

Adversarial Robustness Toolbox Lab

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 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},
pages = {1453--1460}
```

```
}
```

Objectives

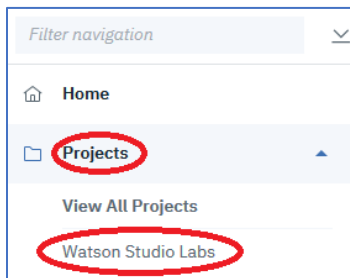
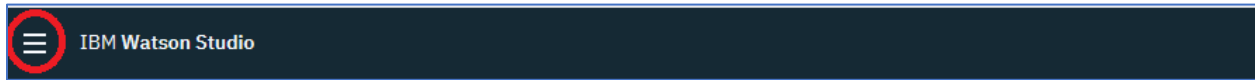
Upon completing the lab, you will learn how to:

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

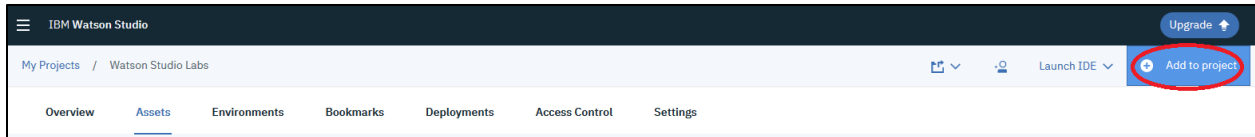
Lab Steps

Step 1 - Create a Jupyter Notebook

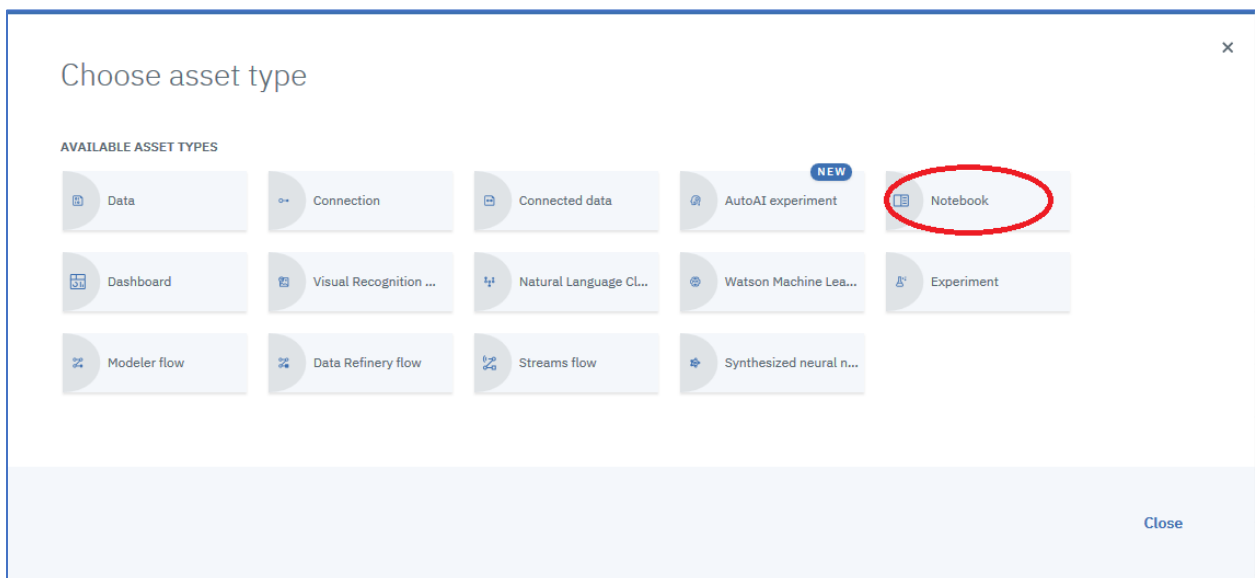
1. Click on the hamburger icon , then click on **Projects**, and then **Watson Studio Labs** (or whatever you named the project)



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



3. Click on **Notebook**



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

`https://github.com/bleonardb3/DS_POT_08-08/blob/master/Lab-8/ART%20Demo.ipynb`

Click **Create Notebook**.

The screenshot shows the 'New notebook' interface. At the top, there are three tabs: 'Blank', 'From file', and 'From URL'. The 'From URL' tab is selected and circled in red. Below the tabs, there are two main sections. The left section has a 'Name' field with 'ART Demo' entered and circled in red, and a 'Description (optional)' text area. The right section has a 'Select runtime' dropdown menu with 'Default Python 3.6 XS (2 vCPU and 8 GB RAM)' selected and circled in red. Below this, there is a 'Notebook URL' field with the URL 'https://github.com/bleonardb3/DS_POT_08-08/blob/master/Lab-8/ART%20Demo.ipynb' pasted and circled in red. At the bottom right, there are two buttons: 'Cancel' and 'Create Notebook', with the 'Create Notebook' button circled in red.

- Before executing the notebook, you will need to insert object storage credentials into one of the cells in the notebook. Please click on the following link → [Object Storage Credentials](#) to obtain your Object Storage Credentials. You should have an index card that provides your assigned object storage number. If not, ask your instructor. Locate the credentials that match your assigned object storage. Highlight the credentials and right-mouse click to get the pop-up menu. Click copy. (see below for Object Storage 1 example)

