Lab-4: Adversarial Robustness Toolbox

Introduction

ART is a library developed by IBM Research which is dedicated to adversarial machine learning. Its purpose is to allow rapid crafting and analysis of attacks and defense methods for machine learning models. ART provides an implementation for many state-of-the-art methods for attacking and defending classifiers. See below links for more information on ART.

```
ART Demo: https://art-demo.mybluemix.net
ART Blog: https://www.ibm.com/blogs/research/2018/04/ai-adversarial-
robustness-toolbox/
ART Github: https://github.com/IBM/adversarial-robustness-toolbox
```

In this lab, you will work with the Adversarial Robustness Toolbox (ART) and implement an adversarial attack and its defense on a trained model. You will work with a model trained on the German Traffic Signs dataset (see Citation below) and perform an attack so that the model misclassifies a stop sign.

Dataset Citation

J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In Proceedings of the IEEE International Joint Conference on Neural Networks, pages 1453–1460. 2011.

```
@inproceedings{Stallkamp-IJCNN-2011,

        author = {Johannes Stallkamp and Marc Schlipsing and Jan Salmen and Christian Igel},
        booktitle = {IEEE International Joint Conference on Neural Networks},
        title = {The {G}erman {T}raffic {S}ign {R}ecognition {B}enchmark: A multi-class
        classification competition},
        year = {2011},
        pages = {1453--1460}

 }
```

Objectives

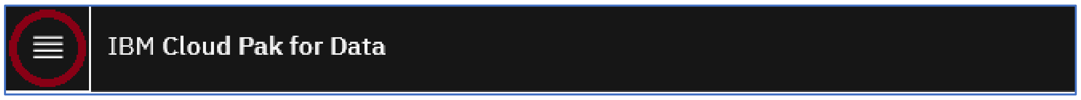
Upon completing the lab, you will learn how to:

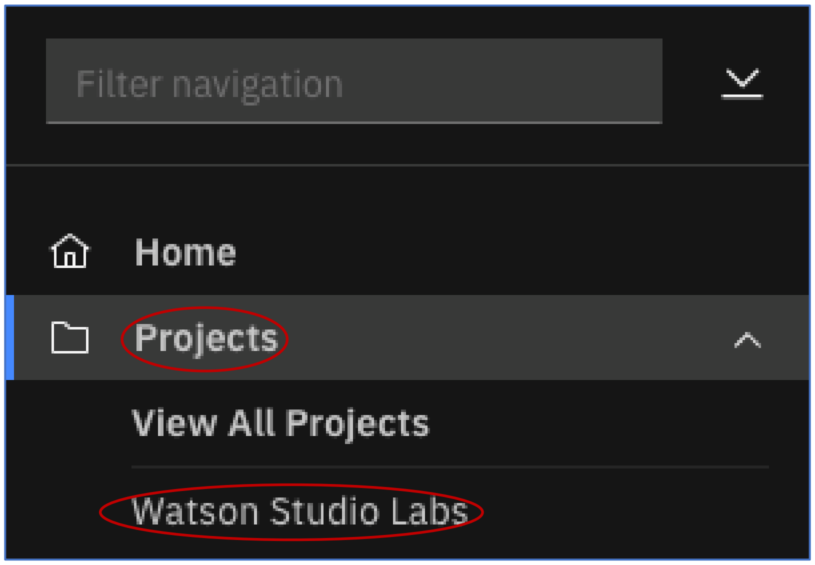
1. load a Tensorflow trained model
2. create an ART classifier object using the loaded model
3. perform an adversarial attack
4. perform a defense to make sure manipulated images can still be classified correctly

Lab Steps

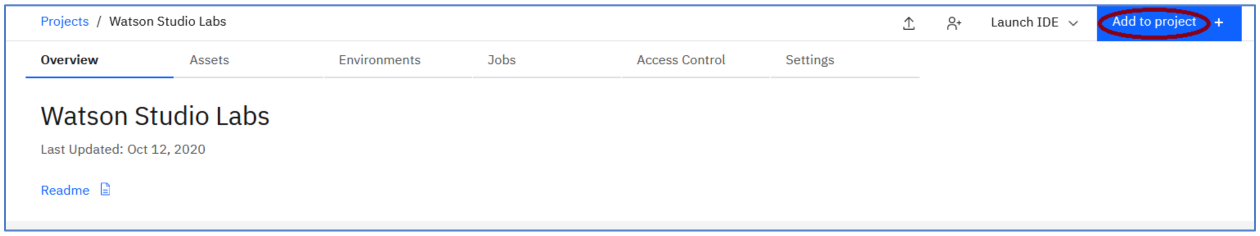
Step 1 - Create a Jupyter Notebook

1. Click on the hamburger icon ☰, then click on **Projects**, and then **Watson Studio Labs** (or whatever you named the project)





