A Artifact Appendix

A.1 Abstract

The artifact discovers the vulnerability gap between manual models and automl models against various kinds of attacks (adversarial, poison, backdoor, extraction and membership) in image classification domain. It implements all datasets, models, and attacks used in our paper.

We expect the artifact could support the paper's claim that automl models are more vulnerable than manual models against various kinds of attacks, which could be explained by their small gradient variance.

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

- **Binary:** on pypi with any platform.
- **Model:** Our pretrained models are available on Zenodo (link). Follow the model path style `{model_dir}/image/{dataset}/{model}.pth` to place them in correct location.
- **Data set:** CIFAR10, CIFAR100 and ImageNet32. Use `--download` flag to download them automatically at first running. ImageNet32 requires manual set-up at their website due to legality.
- **Run-time environment:**

  At any platform (Windows and Ubuntu tested).

  'Pytorch' and 'torchvision' required. (CUDA 11.3 recommended)

  'adversarial-robustness-toolbox' required for extraction attack and membership attack.
- **Hardware:** GPU with CUDA support is recommended.
- **Execution:** Model training and backdoor attack would be time-consuming. It would cost more than half day on a Nvidia Quodro RTX6000.
- **Metrics:** Model accuracy, attack success rate, clean accuracy drop and cross entropy.
- **Output:** console output and saved model files (.pth).
- **Experiments:** OS scripts. Recommend to run scripts 3-5 times to reduce the randomness of experiments.
- **How much disk space required (approximately)?:** less than 5GB.
- **How much time is needed to prepare workflow (approximately)?:** within 1 hour.
- **How much time is needed to complete experiments (approximately)?:** 3-4 days.
- **Publicly available?:** on GitHub.
- **Code licenses (if publicly available)?:** GPL-3.
- **Archived (provide DOI)?:** GitHub commit be2ca9f1be212f31176c4898647f6f5444cee27e.

A.3 Description

A.3.1 How to access

- **GitHub:** pip install -e .
- **PYPI:** pip install autovul
- **Docker Hub:** docker pull local0state/autovul
- **GitHub Packages:** docker pull ghcr.io/ain-soph/autovul

A.3.2 Hardware dependencies

Recommend to use GPU with CUDA 11.3 and CUDNN 8.0. Less than 5GB disk space is needed.

A.3.3 Software dependencies

You need to install `python==3.9, pytorch==1.10.x, torchvision==0.11.x` manually.

ART (IBM) is required for extraction attack and membership attack. `pip install adversarial-robustness-toolbox`

A.3.4 Data sets

We use CIFAR10, CIFAR100 and ImageNet32 datasets. Use `--download` flag to download them automatically at first running. ImageNet32 requires manual set-up at their website due to legality.

A.3.5 Models

Our pretrained models are available on Zenodo (link). Follow the model path style `{model_dir}/image/{dataset}/{model}.pth` to place them in correct location.

A.4 Installation

- **GitHub:** pip install -e .
- **PYPI:** pip install autovul
- **Docker Hub:** docker pull local0state/autovul
- **GitHub Packages:** docker pull ghcr.io/ain-soph/autovul

(optional) Config Path

You can set the config files to customize data storage location and many other default settings. View `/configs_example` as an example config setting.

We support 3 configs (priority ascend):

- **package (DO NOT MODIFY)**
  - `autovul/base/configs/*.yml`
  - `autovul/vision/configs/*.yml`
- **user**
  - `~/.autovul/configs/base/*.yml`
  - `~/.autovul/configs/vision/*.yml`
- **workspace**
  - `./configs/base/*.yml`
  - `./configs/vision/*.yml`

A.5 Experiment workflow

Bash Files

Check the bash files under `/bash` to reproduce our paper results.

Train Models

You need to first run `/bash/train.sh` to get pretrained models.

