Fall 2024 CS53331/4331 Adversarial Machine Learning Assignment 2

Title	**The Certification and privacy attack stuff**
Due date	November 8th 6:00 pm
First Name	Sruthi
Last Name	Mandalapu
Student ID	R11906160
Marks	100

Note: Please answer the following questions and submit them through Blackboard. Be sure to submit it to assignment 1. DO NOT write the report by hand and submit a scanned version. Just write the answers in a Word document and submit it. Both Word and PDF submissions are accepted.

Submission Instruction (3 documents)

You are required to submit three documents:

1. ***Report.*** Just fill out the above report and submit it as a Word or PDF document.
2. ***Ipynb file.*** The code that you have written. Preferably in an ipynb document. You can submit it as a .py file as well.
3. ***Txt file of the code.*** We need your code in the .txt file as well. Use whatever way you prefer. The fastest would be to download the file as a .py file and change the extension to .txt

Objectives

This assignment has three main objectives:

Part 1

1. Continuing on Assignment 1 objectives, the first objective in this assignment will be to certify the build VGG from the last assignment using Randomized smoothing.

Part 2

2. Implement privacy attack (label one and shadow modeling)

Part 3

3. Use DP as a defense

Get started

Download the assignment files from Blackboard. You will need the report (This file), the .ipynb file where you will put your code, the dataset, and the given model (if any). The GTSRB dataset will be used for both parts of this assignment.

Dataset

The German Traffic Sign Recognition Benchmark (GTSRB) dataset consists of almost 51K images of traffic signs. There are 43 classes, and the size of their images is 32×32 pixels. Some of the images are shown below. Please download the dataset from Blackboard or here. More information about the dataset can be found here.

As we did with earlier assignments, it is recommended that you upload the dataset into your personal Google Drive to follow the Colab instructions as they are. Of course, if you prefer to use other than Colab, you will need a similar preprocessing.

Instruction for Colab (repeated from Assignment 0)

To get started with Google Colab, simply go to Google Colab, sign in with your Google account, and create a new notebook. You can write and execute Python code directly in the notebook. To access your dataset stored in your Google Drive (previous step), first run the following code to mount your Drive:

```
from google.colab import drive
drive.mount('/content/drive')
```

Follow the authorization steps, and your Drive will be accessible at **/content/drive/My Drive/.** You can then load your dataset into the notebook by providing the correct file path. This part of the code is provided for you in the .ipynb file of Assignment 0. You will need to setup the drive connection and run the code.

To use the free GPU provided by Colab, you can change the runtime to access a GPU by clicking on **"Runtime" > "Change runtime type" and** selecting **"T4 GPU"** from the **Hardware accelerator** dropdown menu. You can always use higher GPU powers at a cost (Colab Pro is $10 per month), but you should be fine with the free version, considering that you start the assignment early enough.

Colab comes with many pre-installed libraries, but if you need to install additional Python packages, you can do so with pip. For example:

```
!pip install library_name
```

Remember to save your work frequently.

After you've completed your work in Google Colab, you can easily download your notebook from Google Colab, go to **"File" > "Download" > "Download.ipynb"**.

Other than Colab

If you don't prefer Colab or notebook, you always have the option to run it on your computer (especially if it has a GPU) or access HPCC resources at TTU (needs an account with my permission).

Additional resources

1. TensorFlow resource https://www.tensorflow.org/
2. PyTorch resources https://pytorch.org/get-started/pytorch-2.0/
3. Deep learning with Python https://dl-with-python.readthedocs.io/en/latest/
4. Get started with Colab https://colab.research.google.com/

Part1 Certified Training

Task 1: Randomized Smoothing (30 pts)

Our first task with implement randomized smoothing for the VGG model from last time. Your task is to download the model and use Adversarial Robustness Toolbox (ART) to implement randomized smoothing to certify the model. This notebook should be your guide to implementing the task. You will use $\sigma^2 = [\,0.25, 0.5, 1]$ as Gaussian noise parameter. Once needed, use 100 as the number of samples to estimate the certifiable radius. You will find the standard and certified accuracy for different noises. You will also plot the L2 radius versus certified accuracy.

Notes: Training will take time to generate the results. Try it first with only 2 training rounds and then increase to 25. Certification for all the testing will take forever. Thus, whenever you need a testing set, use a subset of the testing samples, not all. We already provided that subset for you in a variable that you must look for ☺.

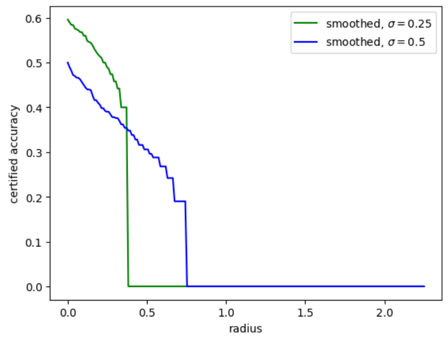
1. [10 pts] Implement what is required.