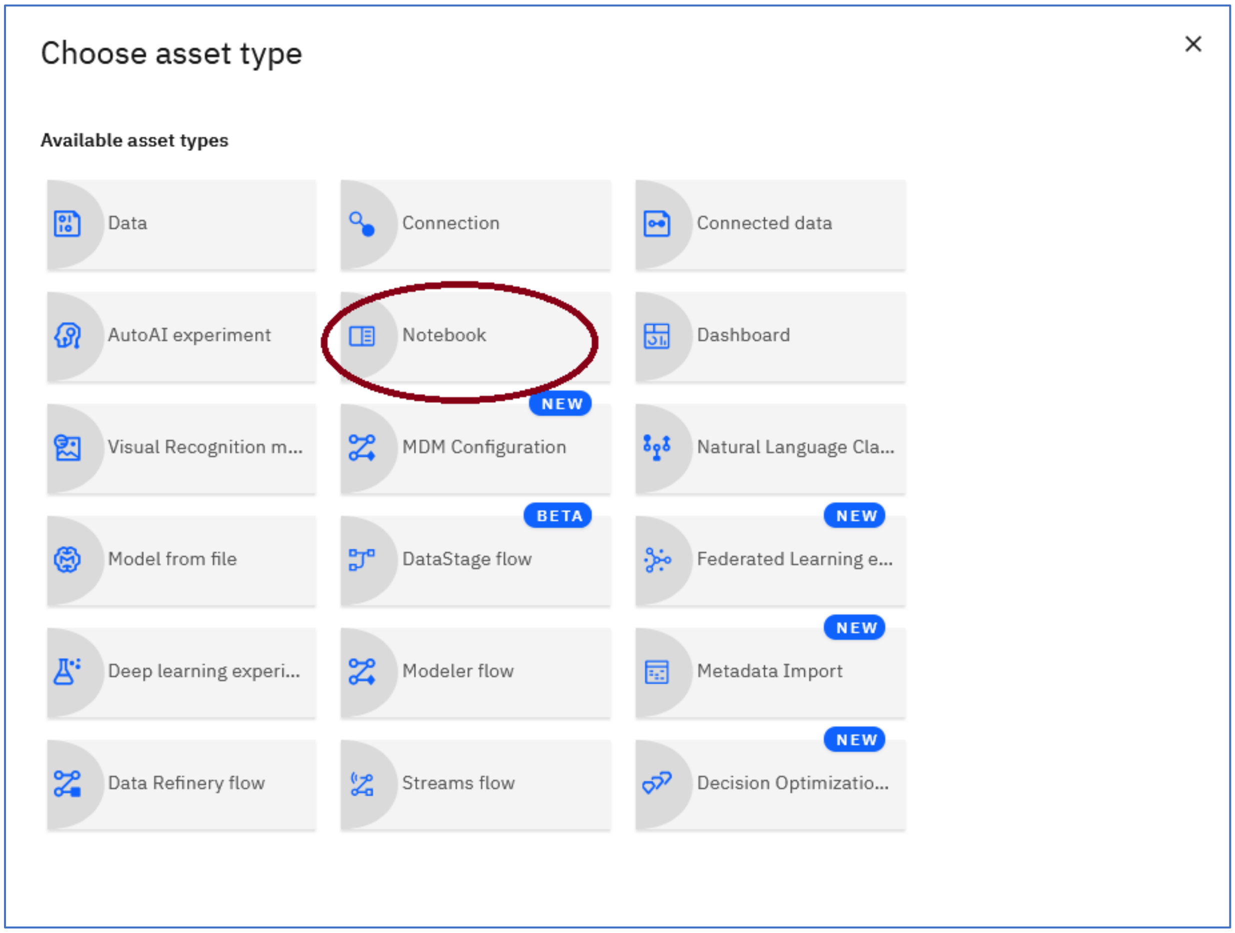
2. We are now going to create a notebook in our project. This notebook will be created from a url that points to the AIF notebook in the github repository. Click the **Add to project** link.



