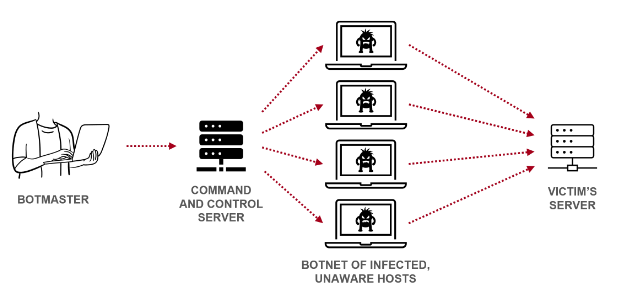
**Lab 1: DDoS-type attacks Recognition with Machine Learning**

**Overview**

Students will learn how to use the Pennylane Python Library to simulate quantum computing in this experiment. Building a circuit with pennlylane and determining the output's state would be our first steps. We would also use it for identifying DDoS-style attacks. By utilizing quantum phenomena, quantum computing provides substitute methods for computing.

**DDoS attacks:** A cyberattack known as distributed denial of service (DDoS) occurs when numerous hosts attempt to connect to a victim's server repeatedly until it collapses and is unable to handle a valid request. It's a coordinated effort from a single location, typically carried out with malicious software, that infects unknowing owners' devices:



It's critical to identify this kind of assault early on in the server connection process. By doing this, the attackers are kept from stealing resources, overloading the system, and ultimately crashing the website or application.

**Scenario:**

Consider yourself attempting to visit the website of your government. All you wanted to do was read about new laws or print some forms, but the website won't load. The servers can be the target of a cyberattack. Hybrid Quantum Neural Network (H-QNN) is a reliable model that can swiftly and without causing any delays categorize a user request, for instance, as potentially benign or a DDoS attack. It does not require transmitting it to a quantum device for prediction because it is simply simulated locally.

**Dataset:** DDoS Evaluation Dataset ([CIC-DDoS2019](https://www.unb.ca/cic/datasets/ddos-2019.html))

The network traffic analysis results are included in this dataset, together with labeled flows based on protocols, attacks, source and destination ports, source and destination IPs, and time stamps. Every data point has the labels "benign" (a data bit can indicate any kind of threat) and "Simple Service Discovery Protocol" (a DDoS assault type).

To download the data:

go to the dataset website and click on the download

submit your information to get the access

go to CSVs directory,

download CSV-01-122.zip file

After that, unpack and find in 01-12 directory DrDoS\_SSDP.csv file. That's our data.

**Set up Environment:**

Data preprocessing: pandas, Numpy

Quantum part of the H-QNN: Pennylane

Classical part of the H-QNN: TensorFlow

Evaluation: ScikitLearn

Visualization: Matplotlib, Seaborn

**Task 0: Installation of Pennylane**

We will review some sample examples below to help us become comfortable with the syntax needed to build circuits and simulate them in pennylane. Pennylane is a Python library that requires version 3.7 or higher to function.

1. Install Pennylane and Jupyter notebook

Open the terminal: cmd

Verify that you have an updated version of python, by typing “python –version”

To install pennylane, type: “pip install pennylane”

Then open jupyter by typing “jupyter notebook” on the terminal.

Or open google colab and do “pip install pennylane”

To check the installation of pennylane is working properly, type “ import pennylane as qml” in the jupyter notebook

b. Let us define a quantum device,

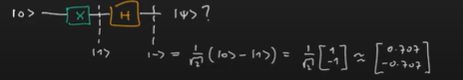
The most common type of device is default.qubit

|  |
| --- |
| Code1 |

If no error is generated, then your device is working

**Task 1: Simulate the evolution of state zero through the circuit and to simulate the state of the output**

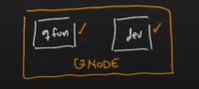
Understand quantum state and the representation in both bracket notation and state vector notation. How quantum states evolve over unitaries, gates and dealing with composites states. Let us start with a one qubit circuit where we initialize the qubit in state zero and then apply an x and a hadamard gate. We are interested in finding the state side the output is



Implementation:

1. Import pennylane as qml

2. Define a function: Inside the function is where we are going to apply our gate to the circuit. We want to simulate the evolution of state zero through the circuit and to simulate the state of the output. Then we will wrap our quantum function and the device through a Qnode to execute the simulation



|  |
| --- |
| Code 2a |

3. Then we define our device with the standard cubit simulator known as default.qubit and then specify the number of qubit you want to simulate, in the circuit we use one so wires = 1

|  |
| --- |
| Code 2b |

4. Then create Qnode and then pass the quantum function and the device, to execute the simulation.

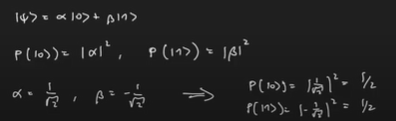
|  |
| --- |
| Code 2c |

5. Then draw our circuit

|  |
| --- |
| Code 2d |

|  |
| --- |
| Show the screen shot of the Circuit and print out the state of the output |

