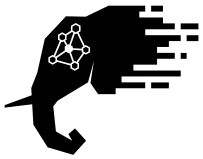

Task 1: Model Stealing

You applied for a summer internship at a well-known Micro Innovations and Nonsense Inc. (MiNI) enterprise. As a smart and promising AI student, you were assigned to the red team to test the security of their system. In short, your daily routine from now on is to compromise system security measures for the Beta version of the MiNI system.

Today after the morning coffee you open your email box and find an email from Mike (Red Team Leader):

> *Hi XYZ,*
>
> *Our company trained a foundation encoder model on a huge and valuable dataset. It creates useful high-dimensional representations of various images that can be later used in various specific downstream tasks. We want to share the model through API endpoints, charging the users for each query. Querying means that users send their images, and obtain representations from our encoder.*
>
> *We know that our competition is interested in that model and would probably try to steal it as soon as it is available online. Using specific queries to our API they can use outputs to train a copy of our model multiple times cheaper.*
>
> *That is why we have implemented several countermeasures to prevent this scenario. A common protection method that works well for classifiers relies on simply adding some noise to the API output. Of course, we could not use this method for our state-of-the-art encoder model. What good would it be if it returned low-quality representations to our API users? We would quickly fall out of business. Instead, we employ a novel adaptive noising technique that adjusts the representation quality depending on the user behavior. In this way,*

our clients, who use our API encoder to solve downstream tasks as intended, and experience no utility drop. However, the attackers querying the model with diverse data to copy its capabilities will quickly receive just garbage and useless responses.

However, in this field, it pays to be extra sure. That is when you come in as the red team. Your task is to test the safety of our API by trying to steal it despite the protective adaptive nosing mechanism. We are interested to see how closely the stolen copy that you obtain will match the outputs that our state-of-the art encoder can produce.

GL,

Mike

Endpoint

We provide an endpoint for the encoder model. These endpoints receive as an input query the image in the PNG format, and your team's API key. Every endpoint returns the representation of the image from an SSL vision encoder. The encoder is protected by an adaptive noising mechanism that adds noise to the representations depending on the diversity of queries previously sent to the API. The process is gradual.

An example query is provided here.

Resetting

There is a possibility to reset the API endpoint and the noising mechanism. However, this can only be done a limited number of times (10 per team). After each restart you get a new encoder to steal, therefore, all your previous representations become useless. We suggest saving the representations returned by the API locally.

The boilerplate code of doing exactly that is provided here.

Dataset

You have access to ModelStealingPub dataset: image_id (int), image (PIL.Image.Image), image_label (int). This is a small subset of the encoders' training set.

See boilerplate code for loading here.

Submission

The submission should be an ONNX model checkpoint. The model should receive images (of the same resolution as the victim model, i.e, **3×32×32**) as inputs and return representations (of the same dimension as the victim model, i.e, **512**).

Example submission code is provided here.

Evaluation

Your submissions will be evaluated in the following way: we pass a secret data subset through the victim model and your submitted model to receive representations. We calculate the **mean L2 loss** between the victim and your model. The lower the distance the better.

Tips

- Querying the victim model with more input data increases the noise added by adaptive nosing. However, you can do with the input data whatever you want before passing it to the stolen model. Consider some clever strategies like data augmentations to leverage this fact
- You can use any loss function between the victim and stolen copy outputs. Some loss formulations like contrastive learning make better use of the data than others.
- Use any architecture of the local model that you want