Programming Assignment 1: AI Security

CS 419, Rutgers, Spring 2021
Shiqing Ma

January 31, 2021

General helpful tools and resources:

Models: https://github.com/kuangliu/pytorch-cifar

Model conversion tools: https://github.com/microsoft/MMdnn

Problem 1: Understanding Attacks

(Easy) We have learned how to perform a white-box attack against Deep Neural Networks. In this task, you are required to try to compare different attacks and their effects. We will be working on the ImageNet dataset.

Task:

Use FoolBox to implement at least 10 different attacks on the ResNet-18 model, and compare their effects including 1) attack success rate; 2) different p-norm distances when p = 0, 1, 2, inf; 3) time cost of each attack.

Attack settings: you can use the default attack parameters. Make sure you compare them under the same constraints, i.e., targeting the same target label or untarget attack for all of them. Please clearly describe your settings in your notebook.

Useful materials:

To get started, here is an existing example: https://colab.research.google.com/github/jonasrauber/foolbox-native-tutorial/blob/master/foolbox-native-tutorial.ipynb#scrollTo=Whh9Yqh7D2au

DeepFool: https://foolbox.jonasrauber.de

Artifacts:

The provided example already contains an implementation of multiple attacks. You will need to expand it to some other attacks, and measure the time cost, p-norm values (DeepFool has existing methods you can use) and also attack success rate.

You are required to submit a Jupyter Notebook that is runnable on Google Colab with detailed explanation of your attack and results. The name of the notebook should be "problem1.ipynb".

Grading:

Total: 2 points. As long as you have runnable code with reproducible results, you can get all points. If code does not run or has errors, you get 0. If missing one result: -0.1 for each of the incomplete one.

Problem 2: Attack that Transfers

(Challenging) Traditional attacks aim to attack a single model. As we know, adversarial examples do transfer, which is the foundation of many black box attacks. How about white box attacks then?

Task:

Please implement one FGSM based attack and improve it to generate adversarial examples that work on multiple models including ResNet-18, MobileNet, VGG-19 and AlexNet. We will be using the ImageNet dataset. In your report, please document your attack design and effects including 1) attack success rates for different models; 2) average p-norm distances for 10 inputs; 3) time cost of your attack.

During development, you can choose your own image to work on. During testing, we will replace them with our own image. Please make it easy for TAs to change the image paths.

Useful materials:

Pre-trained models: https://github.com/onnx/models#vision

Model conversion tools: https://github.com/microsoft/MMdnn

Artifacts:

A full FGSM attack implementation can be found at: https://github.com/soumyac1999/FGSM-Keras/blob/master/targeted_attack.ipynb

Notice that you can directly use it to generate adversarial examples. However, these adversarial examples usually cannot successfully attack all four models. Your job is to

solve this problem and generate adversarial examples that can successfully attack all these four models.

You are required to submit a Jupyter Notebook that is runnable on Google Colab with detailed explanation of your attack and results. (File name: “problem2.ipynb”)

Grading:

Total: 3 points. If you can attack all 4 models on all 10 images, you can get all points. If code does not run or has errors, you get 0. If one image fails to attack more than 2 models (namely, 3 or 4 out of 4), -0.2. If one image fails to attack 1 model, -0.1.

Problem 3: Black-box Attack (Bonus)

(Hard) Google Could Vision API provides ML services for image classification. It is a black box model and limit free users to try less than 1000 units/month. Can you attack it?

Task:

Try to implement a black box attack that can fool Google Cloud Vision API. It is well known that if you completely change all pixels, the attack will be a success. But we want to maintain a small perturbation. Try to have the least perturbations if possible and try to use as less queries as possible.

Useful materials:

Google Cloud Vision API: https://cloud.google.com/vision

SimBA attack: https://arxiv.org/pdf/1905.07121.pdf

SimBA attack implementation: https://github.com/cg563/simple-blackbox-attack

Artifacts:

You are required to submit a Jupyter Notebook that is runnable on Google Colab with detailed explanation of your attack and results. (File name: “problem3.ipynb”)

Grading:

Total: 3 points. If it is functioning, that is already awesome, and you can get 2 points. If you can limit the queries to less than 500, you can another extra point.

Problem 4: Understanding backdoors

(Easy) DNNs are easily poisoned. Let us try to create our own triggers.

Task:

Poisoning the CIFAR-10 dataset to create a ResNet-18 model with backdoors. You have the freedom to choose your own trigger. Evaluate this poisoned model on benign dataset and poisoned dataset to see its accuracy and attack success rate.

Notice that CIFAR images are relatively small (32x32x3), and this is not your concern. Your trigger cannot be too big, and it has to be smaller than 6x6.

There is no requirement of number of poisoned samples. It is related to the trigger you choose — if the trigger is simple, you may poison a small portion; otherwise, you can poison more. Tune the parameters for your trigger so that it can achieve nearly 100% attack success rate without significantly affecting the benign accuracy.

Useful materials:

By now, you should be familiar with model training. A full implementation of backdoor attack has been provided: https://towardsdatascience.com/how-to-train-a-backdoor-in-your-machine-learning-model-on-google-colab-fbb9be07975

ResNet-18 for CIFAR-10 models: https://github.com/kuangliu/pytorch-cifar

Artifacts:

You are required to submit a Jupyter Notebook that is runnable on Google Colab with detailed explanation of your attack and results. (File name: "problem4.ipynb")

Grading:

Total: 2 points. If code does not run, 0. If benign accuracy for original model and poisoned model is less than 90%, -1. If attack success rate is less than 95%, -1.

Problem 5: Generating Fake Videos

(Easy) Use existing DeepFake models to create fake videos.

Task:

Training a DeepFake model may be hard, but using them surely is easy. In this task, you are asked to create a DeepFake video.

1. You need to select a driving video with human faces in it.
2. You need to pick a target person image. We will refer this person as T.
3. Replace all human faces in the driving video with T's face.
4. The video quality has to be no low than 480p.
5. The video length has to be no shorter than 10 seconds.
6. Please publish your video on Google Drive and download it using Google Colab in your submission so that TAs and run it as long as the Internet is connected.
7. Faces in the driving video cannot be frozen or static. Namely, the person has to be speaking or acting.

Useful materials:

A tutorial: https://colab.research.google.com/github/AliaksandrSiarohin/first-order-model/blob/master/demo.ipynb#scrollTo=QnXrecuX6_Kw

This Google Colab has a complete walkthrough about how to do this task, including two demos.

Artifacts:

You are required to submit a Jupyter Notebook that is runnable on Google Colab with detailed explanation of your code and results. (File name: "problem5.ipynb")

Grading:

Total: 3 points. If code does not run, 0. A working solution, +1. No obvious faked frames, +1. High quality videos (>1080p), +1.