|  |  |
| --- | --- |
| Insert title | |
|  | |
| Student | Main supervisor |
| - Luca Lucchina | - Angelo Consoli |
| Co-supervisors |
| - |
| Company (if any) |
| - |
| Degree course | Project identifier (if any) |
| Data Science and Artificial Intelligence | - |
| Year |  |
| * 2025 |  |

Table of Contents

[Abstract 3](#_Toc205634157)

[Chapter 1: Omniport Data Exploration 4](#_Toc205634158)

[1.1 Geographical features 4](#_Toc205634159)

[1.2 Time based features 6](#_Toc205634160)

[1.3 Payload Features 7](#_Toc205634161)

[Chapter 2: AI unsupervised approach 7](#_Toc205634162)

[2.1 Full dataset 7](#_Toc205634163)

[2.2 Grouped dataset 8](#_Toc205634164)

[Chapter 3: AI semi supervised approach 9](#_Toc205634165)

[3.1 Model trained on the cic and predicting on omniport 9](#_Toc205634166)

[3.2 Pseudo-Labeling configuration on CIC IDS 2017 9](#_Toc205634167)

[3.3 Omniport Pseudo-Labeling Implementation 19](#_Toc205634168)

[Chapter 4: Payload analysis 22](#_Toc205634169)

# Abstract

The goal of this project was to analyse and classify a series of unlabelled cyberattacks using payload characteristics and various attack details. A key challenge stemmed from the nature of cyber-attacks, as unsupervised approaches proved ill-suited due to the lack of clear cluster boundaries observed in labelled datasets. This prompted exploration of semi-supervised methods, which showed improved results but remained limited by the absence of true labels and the need to rely on external labelled datasets, introducing further uncertainty. A semi-supervised method was attempted which performed to a much higher level however the uncertainty and evaluation is not perfect caused by the absence of true labels complicated the evaluation process and the requirement of a different labelled dataset from Kaggle introduced additional errors and uncertainties. Further analysis was performed to understand the dataset and trends further: attack timings, locations, payloads. Showcasing minor trends, high amounts of empty payloads and corrupted payloads enhancing the difficulty of the clustering and classification.

# Chapter 1: Omniport Data Exploration

The Omniport dataset, provided by my supervisor Angelo Consoli, consists of nearly one million recorded interactions between cyber attackers and honeypot servers deployed in cities such as Singapore, Paris, Mexico City, and New York and various other locations worldwide. A honeypot is a deliberately exposed, isolated system or service designed to attract attackers, allowing researchers to observe malicious activity without putting real systems at risk.

## Geographical features

The data included the ip of the source and the destination, the country and city for both as well as the coordinates. Looking at the map shown in figure 1 we can see that there is little observable trend in the distribution of attack locations.