Watson Studio Labs

Last Updated: Oct 12, 2020

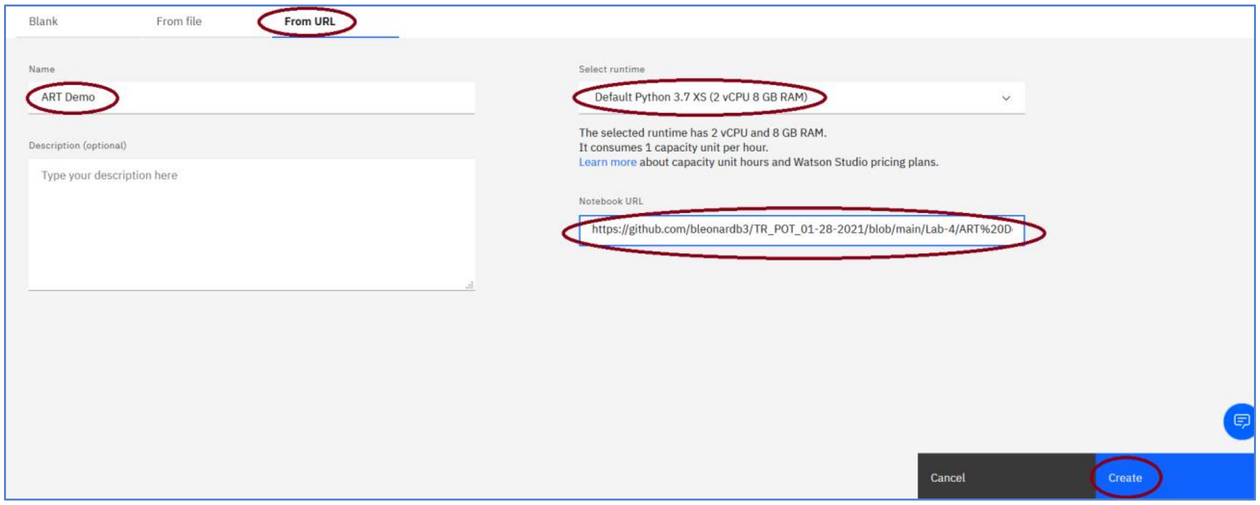
3. Click on **Notebook**



4. Click on **From URL** under **New Notebook,** enter **ART Demo** for the **Name**, optionally enter a **Description**, leave the default **runtime,** and copy and paste the following url into the **Notebook URL** field.

