Coding Task 1: Membership Inference

Implement a membership inference attack and achieve the **highest True Positive Ratio and AUC ROC**.

- We give you a trained model (Resnet18) and a public dataset (custom dataset containing fields: ids, images, labels (class labels), membership (0 means this sample is a nonmember, 1 means is a member)), your task is to determine for each point yes/no, meaning: member/no-member.
- You also get a second, private dataset, with the membership field being filled with Nones. Your goal is to classify each sample as a member or nonmember, with a continuous score, not 1/0 (that's for the AUC ROC metric to make sense).
- The underlying training dataset is undisclosed.
- You get an example code for loading the data and submitting.

Task artifacts

- PRIVATE
- PUBLIC
- MODEL

Evaluation

Endpoint for this task is **http://149.156.182.9:6060/task-1/submit**. You can only submit a CSV with scores every hour. If your submission is malformed or wrong, you should get a comprehensive error message. However, even if you fail, you still get a 1h cooldown.

- A leaderboard on 30% of the samples from your submission will be available at any time on the scoreboard. It is an intermediate scoreboard; you can use it to evaluate your results and compare them to other teams while working on the solution.
- The final leaderboard on 100% of the samples will be revealed once the deadline for the assignment passes.

Coding Task 2: Model Stealing

Your task is to implement and execute a model stealing method against the protected Self-Supervised Learning (SSL) encoder protected behind an API. Your goal is to achieve the **lowest L2 distance** of output representations on private images between your stolen copy and the attacked encoder.

- You are provided with a subset of the training data of the victim model. The structure of the dataset is similar to the previous assignment.
- You can query the model using a dedicated API, details are below. The model is protected using B4B.
- Your submission will be an ONNX file containing the stolen model
- The scoreboard works the same as for the first assignment.

Task artifacts

- PUBLIC

API service

Teams will get access to the encoder via API. You can request a new API every 4 hours (**http://149.156.182.9:6060/task-2/reset**). Every time you'll get a different encoder to steal. The API expects up to 1000 images as an input and allows for one query every minute (**http://149.156.182.9:6060/task-2/query)**. You are limited to 100k images queried per API, after crossing that threshold you will not get anything from the API. This is to simulate the real-world scenario, where you want to use as few queries as possible to steal the encoder, as the queries are pricey.

Evaluation

The encoder you steal takes an image of 3x32x32 as an input and outputs a representation of size 1024 (after transformations). The evaluation endpoint (for this task, **http://149.156.182.9:6060/task-2/submit**) expects contestants to provide an ONNX version of your stolen copy. A couple of things can go wrong on your side:

- Incorrect input dimensionality expected by your model (has to be 3x32x32)
- Model not allowing batched input
- Incorrect name of the input (see example submission please)
- Incorrect output dimensionality expected by the evaluation endpoint (has to be 1024)

The evaluation procedure is as follows:

1. We load your model
2. We obtain representations of images from a private dataset
3. We calculate the L2 distance between the submitted model's representations and the victim model's representations

Same as in the first Assignment, you're limited to only one submission per hour, and the immediate result you get back is evaluated on 30% of the data, and the final (after the deadline) will be on 100% of the private data.

Coding Task 3: Robustness

Teams train a robust classifier against adversarial examples. Your goal is to achieve the **highest accuracy** for **clean**, as well as **adversarial examples** created using FGSM and PGD.

- You are provided with a training dataset
- Your submission will be an PyTorch *.pt file containing your model's state dict, as well as the model class name.
- The samples, on which your model will be evaluated, come from the same distribution as the provided dataset.

Task artifacts

- TRAINING

Submission

The evaluation endpoint (**http://149.156.182.9:6060/task-3/submit**) expects teams to provide a PyTorch *.pt file consisting of the model's state dict, as well as the model's class name (string). In order for us to be able to run the evaluation, we require them to provide one of the following classes (from torchvision.models):

- resnet18
- resnet34
- resnet50

Evaluation

The evaluation procedure is as follows:

1. We load your model, and run assertions on it.
2. We calculate clean accuracy on our private samples.
    a. **We require the clean accuracy to be above 50% for model's results to be accepted.**
3. We run adversarial attacks on the samples and compute accuracies of the perturbed data.

Very important note: your results will be overwritten with each submission. It's because robustness and clean performance are a trade-off game, and we want to consider that. Same as in the first Assignment, you're limited to only one submission per hour, and the immediate result you get back is evaluated on 30% of the data, and the final (after the deadline) will be on 100% of the private data.

Coding Task 4 (backdoors, backdoors everywhere)

You are provided with a resnet50 model for image classification that has been backdoored. Your task is to remove the backdoor vulnerability from the model, **without degrading the accuracy** of the final model.

Task artifacts

- RESNET50
- PUBLIC

Evaluation

The evaluation endpoint (**http://149.156.182.9:6060/task-4/submit**) expects you to provide a ONNX version of the purified Resnet50 model.

A couple of things can go wrong on your side:

- Incorrect input dimensionality expected by your model (has to be 3x64x64)
- Incorrect name of the input (see example submission please)
- Incorrect output dimensionality expected by the evaluation endpoint (has to be 1024)
- We provide you with an example submission, as well as the assertions on the side of the evaluation server regarding your submitted file. We encourage you to first run the assertions, and only then submit your model for evaluation.

The evaluation procedure is as follows:

1. We load your model
2. We calculate the accuracy of your final model on a hidden evaluation dataset.
3. We calculate the backdoor score as 1- average attack success rate (ASR) on a hidden evaluation dataset.
4. Final score is calculated as 0.5 * ACC + 0.5 * backdoor score.

Same as in the first Assignment, you're limited to only one submission per hour, and the immediate result you get back is evaluated on 30% of the data, and the final (after the deadline) will be on 70% of the private data.

Scoreboard:

The board with each team’s current scores is available here: http://138.68.67.242:3000/