6. Now let us find the probability of measuring zero which is associated with projecting into state zero or a one which is associated with projecting it to state one

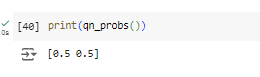


Finding the total unitary from input to output, the total unitary is the multiplication of the matrix of the X gate and hadamard gate

Now we write a function that call the gate within a qnote and then we get the quantum state and probabilities.

|  |
| --- |
| Code 3 |

Show the result of the probabilities



**Task 2: DDOS attack with Classifier machine learning: Preprocessing**

a. **Importing all needed packages:**

|  |
| --- |
| Code 4 and Code 4a |

Next, let’s set up global variables:

|  |
| --- |
| Code 5 |

Let’s see the size of our data:

|  |
| --- |
| Code 6 |

Now we get rid of columns with only one value, irrelevant columns, columns with ‘NaN’ values, correlated columns.

|  |
| --- |
| Code 7 |

|  |
| --- |
| Screenshot of the output |

We can see some NaNs in Flow Bytes/s feature, we'll drop this column out of our training input:

|  |
| --- |
| Code 8 |

Now, we’ll remove columns with ‘irrelevant’ information:

|  |
| --- |
| Code 9 |

After that, let’s remove columns with one unique value:

|  |
| --- |
| Code 10 |

Finally, we eliminate correlated columns. Let’s generate correlation matrix:

|  |
| --- |
| Code 11 |

We won’t plot the whole matrix, since it’s too noisy with that number of features. You can plot it easily with this code:

|  |
| --- |
| Code 12 |

Let’s generate pairs to eliminate from matrix. We’ll set the correlation factor threshold on 0.94:

|  |
| --- |
| Code 13 |

Show outputpf the correlation



## **Task 3: Training data preparation**

Initially, we must decrease the quantity of samples within our dataset. We selected 10,000 samples out of over 2,600,000 since our quantum model is far slower than a classical neural network and training for all the data has taken a long time. Because there are very few data points in the dataset (around 0.03% are classified as BENIGN), we will remove all samples that are classified as non-threatening and select the remaining samples at random to create a total of 10,000..

|  |
| --- |
| code 14 |

We’ll choose randomly n samples from over 2,600,000 rows:

* all (763) samples tagged as 'BENIGN'
* n-763 samples tagged as 'DrDoS\_SSDP'

|  |
| --- |
| Code 15 |

Lastly, labels must be converted from strings to integers, with 0s denoting a benign data item and 1s denoting a DDoS attack:

|  |
| --- |
| Code 16 |

**b.** **PCA feature extraction:**

Then, we need to reduce our input for the model. Our dataset, even with previously removed correlated columns, has 35 features. To make a robust quantum model, we need to perform feature extraction. With PCA fitted on 10,000 previously chosen samples, we can go down to 2 features in ranges -1to 1.

|  |
| --- |
| Code 17 |

In task 2 and 3, We’ve prepared the data for training small, quantum models with dropping some features and performing the Principal Component Analysis (PCA)

## **Task 4: Hybrid- Quantum Neural Networks (H-QNN) and Deep Neural Network (DNN) models training**

We’ll train a Hybrid Quantum Neural Network (H-QNN) on a DDoS-type attacks dataset from the dataset above. Let’s begin with importing all needed packages.

|  |
| --- |
| Code 18 |

Next, let’s set up global variables:

|  |
| --- |
| **code 19** |

**b.** **Train/test/validation split:**

We utilized the selected 10,000 samples from Tasks 2 and 3 as a train and test dataset. Subsequently, we divided them into training, validation, and testing segments, with a ratio of 70%-15%-15%.

|  |
| --- |
| Code 20 |

## **c.** **H-QNN and DNN models training:**

**Model:**

A fully connected layer Hybrid Quantum Neural Network (H-QNN) architecture will be used for the input and output. PennyLane's implementation of a quantum layer is the hidden layer. We first convert it from a parametrized quantum circuit to a trainable Keras layer.

Let’s specify our training parameters:

|  |
| --- |
| **Code 21** |

**d.** **Quantum layer**

Many high-level wrappers are available from Pennylane for building quantum circuits for Quantum Neural Networks. We will utilize these to create the Pennylane quantum circuit object, or qnode, which will be our Quantum Layer in the H-QNN later on. We need to implement:

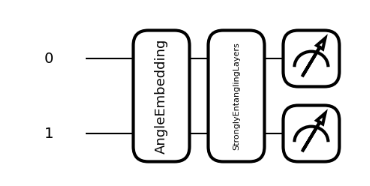
**Classical data encoding**:

**Trainable ansatz**:

**Data decoding**:

Let’s code it:

|  |
| --- |
| Code 22 |

**Show output:** 

Two dense layers will be used as the input and output layers to finish the model. Since our model produces binary classification with two integers as the output, the first one will have a sigmoid activation function and the second one a softmax. We must include our qnode in qml to make it a valid Keraslayer.QNN.wrapper for KerasLayer:

|  |
| --- |
| Code 23 |

**e.** **H-QNN training:**  
Finally, we are ready to run the training:

|  |
| --- |
| Code 24 |

As we can see, our model trained in just one epoch. Let’s evaluate it and see the results in a confusion matrix:

|  |
| --- |
| Code 25 |

**Task 5: DNN model**

This shows that H-QNN can be very effective in a simple classification. For comparison, we’ll train a classical DNN model. As activation, we’ll use relu instead of sigmoid, but the layers shape stays the same:

|  |
| --- |
| Code 26 |

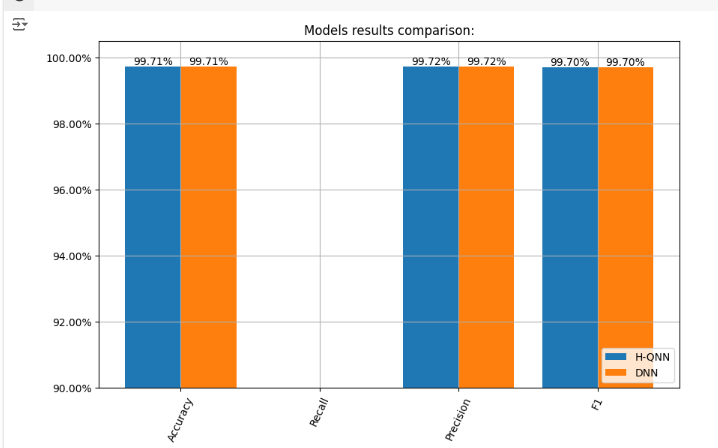
**Training:**

|  |
| --- |
| **Code 27** |

As we can see, the results are similar. Let’s put all the metrics on the same plot:

|  |
| --- |
| Code 28 |

**Show the model result comparison**



**Question:**

**1.** **In short sentence explain the following terms**

|  |  |
| --- | --- |
| **Terms** | **Description** |
| **Pennylane** | PennyLane is a software library that bridges quantum computing and machine learning, allowing users to develop and simulate quantum algorithms and quantum machine learning models. |
| **Quantum device** | In quantum computing, a quantum device is defined to specify the physical or simulated quantum hardware that will execute quantum operations. This includes details like the number of qubits, the type of qubits (superconducting, trapped ions, etc.), and the quantum gates available. Defining a quantum device is crucial because it determines the computational resources, the types of algorithms that can be run, and the accuracy of the results, thus ensuring that the quantum program is tailored to the capabilities and limitations of the specific hardware or simulator in use. |
| **Superposition**: | Quantum states can exist in a superposition, meaning a system can be in multiple states at the same time. For example, a quantum bit (qubit) can be in a state |0⟩, |1⟩ |
| **Wave Function**: | It contains all the information about the system and can be used to calculate probabilities of different outcomes. |
| **Measurement and Collapse**: | When a quantum state is measured, the superposition collapses to one of the possible states. |
| Entanglement | Entanglement is a phenomenon where quantum states of two or more particles become linked, such that the state of one particle cannot be described independently of the others. |
| **composite state** | A **composite state** in quantum mechanics refers to the combined state of two or more quantum systems. When you have multiple quantum systems (like two particles), each with its own quantum state, you can describe the total system using a composite state. |
| **Circuit** | a circuit is a sequence of quantum operations or gates applied to qubits to perform a specific computation or algorithm. It defines how quantum data is processed within a quantum algorithm. |

2. **Explain the Role of Feature Extraction in DDoS Detection**

|  |
| --- |
| Feature extraction from network traffic data helps a Deep Neural Network focus on the most relevant patterns, improving its accuracy in recognizing DDoS-type attacks. An important feature could be the number of requests per second from a single IP address. |

3. Describe the Advantage of Hybrid Quantum Neural Networks over classical Neural Networks

|  |
| --- |
| Hybrid Quantum Neural Networks can process complex patterns more efficiently than classical neural networks, potentially improving the detection of DDoS-type attacks as quantum computing technology advances. |

4. Explain the business impact of [DDOS attack](https://www.f5.com/labs/learning-center/what-is-a-distributed-denial-of-service-attack)

|  |
| --- |
| Financial loss    Loss of customers and customer confidence.    Threat of legal action    Reputation and goodwill |