https://github.com/bleonardb3/TR_POT_01-28-2021/blob/main/Lab-4/ART%20Demo.ipynb

Click **Create.**



Name: ART Demo

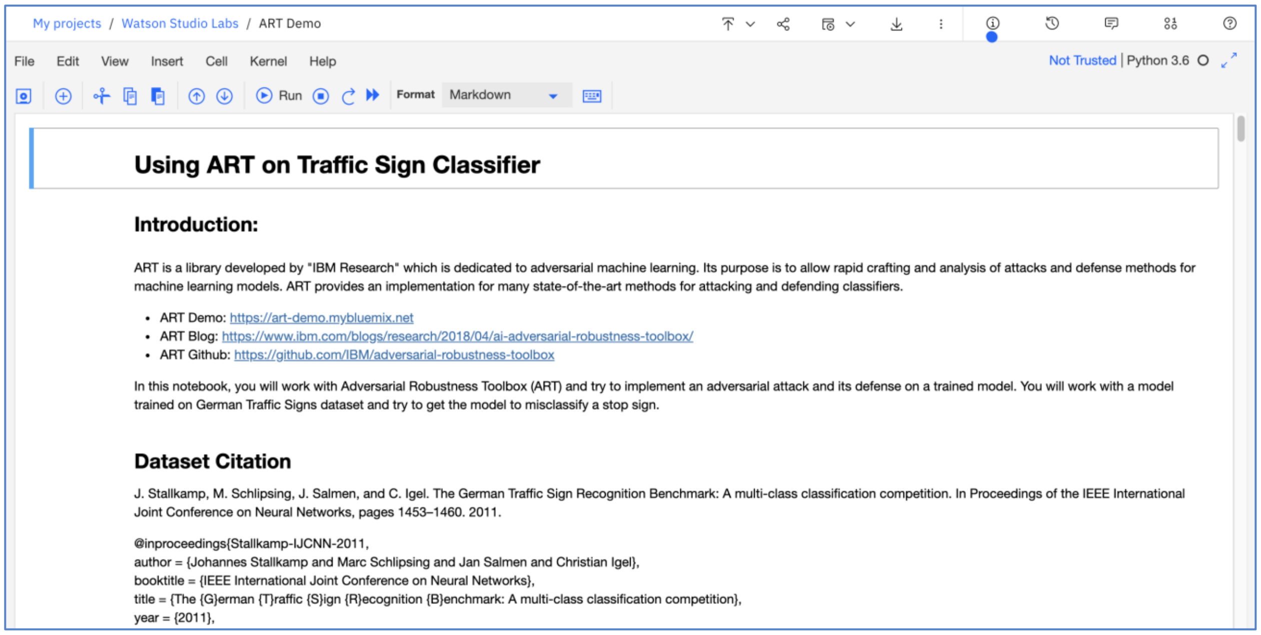
Description (optional): Type your description here

Select runtime: Default Python 3.7 XS (2 vCPU 8 GB RAM)

The selected runtime has 2 vCPU and 8 GB RAM.
It consumes 1 capacity unit per hour.
Learn more about capacity unit hours and Watson Studio pricing plans.

Notebook URL: https://github.com/bleonardb3/TR_POT_01-28-2021/blob/main/Lab-4/ART%20D

5. Place the cursor in the first documentation cell.


My projects / Watson Studio Labs / ART Demo
File Edit View Insert Cell Kernel Help
Not Trusted | Python 3.6
Run
Format Markdown

Using ART on Traffic Sign Classifier

Introduction:

ART is a library developed by "IBM Research" which is dedicated to adversarial machine learning. Its purpose is to allow rapid crafting and analysis of attacks and defense methods for machine learning models. ART provides an implementation for many state-of-the-art methods for attacking and defending classifiers.

- ART Demo: https://art-demo.mybluemix.net
- ART Blog: https://www.ibm.com/blogs/research/2018/04/ai-adversarial-robustness-toolbox/
- ART Github: https://github.com/IBM/adversarial-robustness-toolbox

In this notebook, you will work with Adversarial Robustness Toolbox (ART) and try to implement an adversarial attack and its defense on a trained model. You will work with a model trained on German Traffic Signs dataset and try to get the model to misclassify a stop sign.

Dataset Citation

J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In Proceedings of the IEEE International Joint Conference on Neural Networks, pages 1453–1460. 2011.

```
@inproceedings{Stallkamp-IJCNN-2011,
author = {Johannes Stallkamp and Marc Schlipsing and Jan Salmen and Christian Igel},
booktitle = {IEEE International Joint Conference on Neural Networks},
title = {The {G}erman {T}raffic {S}ign {R}ecognition {B}enchmark: A multi-class classification competition},
year = {2011},
```

6. Execute the cells in the notebook. Please go cell by cell and read the documentation or code comments to follow along. For those unfamiliar with Jupyter notebooks, read below.

   A Jupyter notebook consists of a series of cells. These cells are of 2 types (1) documentation cells containing markdown, and (2) code cells (denoted by a bracket on the left of the cell) where you write Python code, R, or Scala code depending on the type of notebook. Code cells can be run by putting the cursor in the code cell and pressing **<Shift><Enter>** on the keyboard. Alternatively, you can execute the cells by clicking on **Run icon** ▶ Run on the menu bar that will run the current cell (where the cursor is located) and then select the cell below. In this way, repeatedly clicking on **Run** executes all the cells in the notebook. When a code cell is executed the brackets on the left change to an asterisk ‘*’ to indicate the code cell is executing. When completed, a sequence number appears. The output, if any, is displayed below the code cell.