of_each_subfolder>/
    ↪ aggregated_predictions_test_cleaned.csv
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match.py --
    ↪ train_path <path_of_each_subfolder>/
    ↪ aggregated_predictions_train_cleaned.csv --
    ↪ val_path <path_of_each_subfolder>/
    ↪ aggregated_predictions_val_cleaned.csv --
    ↪ test_path <path_of_each_subfolder>/
    ↪ aggregated_predictions_test_cleaned.csv --
    ↪ feature_set 'o' --output_dir <
    ↪ output_directory>
```

A.7.6 Different Model Comparison in Trajectory Classification

[*To check the raw results*] Check each `round_integ_test_predictions_original.csv` under each subfolder of `/home2/data1/J.Zhang/XGuardian/XGuardian/Experiments/5.4.4_Different_Models_in_Trajectory_Classification/integration`. Use `round_eval_metrics_stats.py` to calculate the metrics.

```
$ python round_eval_metrics_stats.py
```

[*To reproduce*] Run the following command with the conda environment `detector_clean` activated.

```
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Experiments/5.4.4
    ↪ _Different_Models_in_Trajectory_Classification
    ↪ /integration/GRU-fc.py XGuardian/XGuardian/
    ↪ Design/dataset/Train_cleaned.csv XGuardian/
    ↪ XGuardian/Design/dataset/Validation_cleaned
    ↪ .csv XGuardian/XGuardian/Design/dataset/
    ↪ Test_cleaned.csv <output_dir> 6 1 "
    ↪ isCheater,steamid,kill_num,match_id,
    ↪ in_crouch,is_walking,is_strafing,weapon,
    ↪ angle,x,y,distance"
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match-
    ↪ Preprocessing.py <path_of_each_subfolder>/
    ↪ aggregated_predictions_train.csv <
    ↪ path_of_each_subfolder>/
    ↪ aggregated_predictions_train_cleaned.csv
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match-
    ↪ Preprocessing.py <path_of_each_subfolder>/
    ↪ aggregated_predictions_val.csv <
    ↪ path_of_each_subfolder>/
    ↪ aggregated_predictions_val_cleaned.csv
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match-
    ↪ Preprocessing.py <path_of_each_subfolder>/
    ↪ aggregated_predictions_test.csv <
    ↪ path_of_each_subfolder>/
    ↪ aggregated_predictions_test_cleaned.csv
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match.py --
    ↪ train_path <path_of_each_subfolder>/
    ↪ aggregated_predictions_train_cleaned.csv --
    ↪ val_path <path_of_each_subfolder>/
    ↪ aggregated_predictions_val_cleaned.csv --
    ↪ test_path <path_of_each_subfolder>/
    ↪ aggregated_predictions_test_cleaned.csv --
    ↪ feature_set 'o' --output_dir <
    ↪ output_directory>
```

A.7.7 Different Input Style Comparison in Elimination Prediction

[*To check the raw results*] Check each `round_integ_test_predictions_original.csv` under each subfolder of `/home2/data1/J.Zhang/XGuardian/XGuardian/Experiments/5.4.5_Input_Styles_in_Elimination_Predictions`. Use `round_eval_metrics_stats.py` to calculate the metrics.

```
$ python round_eval_metrics_stats.py
```

[*To reproduce*] Run `XGuardian-Inspector-Match.py` for the final detection result. In which, `<feature_selection>` can be set as `'o'` (original, please use this one normally), `'b'` (binary), and `'a'` (all, original+binary).

```
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match.py --
    ↪ train_path XGuardian/XGuardian/Experiments/
    ↪ 5.2_Detection_Performace/
    ↪ overall_performance/TrainValTest/
    ↪ aggregated_predictions_train_cleaned.csv --
    ↪ val_path XGuardian/XGuardian/Experiments/
    ↪ 5.2_Detection_Performace/
    ↪ overall_performance/TrainValTest/
    ↪ aggregated_predictions_val_cleaned.csv --
    ↪ test_path XGuardian/XGuardian/Experiments/
    ↪ 5.2_Detection_Performace/
    ↪ overall_performance/TrainValTest/
    ↪ aggregated_predictions_test_cleaned.csv --
    ↪ feature_set <feature_selection> --
    ↪ output_dir <output_directory>
```

A.7.8 Comparative Study

[*To check the raw results*] In terms of HAWK. Check each `round_integ_test_predictions_original.csv` under each subfolder of `/home2/data1/J.Zhang/XGuardian/XGuardian/Experiments/5.5_Comparative_Study`. Use `round_eval_metrics_stats.py` to calculate the metrics.