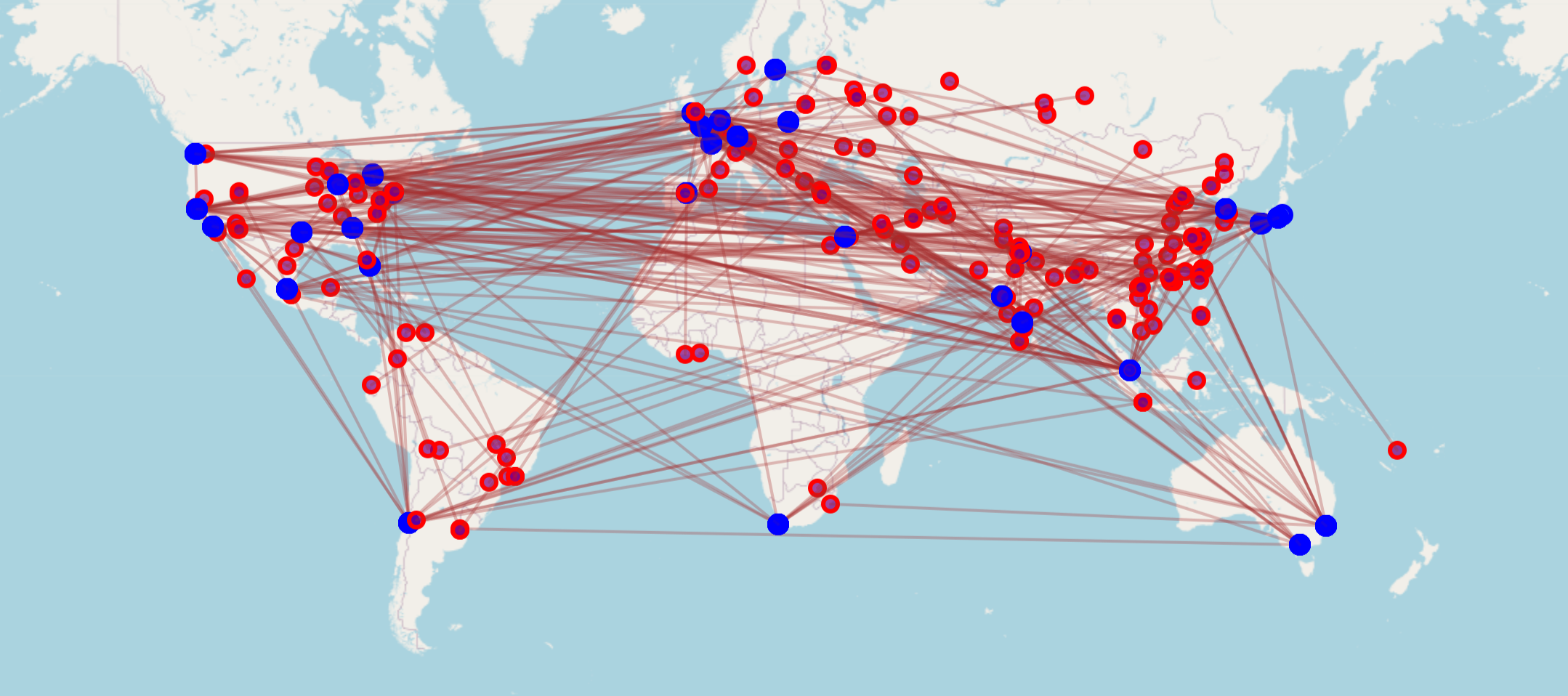


Fig 1: red dots being attackers and blue dots the honeypot servers showcasing little attacking trend

For the future models to perform better, I kept the IP addresses but split them into quartets so that each quartet is treated as its own feature, allowing each component to contribute its own detail. For the latitude and longitude, I applied cosine and sine transformations to remove the artificial linearity of degree values ranging from −180 to 180, and simplified it to the cosine representation, which effectively captures the circular nature of geographic coordinates and allows the model to better interpret proximity across the 180° boundary.

Noticeably there are a lot more attacks going out of China and the USA probably due to the high infrastructure and maybe potential motives, as shown in Figures 2 and 3.

