Trustworthy Machine Learning, SS 2025
Prof. Dr. Adam Dziedzic and Prof. Dr. Franziska Boenisch
Assignment Sheet #2: Model Stealing

Teaching Assistant: Nupur Kulkarni

Deadline: 25th of June

Task: Implement and execute a model stealing method against the protected encoder provided with a mockup API.

Model Stealing is when an attacker uses an API to send queries to a victim model behind the API and utilizes the outputs (labels or embeddings) to train a replica of the victim model. In this assignment, you are yet again in the *attacker's shoes*. You are provided with a trained encoder of unknown architecture hidden behind an API and protected using B4B defense. Your goal is to achieve the ***lowest L2 distance*** of output representations on private images between your stolen copy and the attacked encoder. Before you steal the model, it will be good to revise how model stealing attacks and defenses work from lectures - Model Stealing and Defenses for Supervised Learning and Model Stealing and Defenses for Self-Supervised learning

As an attacker, you have access to:

- Trained encoder of unknown architecture hidden behind an API and protected using B4B noising (see paper and lecture to understand how B4B works)
- Subset of encoder's training data (MODEL_STEALING_PUB). The structure of the dataset is similar to the previous assignment, you can reuse the code from assignment 1. Note that the normalized parameters are also the same as in assignment 1.
- An API to send queries to. The API can be launched on your local machine.

We also provide you with an example code as a starting point for the assignment. It will help you execute the steps below seamlessly.

Steps to execute this assignment:

- **Requesting the API** - You need access to the encoder, so first things first. You need to first access the API, as you get access to the encoder via the API. To do that, request the API using the following code.

```
### REQUESTING NEW API ###
TOKEN = "insert your token here"

response = requests.get("http://34.122.51.94:9090" + "/stealing_launch",
headers={"token": TOKEN})
answer = response.json()

print(answer)
if 'detail' in answer:
    sys.exit(1)

# save the values
SEED = str(answer['seed'])
PORT = str(answer['port'])
```

Important things to note regarding the API -

1. You can request a new API every 4 hours, everytime you will get a new encoder to steal.
2. Note the seed you get back from the code above, as you will need it for submission. If you steal the encoder perfectly, but provide the wrong seed, your L2 will be on par with a random submission. Additionally, note the port you obtain, as you'll need it to perform the next step.

- **Querying the API** - Great that you have an API now, so you can send queries to the encoder hidden behind it.

```
def model_stealing(images, port):
    endpoint = "/query"
    url = f"http://34.122.51.94:{port}" + endpoint
    image_data = []
    for img in images:
        img_byte_arr = io.BytesIO()
        img.save(img_byte_arr, format='PNG')
        img_byte_arr.seek(0)
        img_base64 =
base64.b64encode(img_byte_arr.getvalue()).decode('utf-8')
        image_data.append(img_base64)

    payload = json.dumps(image_data)
    response = requests.get(url, files={"file": payload}, headers={"token":
"insert your token here"})
    if response.status_code == 200:
        representation = response.json()["representations"]
        return representation
    else:
        raise Exception(
            f"Model stealing failed. Code: {response.status_code}, content:
{response.json()}"
        )

out = model_stealing([dataset.imgs[idx] for idx in
np.random.permutation(1000)], port="insert port obtained from code snippet
in previous step")
```

Important things to note regarding queries to the API:

1. The API expects **1000 images** as input and allows for **one query**. To simplify, when you send 1 query to the API, you're sending 1000 images in that query.
2. The encoder you steal takes an image of 3x32x32 as input and outputs a representation of size 1024 (after transformations). If you wonder what transformations are in this context, go back to the paper and lecture.
3. You are limited to 100k queries 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.

- **Stealing the model** - This is where your innovative ideas come in as an attacker. Now that you have the output representations obtained by querying the encoder, analyze them and train a new encoder (i.e., the stolen copy of the victim encoder). Do not forget that the victim encoder is protected by B4B defense!

  Ready to check how close your stolen encoder is to that of the victim encoder? It is time to discuss how to submit and the details of the evaluation process.

How to submit your stolen copy?

1. The evaluation endpoint (for this task, http://34.122.51.94:9090/stealing) expects you to provide an **ONNX version of your stolen copy** (do not forget to install onnx and onnxruntime libraries), the **token provided to you** already via email (same that was used for the first assignment), as well as the **seed you see after launching your local API**. The *seed is necessary*, and if you send an incorrect one to the evaluation API, expect your L2 error to be extremely high.
2. Refer to this code to make an example submission.

What can go wrong during submission?

A couple of things can go wrong on your side:
1. Incorrect input dimensionality expected by your model (has to be 3x32x32)
2. Incorrect name of the input (see example submission, please)
3. Incorrect output dimensionality expected by the evaluation endpoint (has to be 1024)
4. Wrong seed (both the format and the value)

To help you out, we also provide you with assertions in this code on the side of the evaluation server regarding your submitted file. We encourage you first to run the assertions and only then submit your model for evaluation.

How do we evaluate your submission?

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.

Scoreboard

- You can access the scoreboard for this task here: http://34.122.51.94:9090/score_board, located in the *Model Stealing* tab. This will help you to compare your solutions with other teams and see where you stand.
- Just like the first assignment, this scoreboard only shows the results on 30% of the samples from your submission, so it is an intermediate scoreboard.
- The evaluation is split into two sets: immediate and final (30:70). The score you will see before the deadline is for the immediate set; the final (now hidden) one will be revealed once the deadline for the assignment passes.

Read the following instructions carefully

- Create a GitHub repository TML25_A2_YourTeamNumber. For example, if you are in Team 13, your repository should be named TML25_A2_13. *The team number here should be the one assigned to you via email and not the one you see in CMS.*
- Upload your code files to the repository.
- The documentation for the assignment is split into a **README and a report, and both should be submitted**. The final grade will heavily depend on the documentation.
- Add a README.md to your GitHub repository that points us to the most important files and pieces of code.
- Add a report (PDF format) to your GitHub repository that describes your solution.
- **Submit a zip file (from your GitHub repository) with your readme, report, and the whole repository (only the important files, please avoid including the large models and artifacts in your repository).**
- You are only supposed to make *one submission* per group.