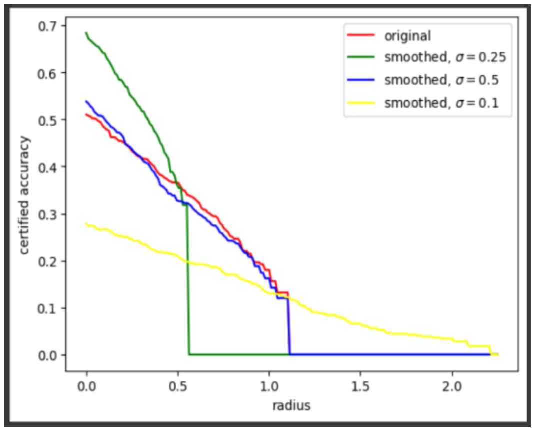
2. [10 pts] Fill in Table 1 with the required values

Table 1: Standard and Certified Accuracies for different models

Model	**L2 Radius**	**Standard accuracy**	**Standard coverage rate**	**Certified accuracy**
VGG16 original	40.2%	75.57%	69.6%	16.3%
Smoothed $\sigma^2 = 0.25$	29.73%	88.20%	78%	13.00%
Smoothed $\sigma^2 = 0.5$	36.88%	83.38%	68.6%	15.99%
Smoothed $\sigma^2 = 1$	44.34%	45.31%	66.2%	12.61%

3. [5 pts] Plot L2 radius (from 0 to 3) vs certified accuracy for the three models (original, smoothed 0.25, and smoothed 5) in one figure. An example of some smoothed certified accuracies is below.





- yellow color for smoothing at 1 (wrongly printed)

4. [5 pts] Briefly provide insights on the results.
   As we can observe the standard accuracy for original classifier is relatively high at around 75.57%, but the certified accuracy is much lower (16.3%). This shows that the model is vulnerable to adversarial attacks, and its robustness cannot be guaranteed without defense mechanisms. The addition of noise via randomized smoothing helps in certifying the model's robustness, but it comes at the cost of lowering the certified accuracy as the noise level increases. This is because higher noise (higher sigma) results in a larger radius but less certainty in classification, leading to a drop in certified accuracy. And also, we can observe the smoothed models with added noise (sigma = 0.25, 0.5, and 1) show increased standard accuracy but with decreased certified accuracy. As the noise parameter increases, the L2 radius increases, and the certified accuracy decreases. For example, the smoothed model with sigma = 0.25 has a relatively high standard accuracy but the lowest certified accuracy of 13.00%.

Part 2 Privacy Attacks

Task 2-Label-only attack (25 pts)

Now, let's do some **privacy attacks**. We will focus on Membership inference attacks. This task will implement label-only attacks, while the next will implement shadow modeling. As discussed in class, the label-only attack is developed based on the fact that it costs an attacker so much to generate a successful adversarial sample for a member sample. In contrast, the cost will be much less for a non-member sample. Thus, it uses HopSkipJump to generate a successful adversarial sample and then measure the perturbation using the L2 norm. If the perturbation is above a threshold, it will be classified as a member; if not, it will be classified as a non-member. Hence, we will follow the following steps to perform the attack (Your reference code is this notebook):

1- Generate a dataset (already given): The training and testing datasets are merged into one, with the training classified as members and the testing as non-members. We picked 1500 samples from this dataset to build our attack model.

2- Calculate the threshold (to be implemented by you): Your task here is to use existing implementations in ART to calculate/calibrate the threshold. All parameters are given to you in the code (in comments).

3- Infer the data: Use the 1500 provided samples to predict whether they are members. Then, compare your performance to the ground truth and present the results.

One parameter you will need to set is the maximum number of queries (used in step 2). Set this parameter to 5 and 10. All that is needed is to answer the questions below.

Note on resources: It takes time to generate the adversarial examples. A good GPU computer will take you around 1-3 hours per round. Thus, plan your time accordingly.

1. [10 pts] Implement what is required.

2. [10 pts] Fill out Table 2.

Table 2: Attack accuracy with different number of queries.

Max number of queries	**Members accuracy**	**Non-member accuracy**	**Overall accuracy**
5	88.75%	15.27%	68.86%
10	89.85%	13.30%	69.13%

5. [5 pts] Briefly provide insights on the results.
The model is effective in identifying members in the dataset, with members accuracy above 88% for both 5 and 10 queries. As non-members accuracies are low, attack struggles to correctly classify non-members, with accuracy significantly lower (around 15% for 5 queries and 13% for 10 queries). This indicates that the attack is more confident in identifying members than non-members. Increasing the number of queries slightly improves the overall accuracy, as the attack gets more chances to generate adversarial examples and refine its classification. Overall, this attack identifies the sample with high accuracy if sample is a member or not.

Task 3- Shadow Models attacks (25 pts)

Let's do another privacy attack. This time, it is a "shadow models" attack, discussed in class. This notebook will be your guide. You are going to use ART again, and here are a few steps to follow:

1- Build the shadow models based on the numbers provided below.
2- Measure the accuracies for them (use validation accuracy as a measure)
3- Build the attack model. You can build any binary classifier. You are suggested to use a random forest classifier for simplicity.
4- Evaluate your attack models for members and non-members (the evaluation data is already given to you).
5- Calculate the precision and recall of your attack model.

