Lab 7: Pure Black-Box Evasion Attack

CSC 592: Machine Learning Security and Privacy

**Background**

Many different black-box evasion attacks have been developed in the field of adversarial machine learning. White-box evasion attacks like FGSM, PGD and APGD all share a common threat model. In contrast, black-box attacks like the pure black-box attack and boundary attack have different adversarial threat models, as demonstrated in lecture. Understanding black-box attacks are important as they represent an attacker with less capabilities, which generally translates to a more realistic attack.

In this lab you will work with the most basic black-box transfer attack, the pure black-box attack. Reading the papers are not required for the lab assignment but the following resources may be of interest:

A brief description of the pure black-box attack can be found in this paper: <https://www.mdpi.com/1099-4300/23/10/1359>

A literature survey of different black-box attacks: <https://arxiv.org/abs/2109.15031>

**Lab Assignment**

The pseudocode for the pure black-box attack is given as follows:

1. Load a target model and a trained synthetic model .

2. Find a set which is 1000 clean classwise balanced samples that model recognizes correctly.

3. Use the set , synthetic model and white-box attack PGD to generate the adversarial set .

4. Have model evaluate to see how many examples it can correctly classify.

**Skeleton code, hyperparameters and hints:**

* For attack parameters, use the PGD method given in, “AttackWrappersWhiteBox.py” with 10 steps, and . Use 100 samples for the attack.
* For (the model under black-box attack, whose gradients you cannot access) use ResNet-56. The model is available on Brightspace.
* For (the synthetic model, whose gradients you CAN use) use ResNet-164. The model is available on Brightspace.
* The skeleton code for starting the pure black-box attack is given below. Step 1 is already done for you (loading target model and a trained synthetic model ), you just need to change the file paths to point to where you saved the models. You then need to code steps 2 through 4.

import torch

import DataManagerPytorchL as DMP

import ResNet

def main():

#Replace the next line with the file path of where you saved the ResNet models

modelDir = "C://Users//Kaleel//Desktop//Machine Learning Security and Privacy//Models//ModelResNet56-Run0.th"

modelDirSynthetic = "C://Users//Kaleel//Desktop//Machine Learning Security and Privacy//Models//ModelResNet164-Run0.th"

#Define the GPU device we are using

device = torch.device('cuda')

#Parameters for the dataset

batchSize = 64

numClasses = 10

inputImageSize = [1, 3, 32, 32] #Batch size, color channel, height, width

#Create the ResNet model

model = ResNet.resnet56(inputImageSize, numClasses).to(device)

#Next load in the trained weights of the model

checkpoint = torch.load(modelDir)

model.load\_state\_dict(checkpoint['state\_dict'])

#Switch the model into eval model for testing

model = model.eval()

#Load the synthetic model for the pure black-box attack

syntheticModel = ResNet.resnet164(inputImageSize, numClasses).to(device)

checkpointSynthetic = torch.load(modelDirSynthetic)

syntheticModel.load\_state\_dict(checkpointSynthetic['state\_dict'])

syntheticModel = syntheticModel.eval()

#Load in the dataset

valLoader = DMP.GetCIFAR10Validation(inputImageSize[2], batchSize)

#Check the clean accuracy of the model

cleanAcc = DMP.validateD(valLoader, model, device)

print("CIFAR-10 Clean Val Loader Acc:", cleanAcc)

**Exercise 1:** After coding the pure black-box attack, run the pure black-box attack on ResNet-56 and record a screenshot of your code running and the resulting robustness (accuracy). Your result should appear as shown below if you have correctly coded the attack:

**A screenshot of a computer

AI-generated content may be incorrect.**

**Deliverables**

Submit the following two items on Brightspace:

Deliverable #1: A document containing the answers to exercises.

Deliverable #2: A copy of your code (the .py files).

**Extra Credit (optional)**

The current robustness is 15% for the pure black-box attack with 100 samples. Can you modify the pure black-box attack or run a different black-box attack to decrease the robustness? You cannot change the following parameters: , the number of samples in the attack, or expand the threat model (no using the gradients from ResNet-56). I.e., you must keep the same maximum perturbation and cannot perform

any white-box attacks on the model .

Submit your extra code separately, a screenshot showing improved attack performance and a one paragraph description explaining how you decreased the robustness.

10 additional points on the midterm exam if successfully completed. No partial credit will be awarded for the extra credit (either it is correct or wrong).