```
Object Store 1
credentials = {
  'IAM_SERVICE_ID': 'iam-ServiceId-ef59066e-bf0f-473b-b739-882015cad058',
  'IBM_API_KEY_ID': 'pAAVFSjceTUC9IMVUZq3q1JGeypNYbNy-TChNzgDRk9T',
  'ENDPOINT': 'https://s3-api.us-geo.objectstorage.service.networklayer.com',
  'IBM_AUTH_ENDPOINT': 'https://iam.ng.bluemix.net/oidc/token',
  'BUCKET': 'watsonstudiolabs-donotdelete-pr-bfti4z1sc11ctp',
  'FILE_TRAIN': 'train.p',
  'FILE_VALID': 'valid.p',
  'FILE_TEST': 'test.p',
  'FILE_SIGN': 'signnames.csv',
  'FILE_SIGN_ALL': 'signnames_all.jpg',
  'MODEL_DATA': 'final.',
  'MODEL_CHECKPOINT': '...',
  'MODEL_CRPT_INDEX': '...',
  'MODEL_META': 'final.',
  'FILE_STOP': 'stop_00',
  'FILE_DIAG_1': '001.p',
  'FILE_DIAG_2': '002.p'
}

Object Store 2
credentials = {
  'IAM_SERVICE_ID': 'iam-ServiceId-a452669b-5537-4389-9698-a24eb0798d53',
  'IBM_API_KEY_ID': 'Tnq5-da4uJmwsa5ShiJjpdXOo_urFVJbTPda4r5d63c3',
  'ENDPOINT': 'https://s3-api.us-geo.objectstorage.service.networklayer.com',
  'IBM_AUTH_ENDPOINT': 'https://iam.ng.bluemix.net/oidc/token',
  'BUCKET': 'watsonstudiolabs-donotdelete-pr-ds17v2sutgi20i',
  'FILE_TRAIN': 'train.p',
  'FILE_VALID': 'valid.p',
  'FILE_TEST': 'test.p',
  'FILE_SIGN': 'signnames.csv',
  'FILE_SIGN_ALL': 'signnames_all.jpg',
}
```

6. Scroll down in the notebook until you see **INSERT OBJECT STORAGE CREDENTIALS**. In the code cell below, place the cursor on a new line in the code cell underneath **#Insert credentials below**. Right-mouse click to get the pop-up menu and click **Paste** to paste the credentials.