```
$ python round_eval_metrics_stats.py
```

In terms of statistical methods. Run the following commands.

```
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Experiments/5.5_Comparative_Study/
    ↪ stats_method/AccA_eval.py
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Experiments/5.5_Comparative_Study/
    ↪ stats_method/HitAcc_eval.py
```

[*To reproduce*] In terms of HAWK. Run the following commands. Due to minimizing the size of our artifact, we deleted some intermediate files (e.g., `aggregated_prediction_x.csv`, etc.), you may reproduce it by using `XGuardian-Inspector.py`.

```
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Experiments/5.5_Comparative_Study/hawk/
    ↪ hawk_xguardian.py XGuardian/XGuardian/
    ↪ Experiments/5.5_Comparative_Study/hawk/
    ↪ HAWK_train.csv XGuardian/XGuardian/
    ↪ Experiments/5.5_Comparative_Study/hawk/
    ↪ HAWK_validation.csv XGuardian/XGuardian/
    ↪ Experiments/5.5_Comparative_Study/hawk/
    ↪ HAWK_test.csv <output_dir> 6 1 "isCheater,
    ↪ steamid,kill_num,match_id,in_crouch,
    ↪ is_walking,is_strafing,weapon,angle,x,y,
    ↪ distance"
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match-
    ↪ Preprocessing.py XGuardian/XGuardian/
    ↪ Experiments/5.5_Comparative_Study/hawk/
```

```
    ↪ aggregated_predictions_train.csv XGuardian/
    ↪ XGuardian/Experiments/5.5_Comparative_Study
    ↪ /hawk/aggregated_predictions_train_cleaned.
    ↪ csv
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match-
    ↪ Preprocessing.py XGuardian/XGuardian/
    ↪ Experiments/5.5_Comparative_Study/hawk/
    ↪ aggregated_predictions_val.csv XGuardian/
    ↪ XGuardian/Experiments/5.5_Comparative_Study
    ↪ /hawk/aggregated_predictions_val_cleaned.
    ↪ csv
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match-
    ↪ Preprocessing.py XGuardian/XGuardian/
    ↪ Experiments/5.5_Comparative_Study/hawk/
    ↪ aggregated_predictions_test.csv XGuardian/
    ↪ XGuardian/Experiments/5.5_Comparative_Study
    ↪ /hawk/aggregated_predictions_test_cleaned.
    ↪ csv
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match.py --
    ↪ train_path XGuardian/XGuardian/Experiments/
    ↪ 5.5_Comparative_Study/hawk/
    ↪ aggregated_predictions_train_cleaned.csv --
    ↪ val_path XGuardian/XGuardian/Experiments/
    ↪ 5.5_Comparative_Study/hawk/
    ↪ aggregated_predictions_val_cleaned.csv --
    ↪ test_path XGuardian/XGuardian/Experiments/
    ↪ 5.5_Comparative_Study/hawk/
    ↪ aggregated_predictions_test_cleaned.csv --
    ↪ feature_set 'o' --output_dir <
    ↪ output_directory>
```

In terms of statistical methods. Run the following commands.

```
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Experiments/5.5_Comparative_Study/
    ↪ stats_method/AccA_eval.py
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Experiments/5.5_Comparative_Study/
    ↪ stats_method/HitAcc_eval.py
```

A.7.9 Generalizability

[*To check the raw results*] Check each `round_integ_test_predictions_original.csv` under each subfolder of `/home2/data1/J.Zhang/XGuardian/XGuardian/Experiments/5.6_Generalizability/farlight84/result/mobile`. Use `round_eval_metrics_stats.py` to calculate the metrics.

```
$ python round_eval_metrics_stats.py
```

[*To reproduce*] Run the following commands. Due to minimizing the size of our artifact, we deleted some intermediate files (e.g., `aggregated_prediction_x.csv`, etc.), you may reproduce it by using `XGuardian-Inspector.py`. For HAWK's results and reproducible procedures, please check the previous subsubsection.

```
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Experiments/5.6_Generalizability/farlight84
    ↪ /src/XGuardian/XGuardian-Inspector.py
    ↪ XGuardian/XGuardian/Experiments/5.6
    ↪ _Generalizability/farlight84/dataset/mobile
    ↪ /Farlight84_Mobile_train.csv XGuardian/
    ↪ XGuardian/Experiments/5.6_Generalizability/
    ↪ farlight84/dataset/mobile/
    ↪ Farlight84_Mobile_val.csv XGuardian/
    ↪ XGuardian/Experiments/5.6_Generalizability/
    ↪ farlight84/dataset/mobile/
    ↪ Farlight84_Mobile_test.csv <output_dir> 6 1
    ↪  "isCheater,steamid,kill_num,match_id,angle
    ↪ ,x,y,platform"
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match-
    ↪ Preprocessing.py XGuardian/XGuardian/
    ↪ Experiments/5.6_Generalizability/farlight84
    ↪ /result/mobile/aggregated_predictions_train
    ↪ .csv XGuardian/XGuardian/Experiments/5.6
    ↪ _Generalizability/farlight84/result/mobile/
    ↪ aggregated_predictions_train_cleaned.csv
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match-
    ↪ Preprocessing.py XGuardian/XGuardian/
    ↪ Experiments/5.6_Generalizability/farlight84
    ↪ /result/mobile/aggregated_predictions_val.
    ↪ csv XGuardian/XGuardian/Experiments/5.6
    ↪ _Generalizability/farlight84/result/mobile/
    ↪ aggregated_predictions_val_cleaned.csv
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match-
    ↪ Preprocessing.py XGuardian/XGuardian/
    ↪ Experiments/5.6_Generalizability/farlight84
    ↪ /result/mobile/aggregated_predictions_test.
    ↪ csv XGuardian/XGuardian/Experiments/5.6
    ↪ _Generalizability/farlight84/result/mobile/
    ↪ aggregated_predictions_test_cleaned.csv
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Design/src/XGuardian-Inspector-Match.py --
    ↪ train_path XGuardian/XGuardian/Experiments/
    ↪ 5.6_Generalizability/farlight84/result/
    ↪ mobile/aggregated_predictions_train_cleaned
    ↪ .csv --val_path XGuardian/XGuardian/
    ↪ Experiments/5.6_Generalizability/farlight84
    ↪ /result/mobile/
    ↪ aggregated_predictions_val_cleaned.csv --
    ↪ test_path XGuardian/XGuardian/Experiments/
    ↪ 5.6_Generalizability/farlight84/result/
    ↪ mobile/aggregated_predictions_test_cleaned.
    ↪ csv --feature_set 'o' --output_dir <
    ↪ output_directory>
```

A.7.10 Case Study

[*To check the raw results*] Each subfolder of `/home2/data1/J.Zhang/XGuardian/XGuardian/Experiments/5.7_Case_Study` contains the corresponding results.
[*To reproduce*] For elimination scope, please follow the procedures in Appendix A.6.2. For match scope, run the following command.

```
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Experiments/5.7_Case_Study/match_scope/
    ↪ shap_case_Study.py --train_path XGuardian/
    ↪ XGuardian/Experiments/5.2
    ↪ _Detection_Performace/overall_performance/
    ↪ TrainValTest/
    ↪ aggregated_predictions_train_cleaned.csv --
    ↪ dataset_path XGuardian/XGuardian/
    ↪ Experiments/5.2_Detection_Performace/
```

```
    ↪ overall_performance/TrainValTest/
    ↪ aggregated_predictions_test_cleaned.csv --
    ↪ model_path XGuardian/XGuardian/Experiments/
    ↪ 5.2_Detection_Performace/
    ↪ overall_performance/TrainValTest/
    ↪ logistic_regression_model_original.joblib
    ↪ --feature_set 'o' --output_dir <output_dir>
```

A.7.11 Robustness to Adversarial Attack

[*To check the raw results*] Use `/home2/data1/J.Zhang/XGuardian/XGuardian/Experiments/5.8_Adversarial_Attack/src/round_eval_metrics_stats.py` to calculate the metrics. The directory of the file `round_integ_test_predictions_original.csv` must be modified before the execution at Line 5.