Your changing parameter in this task is the number of shadow models. You will set it to 1,2 and 5. Please note the needed metrics so you can plan your code accordingly.

Note on the implementation: Our implementation will look strange. Shadow models will require lots and lots of data. Thus, in a general setup, researchers use 80% of the data for shadow models, and only 20% is for training/testing/validation. However, given that we are using a model already built from Assignment 1, we cannot use this general setup. Thus, our shadow models' data will concatenate training and testing while the validation will be left out to test performance accuracy. To evaluate the attack, we generated a separate dataset that is taken from training and testing. You will need to locate and work with that data. In general, your attack results won't be good due to the above design, but the idea is that you should analyze it.

1. [10 pts] Implement what is required.

2. [10 pts] Fill out Table 3.

Table 3: Shadow model performance and attack performance

Metric	1 shadow model	2 shadow models	5 shadow models
Average Accuracy (using validation)	55.98%	53.98%	55.19%
Members accuracy	93.41%	92.99%	92.98%
Non-members accuracy	79.28%	80.47%	81.57%
Overall accuracy	89.65%	89.65%	89.94%
Attack precision	92.55%	92.91%	93.29%
Attack recall	93.41%	92.99%	92.98%

3. [5 pts] Briefly provide insights on the results.
   In general scenario shadow models allow attackers to simulate the behavior of the target model and perform membership inference attacks. With more shadow models, the attacker can better approximate the behavior of the target model, leading to more accurate attack results. As we can observe on increase in shadow models there is quite increase in Non-members accuracy. As the number of shadow models increases, the attack model gets better at distinguishing between members and non-members. This is reflected in the improvement in attack precision from 1 to 5 shadow models. While 1 shadow model offers reasonable performance, 2 and 5 shadow models show a slight improvement in non-members accuracy and overall attack performance.

Part 3

Task 4- Deferentially private model performance (20 pts)

Now, let’s defend test defense with DP. We have built a DP-SGD model using the TensorFlow privacy library. The model is given to you, and you just need to analyze it. First, check the overall DP code and answer the below questions. Then, analyze the given DP model in terms of accuracy and resiliency to membership inference attacks. Use a label-only membership inference attack with 5 max-queries and the same setup as in Task 2.

1. [5 pts] Answer the following questions about the provided model or code.
    a. What is the value of the L2 norm used to build the model?
       L2 norm – 2
    b. What is the value of the noise added?
       Noise – 1.6
    c. What is the batch size and the micro-batch size to build the model?
       Batch Size – 128
       Micro Batch Size – 128
    d. Modify the commented code to enhance your accuracy performance (you can accept less privacy in this case). You don’t need to run the code; just tell us about the changes and maybe paste your code here.
       from_logits=True (put True instead of False)

2. [5 pts] Measure the differential privacy guarantee of the given model.
   Differential privacy (DP) guarantee for a machine learning model using DP-SGD. It computes the privacy parameters epsilon and delta, where epsilon indicates the privacy level (lower is more private) and delta represents the failure probability. The code uses training sample size, batch size, noise multiplier, epochs, and delta to estimate how much information could be leaked during training, printing the privacy guarantee in the form of (epsilon, delta). The differential privacy guarantee is explained in the code given as: (2.56, 1e-05).

3. [5 pts] Calculate the model’s performance under attacks and any other observations regarding the results.

Table 4: DP vs original performance

Model	**Accuracy**	**Members accuracy**	**Non-member accuracy**	**Overall accuracy**
VGG16 (Original)	90.12%	88.75%	15.27%	68.86%
DP	45.38%	31.79%	71.21%	42.20%

4. [5 pts] Briefly explain why DP accuracy is lower than VGG original. Provide insights on the results.
   DP is a defense mechanism which avoids the data leakage, even when attacker tries to infer the data. As we can observe, the accuracy levels when applying the defense are lower - that explains the differential privacy is effectively working disallowing the attacker to maintain

data privacy. In Task 2 Membership Inference Attack, this allows a normal VGG classifier allowing to determine if specific data samples were part of the model's training set (members) or not (non-members) by analyzing distances to decision boundaries. Where as, DP adds noise to training process which reduces the sensitivity of the model to individual data points. This means that even if an attacker tries to infer whether a specific record was in the training set, the model's responses won’t reveal much about any one record. The noise ensures that the model's output is statistically similar whether or not a particular data point is included in the training set. As a result, DP provides formal privacy guarantees, limiting the information leakage about individual training samples. This makes it harder for attackers to distinguish between members and non-members effectively.

Submission Instruction (3 documents)

You are required to submit three documents:

1. ***Report.*** Just fill out the above report and submit it as a Word or PDF document.
2. ***Ipynb file.*** The code that you have written. Preferably in an ipynb document. You can submit it as a .py file as well.
3. ***Txt file of the code.*** We need your code in the .txt file as well. Use whatever way you prefer. The fastest would be to download the file as a .py file and change the extension to .txt