If you run it for the first time, please run with `--download` flag to download the dataset:

`bash ./bash/train.sh "--download"`

It takes a relatively long time to train all models, here we provide our pretrained models on Zenodo (link). Follow the model path style `{model_dir}/image/{dataset}/{model}.pth` to place them in correct location.

Run Attacks

`/bash/adv_attack.sh`

`/bash/poison.sh`

`/bash/backdoor.sh`

`/bash/extraction.sh`

`/bash/membership.sh`

Run Other Exps

Gradient Variance

`/bash/grad_var.sh`

Mitigation Architecture

`/bash/mitigation_train.sh` (optional)

`/bash/mitigation_backdoor.sh`

`/bash/mitigation_extraction.sh`

Optionally, You can generate these architectures based on DARTS_V2 using `python ./projects/generate_mitigation.py`. We have already put the generated archs in `autovul.vision.utils.model_archs.darts.genotypes`. Note that we have provided the pretrained models for mitigation architectures on Google Drive as well.

For mitigation experiments, the architecture names in our paper map to:

- **darts-i:** `diy_deep`
- **darts-ii:** `diy_noskip`
- **darts-iii:** `diy_deep_noskip`

These are the 3 options for `--model_arch {arch}` (with `--model darts`)

To increase cell depth, we may re-wire existing models generated by NAS or modify the performance measure of candidate models. For the former case, we have provided the script to rewire a given model (link). Note that it is necessary to ensure the re-wiring doesn't cause a significant performance drop. For the latter case, we may increase the number of training steps in the single-step gradient descent used in DARTS.

To suppress skip connects, we replace the skip connects in a given model with other operations (e.g., convolution) or modify the likelihood of them being selected in the search process. Fro the former case, we have provided the script to substitute skip connects with convolution operations (link). Note that it is necessary to ensure the substitution doesn't cause a significant performance drop. For the latter case, we may multiply the weight of skip connect α_{skip} by a coefficient $\gamma \in (0,1)$.

Loss Contours

Take the parameter-space contour as an example. We pick the parameters of the first convolutional layer and randomly generate two orthogonal directions d_1 and d_2 in the parameter space. For simplicity, we set all each dimension of d_1 and d_2 to be either $+1$ or -1 in a random order and ensure that their orthogonality as $d_1 \cdot d_2 = 0$. We then follow Equation (12) in the paper to explore the mesh grid of $[-0.5, 0.5] \times [-0.5, 0.5]$ and plot the loss contour. A similar procedure is applied to plot the loss contour in the input space, but with the grid set as $[-0.2, 0.2] \times [-0.2, 0.2]$

A.6 Evaluation and expected results

Our paper claims that automl models are more vulnerable than manual models against various kinds of attacks, which could be explained by low gradient variance.

Training

(Table 1) Most models around 96%-97% accuracy on CIFAR10.

Attack

For automl models on CIFAR10,

- **adversarial:** (Figure 2) higher success rate around 10% (±4%).
- **poison:** (Figure 6) lower accuracy drop around 5% (±2%).
- **backdoor:** (Figure 7) higher success rate around 2% (±1%) and lower accuracy drop around 1% (±1%).
- **extraction:** (Figure 9) lower inference cross entropy around 0.3 (±0.1).
- **membership:** (Figure 10) higher auc around 0.04 (±0.01).

Others

- **gradient variance:** (Figure 12) automl with lower gradient variance around 2.2 (±0.5).
- **mitigation architecture:** (Table 4, Figure 16, 17) deep architectures (`darts-i, darts-iii`) have larger cross entropy for extraction attack around 0.5, and higher accuracy drop for poisoning attack around 7% (±3%) with setting of 40% poisoning fraction.

A.7 Experiment customization

Use `-h` or `--help` flag for example python files to check available arguments.

A.8 Notes

A.9 Methodology

Submission, reviewing and badging methodology:

- https://www.acm.org/publications/policies/artifact-review-badging
- http://cTuning.org/ae/submission-20201122.html
- http://cTuning.org/ae/reviewing-20201122.html