```
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Experiments/5.8_Adversarial_Attack/src/
    ↪ round_eval_metrics_stats.py
```

[*To reproduce*] Run the following commands. Due to minimizing the size of our artifact, we deleted some intermediate files (e.g., `aggregated_prediction_x.csv`, etc.), you may reproduce it by using `/home2/data1/J.Zhang/XGuardian/XGuardian/Experiments/5.8_Adversarial_Attack/src/XGuardian-Inspector-Prediction-only.py`. Note that the directories of the models `/home2/data1/J.Zhang/XGuardian/XGuardian/Experiments/5.2_Detection_Performace/overall_performance/TrainValTest/gru_cnn_best_model.keras`, `/home2/data1/J.Zhang/XGuardian/XGuardian/Experiments/5.2_Detection_Performace/overall_performance/TrainValTest/aggregation_best_model.keras` and `/home2/data1/J.Zhang/XGuardian/XGuardian/Experiments/5.2_Detection_Performace/overall_performance/TrainValTest/best_aggregation_threshold_with_attention.txt` should be properly modified at Line 187-190, Line 256-260 and Line 263-268, respectively. Simply, you may instead copy these models to the `<output_dir>` directly.

```
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Experiments/5.8_Adversarial_Attack/src/
    ↪ XGuardian-Inspector-Prediction-only.py
    ↪ XGuardian/XGuardian/Experiments/5.8
    ↪ _Adversarial_Attack/data/
    ↪ adversarial_dataset.csv XGuardian/XGuardian
    ↪ /Experiments/5.8_Adversarial_Attack/result
    ↪ 6 1 "isCheater,steamid,kill_num,match_id,
    ↪ in_crouch,is_walking,is_strafing,weapon,
    ↪ angle,x,y,distance"
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Experiments/5.8_Adversarial_Attack/src/
    ↪ XGuardian-Inspector-Match-Preprocessing.py
    ↪ XGuardian/XGuardian/Experiments/5.8
    ↪ _Adversarial_Attack/result/
    ↪ aggregated_predictions_test.csv XGuardian/
    ↪ XGuardian/Experiments/5.8
    ↪ _Adversarial_Attack/result/
    ↪ aggregated_predictions_test_cleaned.csv
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Experiments/5.8_Adversarial_Attack/src/
    ↪ XGuardian-Inspector-Match-Prediction-only.
    ↪ py --test_path XGuardian/XGuardian/
    ↪ Experiments/5.8_Adversarial_Attack/result/
    ↪ aggregated_predictions_test_cleaned.csv --
    ↪ feature_set 'o' --model_path XGuardian/
    ↪ XGuardian/Experiments/5.8
    ↪ _Adversarial_Attack/result/
    ↪ logistic_regression_model_original.joblib
    ↪ --output_dir XGuardian/XGuardian/
    ↪ Experiments/5.8_Adversarial_Attack/result
```

[*Reuse of* INSPECTOR *without training*] Please note that the above three scripts can be used to reuse the pre-trained model from XGUARDIAN, solely for performing prediction functions.

A.7.12 Overhead

[*To check the raw results*] Please check `data preprocessing_overhead.log`, `initialmodeltraining_overhead.log` and `testing_overhead.log` at `/home2/data1/J.Zhang/XGuardian/XGuardian/Experiments/5.9_Overhead/overhead`.
[*To reproduce*] Run the following command with `detector_cleaned` environment.

```
$ python /home2/data1/J.Zhang/XGuardian/XGuardian/
    ↪ Experiments/5.9_Overhead/overhead/
    ↪ GRU_aggregated_2D_feature_selection.py
    ↪ XGuardian/XGuardian/Experiments/5.9
    ↪ _Overhead/overhead/Train_cleaned.csv
    ↪ XGuardian/XGuardian/Experiments/5.9
    ↪ _Overhead/overhead/Validation_cleaned.csv
    ↪ XGuardian/XGuardian/Experiments/5.9
    ↪ _Overhead/overhead/Test_cleaned.csv <
    ↪ output_dir> 6 1 "isCheater,steamid,kill_num
    ↪ ,match_id,in_crouch,is_walking,is_strafing,
    ↪ weapon,angle,x,y,distance"
```

A.7.13 User Study

The original demos are stored at `/home2/data1/J.Zhang/XGuardian/XGuardian/Experiments/5.10_User_Study_(addtional_evaluation)`. Please check our paper for the answer of the questionnaires and the operation procedures.

A.8 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/usenixsec2026/.