```
INSERT OBJECT STORAGE CREDENTIALS

In [ ]: # @hidden_cell
# The following code contains the credentials for a file in your IBM Cloud Object Storage.
# You might want to remove those credentials before you share your notebook.
#Insert credentials below

In [ ]: from
import
cos =
i
i
i
c
e

In [ ]: !mkdi
!mkdi
res=

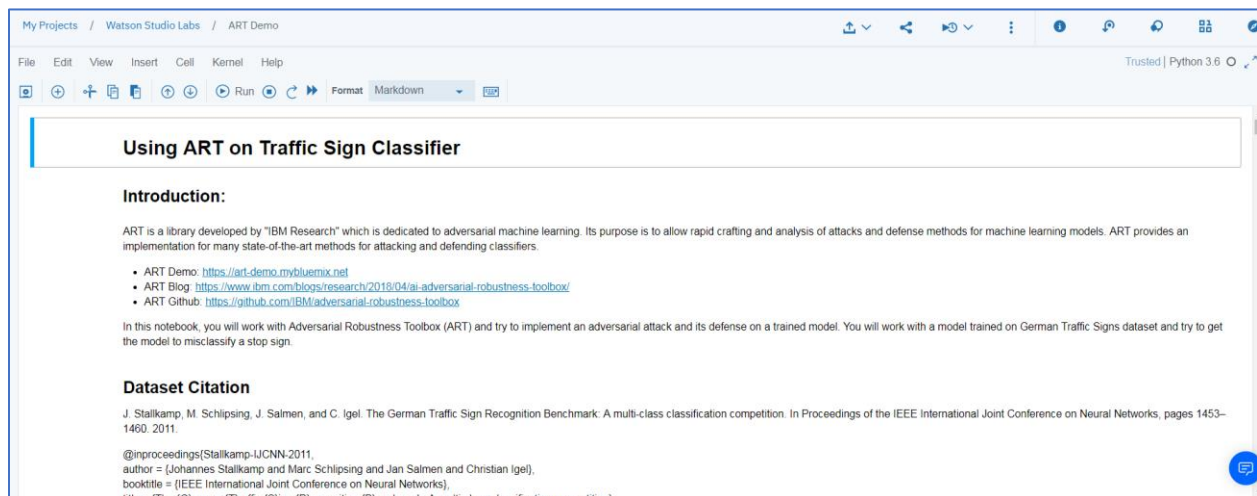
In [ ]: ...txt ^
```

7. The result should appear as below.

INSERT OBJECT STORAGE CREDENTIALS

```
In [ ]: #@hidden_cell
# The following code contains the credentials for a file in your IBM Cloud Object Storage.
# You might want to remove those credentials before you share your notebook.
# Insert credentials below
credentials = {
    'IAM_SERVICE_ID': 'iam-ServiceId-ef59066e-bf0f-473b-b739-882015cad058',
    'IBM_API_KEY_ID': 'pAAVFSJceTUC9IMVUZq3q1JGeyplNYbly-TChNzgDRk9T',
    'ENDPOINT': 'https://s3-api.us-geo.objectstorage.service.networklayer.com',
    'IBM_AUTH_ENDPOINT': 'https://iam.ng.bluemix.net/oidc/token',
    'BUCKET': 'watsonstudiolabs-donotdelete-pr-bfti4z1sc11ctp',
    'FILE_TRAIN': 'train.p',
    'FILE_VALID': 'valid.p',
    'FILE_TEST': 'test.p',
    'FILE_SIGN': 'signnames.csv',
    'FILE_SIGN_ALL': 'signnames_all.jpg',
    'MODEL_DATA': 'final.ckpt.data-00000-of-00001',
    'MODEL_CHECKPOINT': 'checkpoint',
    'MODEL_CKPT_INDEX': 'final.ckpt.index',
    'MODEL_META': 'final.ckpt.meta',
    'FILE_STOP': 'stop_001.png',
    'FILE_DIAG_1': '001.png',
    'FILE_DIAG_2': '002.png'
}
```

8. Scroll back to the top of the notebook and place the cursor in the first documentation cell.



My Projects / Watson Studio Labs / ART Demo

File Edit View Insert Cell Kernel Help Trusted | Python 3.6

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/2019/04/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. Stal Kamp, M. Schlippsing, 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{StalKamp-UCNN-2011,
author = {Johannes Stal Kamp and Marc Schlippsing and Jan Salmen and Christian Igel},
booktitle = {IEEE International Joint Conference on Neural Networks},
title = {The German Traffic Sign Recognition Benchmark: A multi-class classification competition},

9. Execute the code cells in the notebook. 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** 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.