Lab Session 3 (L3)

Cybersecurity for AI

Use Case on Evasion in a Distributed IoT System

Objectives

The objectives of L3 are:

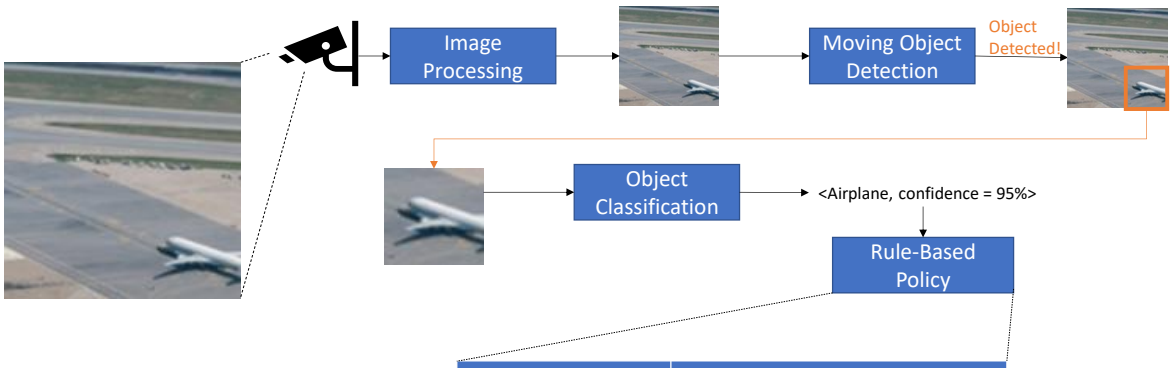
- To practice with the process of adversarial sample generation to hack a machine learning classifier.
- To emulate a distributed system where a Man-In-the-Middle intercepts images between a data source and model, adding adversarial noise to force model misclassification.
- To design, implement, and evaluate different measures to mitigate the impact of evasion by adversarial noise addition.

Statement

Let us consider the following IoT scenario deployed in an airport:

The outdoor premises (tracks, access roads) of the airport are continuously monitored by surveillance cameras that collect images and process them by means of ML algorithms that detect and classify objects in movement. Each camera covers a limited region of several square meters. Cameras are equipped with computing resources, so one *edge node* is available at each camera pole. All the edge nodes are connected with a *centralized controller* in the main terminal by means of high-bitrate connections (wireless or wired).

The ML-based workflow that every camera runs periodically (e.g. every second) is sketched in the following picture:



Class	Risk – Action
Authorized Vehicle	No Risk – None
Non-authorized Vehicle	Very High – Block Aircraft Traffic & Activate Border Police Protocol Against Intrusions
Animal	Medium – Warn Aircraft taking off and landing & Move Surveillance Team to the affected area

Thus, every second the camera collects a frame that is processed by a moving object detection process that detects whether an object is moving on the fixed background covered by the camera. If an object is detected, the region of the image that contains the object is selected and passed to a second process that classifies the type of object. The obtained class jointly with its confidence level is then processed by a rule-based policy determined by the human operator. This process is a simple lookup table that triggers an action depending on the class. It distinguishes between authorized vehicles (normal operation, no action to perform), non-authorized vehicles (a critical issue that requires blocking traffic and activating airport police protocols), and animals (a medium risk issue that triggers warning notifications and mobilize surveillance teams).

In this lab, we will setup a simplified version of this system and incorporate Man-In-The-Middle (MitM) AI attacks to lead classification and rule-based policy to misleading decisions and actions. From now on, we are going to assume that the attacker has access to a copy of the ML classifier running in object classification module and also knows the rule-based policy.

Preliminaries

The attached Jupyter Notebook (tutorial_adversarial.ipynb) contains a tutorial code that you can use to achieve the objectives of this lab. It contains examples for:

1) Managing the CIFAR10 dataset for image classification (https://www.cs.toronto.edu/~kriz/cifar.html)
2) Defining and training a ML classifier based on Deep Neural Networks (DNN)

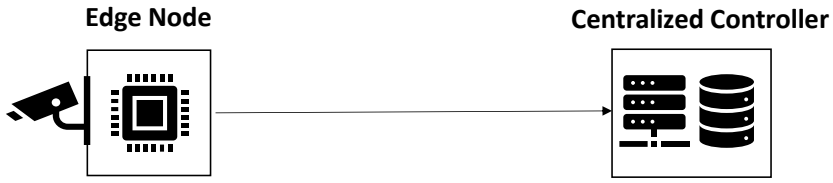
3) Generating adversarial samples by adding noise following a well-known gradient-based method

Run the code (cell by cell), understanding the action of every piece of code and ensuring that you can run it properly with no failures. You can use it for the rest of the lab or dismiss it if you want to use alternative methods.

The tasks of this lab are next detailed (including the score of each task):

Task 1: System Design and Security Analysis (1.5 p.)

Considering the following infrastructure:



[Q1] Let us analyse the following three different configurations for the placement of each of the processes involved in the workflow

Option	Edge Node	Centralized Controller
A	Image processing Moving object detection Object classification	Rule-based policy
B	Image processing Moving object detection	Object classification Rule-based policy
C	Image processing	Moving object detection Object classification Rule-based policy

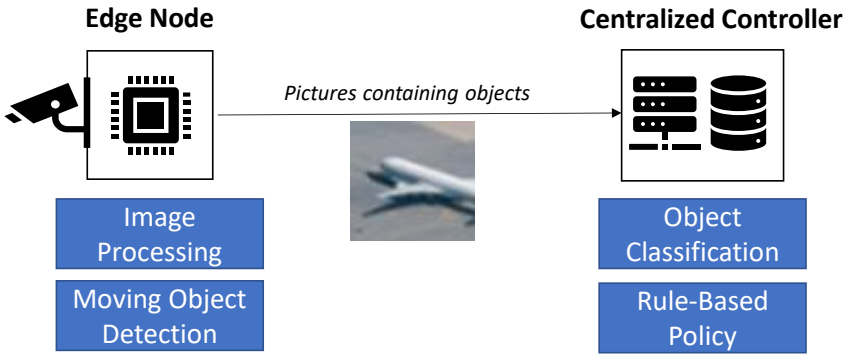
Briefly comment on the strengths and weaknesses from the point of view of performance and security of each option. Indicate which type of data is going to be transmitted between edge node and centralized controller, commenting on its characteristics and how frequent is going to be transmitted. Indicate your preferred choice and why.

You can use the following template:

Option	**A**	**B**	**C**
Performance (pros / cons)			
Security (pros / cons)			
Data Analysis (type, volume, frequency)			
Preferred? Yes/No & Why)			

Task 2: System implementation (1.5 p.)

Regardless of your choice and analysis done in Task 1, for this task we are going to consider the following system design:



[Q2] Prepare a code to pre-process the data and train a DNN classifier that distinguishes the 10 different labels that you can find in CIFAR10 dataset according to the classes of our model:

Model Class	CIFAR10 Label
Authorized Vehicle	*'airplane', 'automobile'*
Non-Authorized Vehicle	*'ship', 'truck'*
Animal	*'bird', 'cat', 'deer', 'dog', 'frog', 'horse'*

Explain the model structure (type of model, number of layers and characteristics), and show performance (accuracy) results of the trained model. Once you have it ready, save it model.

[Q3] Implement a function called "edge_process" that will read random samples from CIFAR10 datasets and return them. Then, implement a function called "controller_process" that receives the output sample from "edge_process" and perform object classification and rule-based policy actions. The latter must contain and make use of the model trained in [Q2]. Verify that the process runs properly and that the actions are in line with the performance of the trained model.

As a suggestion, to solve the previous question, prepare a code file containing the two functions:

```
def edge_process(xxx):

    # your code

    return sample

def controller_process(xxx):

    # your code

    return action
```

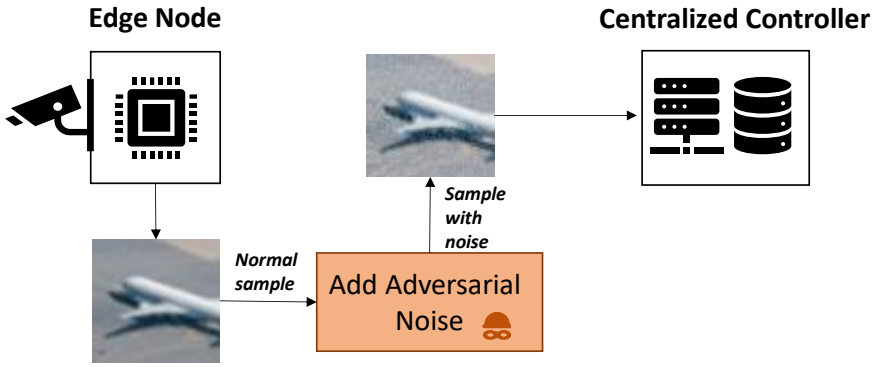
and prepare a main script that import the two functions and run, for a number of samples *n*, the proposed workflow:

```
import edge_process, controller_process
```

```
n = 100

for i in range(n):

    sample = edge_process(xxx) # image collection and
(emulated object detection)

    # transmission between edge node and controller is
emulated

    action = controller_process(sample)
```

Task 3: Man-In-The-Middle (MiTM) Attack (5 p.)

Now we assume that the image with the object is intercepted by an adversary, which will run a process to add adversarial noise to the sample and push it to the controller:



[Q4] Explain (without code) what is the method you used and how it works. Then, prepare a new function called "MiTM_process" that receives a normal sample and returns a sample with adversary noise. Evaluate the function for different configuration of the parameters that you identify. Show how classification is deceived as a function of the amount of noise added. Decide the best configuration of the hyperparameters of your method so that you can get the target malfunctioning generating images that are not easy to detect as altered. Plot some images before and after adversary noise addition in order to visually evaluate its impact

After implementing and validating the attacker function, add it to the main code:

```
    sample = edge_process(xxx) # image collection and
(emulated object detection)

    sample = MiTM_process(sample)

    action = controller_process(sample)
```

[Q5] Analyse critically (pros, cons, requirements, assumptions, side attacks that are required) the three options that the attacker has in order to place the "MiTM_process" function:

1) **In the edge node, i.e., the sample leaves the edge node with the adversary noise already added.**

2) **In an intermediate/remote node/site, i.e., the attacker intercepts the packet traffic containing the image while in transit, perform the noise addition, injects the modified image to the packets, and forward them to the controller.**
3) **In the controller, i.e., the received normal samples are altered before running the classifier.**

[Q6] Modify the "edge_process" code in order to randomly select pictures of authorized vehicles with a probability equal to *p* and pictures of non-authorized vehicles and animals with probability 1-*p*. Comment "MiTM_process" line to check that the script works well and classifier still behaves properly. Then, uncomment "MiTM_process" and run the script for different values of *p*. Analyse the obtained results in the way you thing more interesting. Among the different cases, consider a large value of *p* (i.e., most of the images are authorized vehicles). Guess if the obtained results can lead to find a way to detect that a MiTM attack is happening.

Task 4: Detecting or mitigating the effects of a MiTM attack (2 p.)

Assuming that the adversary is able to inject adversarial noise successfully, let us now concentrate on proposing a mechanism to detect such attack.

[Q7] Define ONE method that could work to detect or mitigate the effect of a MiTM attack. Redo the drawing of task 2 with the addition of the new process/es. Implement it/them as part of edge_process and/or controller_process functions. Evaluate its performance by running them apart and/or included in the main script. Report all the explanation and methodology, results and findings/conclusions, jointly with the code.

Some ideas that you can adopt (not restricted to) are:

- Use different models for classification (model ensemble). You can assume that the attacker can have access to all but only generate noise for one of them at a time. The controller can make the decision based on the multiple answer from multiple classifiers, instead of only one. Use this functionality to detect inconsistencies and irregular behaviour.
- Add adversarial samples in your training dataset. This means, generating synthetic samples consisting each in one original image with adversarial noise added, and include this in the dataset with the same class than the original one. Add images from as many classes as possible.

Decide and comment your choice with the lecturer before the end of the session, and follow his guidelines about how to proceed with [Q7].

Delivery

Zip the document **L3_D1** containing the answers to the questions and the code of all the tasks in a compressed file with name **L3_[your_group_number].zip**. Submit the file to **RACO (Practicals/L3**).

Deadline: Dec 19th