Security & Privacy of Machine Learning — Assignment 1 Write-Up

Daan Brugmans (S1080742)

March 17, 2024

My Jupyter notebook and report can also be found publicly on GitHub with the following URL: `https://github.com/daanbrugmans/ru-security-and-privacy-of-machine-learning-23-24/tree/main/assignment-1`

My fulfillment of the assignment consists of two parts: the setup, which is the code I wrote in order to perform all the experiments that I need to do that will answer the assignment questions, and the assignment questions themselves. I will adhere to this structure for this write-up as well.

1 Setup

1.1 Imports, Preparation, and Data

I use PyTorch as the main environment for building and training a deep learning model, since that is the deep learning framework I am most comfortable with. To this end, I use the *torch*, *torchvision*, and *torchattacks* packages. Prior to training, I execute some preparatory code. I set a seed for *torch*, *NumPy*, and Python itself for improved reproducibility, and I set the device on which we I perform the model training to speed up the process when possible. I load the CIFAR-10 dataset as three *DataLoader* objects for performance and convenience.

1.2 Model

This part of my setup code consists of two class definitions. The first class is called *CIFAR10NeuralNet*. It is a PyTorch convolutional network. It consists of two convolution blocks, followed by three linear blocks, and finally a Softmax block. This is the architecture of the models that we train on the CIFAR-10 dataset. My general approach for this architecture is to upsample the number of output channels significantly during convolution, so that the model can learn varying features from the different channels, and then to iteratively decrease the number of neurons in the linear layers, so that the most important and general features from the channels can be extracted. The second class is called *NeuralModel*. This class is a collection of all processes and objects that are needed for training a *CIFAR10NeuralNet*. It contains an instance of the *CIFAR10NeuralNet*, its loss function, and its optimizer. It also contains functions for training and testing the *CIFAR10NeuralNet*, both using regular clean data, as wel as adversarial training. Finally, it contains a function that can be used to plot the train/validation loss and accuracy for the most recent training run. I have chosen to implement it this way, so that all code related to the neural network and its architecture is encapsulated within a single class. In my opinion, this makes performing varying attacks very clean: with only a few rows of code, I am able to instantiate and train a new model. This makes the experiments easy to read and hides away set implementation details.

1.3 Attacks

Finally, I define the attacks that I perform on the neural network. I define three attack models: the Fast Gradient Sign Method (FGSM), the Gradient Descent Projection (GPD), and the Auto-GPD (AGPD) attacks. The implementations of these models are directly taken from the *torchattacks* library. They are included here, because I wanted to make slight alterations to the standard implementation of *torchattacks*, so that all attacks can be performed both targeted as well as untargeted. The class descriptions for every attack model contains a direct URL that will guide one towards the original implementation.

2 Assignment Questions

For all assignment questions/experiments, I perform attacks with an unchanging set of hyperparameters. These may be found in table 1

Table 1: