

Introduction

Artificial intelligence (AI) models have become increasingly vulnerable to various types of attacks, including adversarial attacks, evasion attacks, and AI poisoning attacks. These attacks can compromise the integrity and reliability of AI systems, leading to severe consequences in critical applications such as healthcare, finance, and national security. Therefore, it is essential to develop robust defence mechanisms to protect AI models against these attacks.

Adversarial Attacks

Adversarial attacks involve adding noise or perturbations to the input data to mislead the AI model into making incorrect predictions. These attacks can be categorized into two types:

- **White-box attacks:** The attacker has complete knowledge of the AI model's architecture, parameters, and training data.
- **Black-box attacks:** The attacker has limited or no knowledge of the AI model's architecture, parameters, and training data.

Evasion Attacks

Evasion attacks involve manipulating the input data to avoid detection by the AI model. These attacks can be launched by adding noise or perturbations to the input data, making it difficult for the AI model to identify the attack.

AI Poisoning Attacks

AI poisoning attacks involve manipulating the training data to compromise the AI model's performance. These attacks can be launched by injecting malicious data into the training dataset, causing the AI model to learn incorrect patterns and relationships.

Defence Mechanisms

Several defence mechanisms can be employed to protect AI models against these attacks:

- **Adversarial Training:** Training the AI model on adversarial examples to improve its robustness against attacks.
- **Input Preprocessing:** Preprocessing the input data to detect and remove noise or perturbations.
- **Anomaly Detection:** Detecting and flagging unusual patterns or behaviour in the input data.
- **Ensemble Methods:** Combining the predictions of multiple AI models to improve robustness against attacks.
- **Game-Theoretic Approaches:** Modelling the interaction between the attacker and the AI model as a game, and using game-theoretic techniques to develop optimal defence strategies.

Azure Prototype Requirements

To develop a robust defence mechanism against attacks on AI models, the following Azure prototype requirements are necessary:

- **Azure Machine Learning:** A cloud-based platform for building, training, and deploying AI models.

- **Azure Notebook:** it is help to run our code for models.

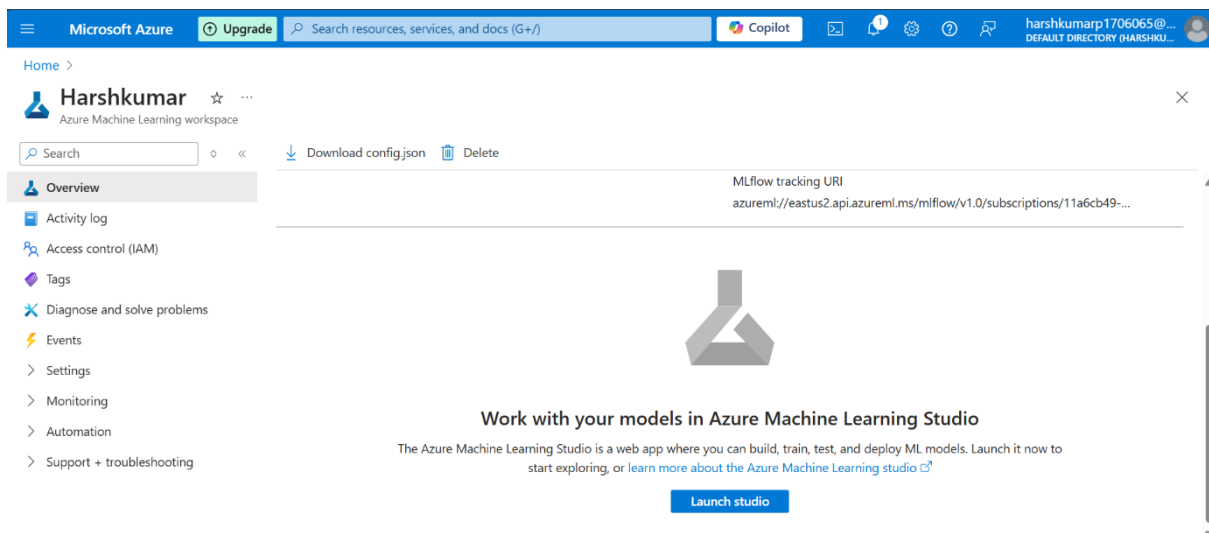
Plan for the Capstone Project

The plan for the capstone project involves the following steps:

1. **Literature Review:** Conduct a comprehensive review of existing defence mechanisms against attacks on AI models.
2. **Data Collection:** Collect and preprocess a dataset for training and testing the AI model.
3. **AI Model Development:** Develop and train an AI model using Azure Machine Learning.
4. **Attack Simulation:** Simulate various types of attacks on the AI model, including adversarial attacks, evasion attacks, and AI poisoning attacks.
5. **Defence Mechanism Development:** Develop and implement defence mechanisms to protect the AI model against attacks.
6. **Evaluation and Testing:** Evaluate and test the effectiveness of the defence mechanisms against attacks.
7. **Deployment:** Deploy the AI model and defence mechanisms on Azure.

Azure implementation:

create azure machine learning studio



computer instance

Azure AI | Machine Learning Studio

Default Directory > Harshkumar > Compute

Compute

The "Kubernetes clusters" tab is now where you can access previous versions of "inference clusters" (also known as "AKS clusters") and "attached Kubernetes" compute types along with any previously created compute targets using those types. [Learn more about Kubernetes clusters.](#)

Compute instances **Compute clusters** Kubernetes clusters Attached computes Serverless instances

Create a single or multi node compute cluster for your training, batch inferencing or reinforcement learning workloads. [Learn more about compute clusters](#)

Alternatively, you can now run a training job without having to create and manage compute by using serverless. [Learn more here](#)

[+ New](#) [Refresh](#) [Delete](#) [Reset view](#) [View quota](#)

Search

Name	State	Size	Location	Created on ↓	Act
CPU-compute-cluster	Succeeded (0 nodes)	Standard_DS3_v2	eastus2	Sep 3, 2024 4:45 PM	0

<https://ml.azure.com/compute/list?wsid=/subscriptions/11a6cb49-8495-4415-9aa8-2dedbbb6c8fc/resourcegroups/Harsh/providers/Microsoft.MachineLearningServices/workspaces/Harshkumar&tid=9231e719-f11-44bf-b162-26f5...>

baseline model

Azure AI | Machine Learning Studio

Model catalog

Default Directory > Harshkumar > Notebooks

Notebooks

Files Samples

Users

- harshkumarp1706065
 - PY adversarial_attack.py
 - attack_stats.png
 - PY baseline_model.py
 - PY defense_mechanism.py
 - harshp_adv.npy
 - PY image.py
 - PY Untitled.py

1243/1875 [=====] - ETA: 0s - loss: 0.0440 - accuracy: 0.1286/1875 [=====] - ETA: 0s - loss: 0.0445 - accuracy: 0.1327/1875 [=====] - ETA: 0s - loss: 0.0444 - accuracy: 0.1368/1875 [=====] - ETA: 0s - loss: 0.0442 - accuracy: 0.1410/1875 [=====] - ETA: 0s - loss: 0.0439 - accuracy: 0.1453/1875 [=====] - ETA: 0s - loss: 0.0441 - accuracy: 0.1493/1875 [=====] - ETA: 0s - loss: 0.0439 - accuracy: 0.1535/1875 [=====] - ETA: 0s - loss: 0.0439 - accuracy: 0.1577/1875 [=====] - ETA: 0s - loss: 0.0436 - accuracy: 0.1620/1875 [=====] - ETA: 0s - loss: 0.0441 - accuracy: 0.1663/1875 [=====] - ETA: 0s - loss: 0.0438 - accuracy: 0.1699/1875 [=====] - ETA: 0s - loss: 0.0438 - accuracy: 0.1731/1875 [=====] - ETA: 0s - loss: 0.0439 - accuracy: 0.1760/1875 [=====] - ETA: 0s - loss: 0.0438 - accuracy: 0.1795/1875 [=====] - ETA: 0s - loss: 0.0441 - accuracy: 0.1829/1875 [=====] - ETA: 0s - loss: 0.0441 - accuracy: 0.1863/1875 [=====] - ETA: 0s - loss: 0.0440 - accuracy: 0.1875/1875 [=====] - 2s 1ms/step - loss: 0.0440 - accuracy: 0.9865

(azureml_py38) azureuser@harshkumarp17060651:~/cloudfiles/code/Users/harshkumarp1706065\$

adversarial attack

Azure AI | Machine Learning Studio

Model catalog

Default Directory > Harshkumar > Notebooks

Notebooks

Files Samples

Users

- harshkumarp1706065
 - PY adversarial_attack.py
 - attack_stats.png
 - PY baseline_model.py
 - PY defense_mechanism.py
 - harshp_adv.npy
 - PY image.py
 - PY Untitled.py

114] CPU Frequency: 2593905000 Hz

1/313 [.....] - ETA: 30s - loss: 0.0409 - accuracy: 1.061/313 [=====] - ETA: 0s - loss: 0.1209 - accuracy: 0.96122/313 [=====] - ETA: 0s - loss: 0.1212 - accuracy: 0.96182/313 [=====] - ETA: 0s - loss: 0.1104 - accuracy: 0.96242/313 [=====] - ETA: 0s - loss: 0.0984 - accuracy: 0.96303/313 [=====] - ETA: 0s - loss: 0.0877 - accuracy: 0.97313/313 [=====] - 0s 833us/step - loss: 0.0901 - accuracy: 0.9707

Test accuracy: 0.9707

1/313 [.....] - ETA: 3s - loss: 3.3021 - accuracy: 0.1562/313 [=====] - ETA: 0s - loss: 3.8807 - accuracy: 0.13124/313 [=====] - ETA: 0s - loss: 3.8336 - accuracy: 0.13186/313 [=====] - ETA: 0s - loss: 3.6633 - accuracy: 0.15248/313 [=====] - ETA: 0s - loss: 3.4358 - accuracy: 0.18311/313 [=====] - ETA: 0s - loss: 3.2891 - accuracy: 0.20313/313 [=====] - 0s 814us/step - loss: 3.2898 - accuracy: 0.1998

Test accuracy (adversarial): 0.1998

(azureml_py38) azureuser@harshkumarp17060651:~/cloudfiles/code/Users/harshkumarp1706065\$

defence mechanism

The screenshot shows the Azure AI Machine Learning Studio interface. The left sidebar contains the 'Model catalog' and 'Authoring' sections. The 'Notebooks' section is active, showing a list of notebooks under the 'harshkumarp1706065' user. The notebook 'defense_mechanism.py' is selected. The main pane displays the execution output of the notebook, which includes a table of results for various attacks and a bar chart titled 'Attack Stats'.

Attack Type	Accuracy
Attack 1	0.8
Attack 2	0.74
Attack 3	0.75
Attack 4	0.76
Attack 5	0.77

code for visual image

The screenshot shows the code for visualizing the attack stats. The code is written in Python and uses the matplotlib library to create a bar chart. The chart is titled 'Attack Stats' and shows the accuracy for five different attack types. The code is as follows:

```
2 import matplotlib.pyplot as plt
3
4 # Define the attack stats (e.g., accuracy, loss, etc.)
5 attack_stats = [0.9, 0.8, 0.7, 0.6, 0.5]
6
7 # Define the labels for the x-axis
8 labels = ['Attack 1', 'Attack 2', 'Attack 3', 'Attack 4', 'Attack 5']
9
10 # Create a bar chart of the attack stats
11 plt.bar(labels, attack_stats)
12
13 # Set the title and labels
14 plt.title('Attack Stats')
15 plt.xlabel('Attack Type')
16 plt.ylabel('Accuracy')
17
18 # Save the image to a file
19 plt.savefig('attack_stats.png')
20
21 # Display the image
22 plt.show()
```

visual image of attacks