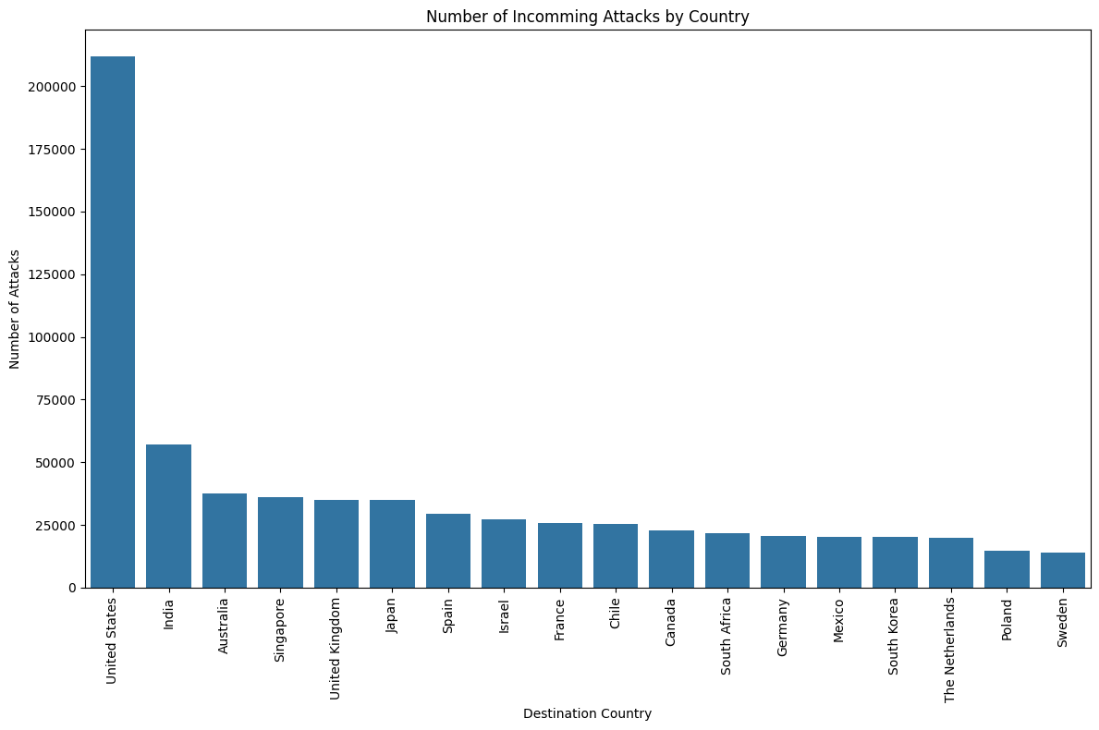


Fig 2: graph of incoming attacks, showcasing high number of incoming attacks for USA and India

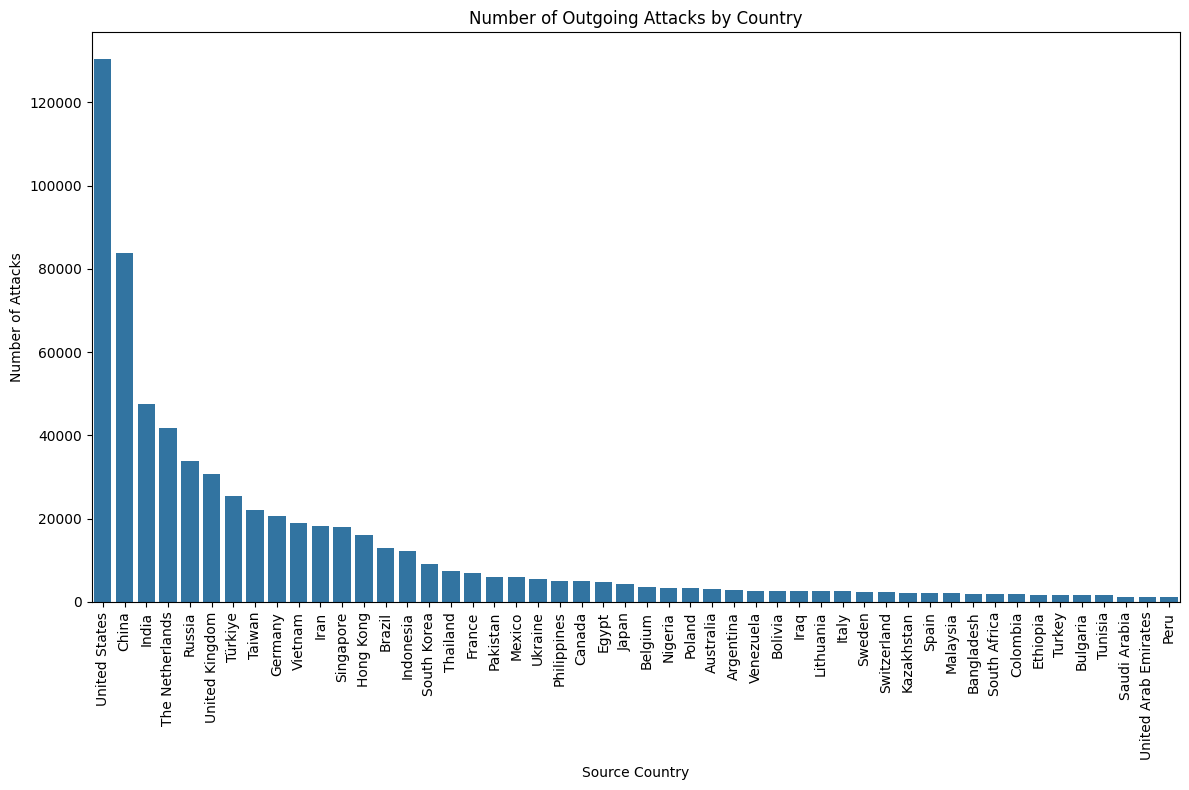


Fig 3: graph of outgoing attacks, showing high number of outgoing attacks for USA and China mainly

## Time based features

Time could be a very important element for prediction as humans inherently have patterns and preferences whilst bot net not really and work all through out the days. The issue that I came to find is that there aren’t really any patterns when looking at UTC time. I also thought that there would be trend on the local time of attack and the local time of when the attack would come, at night or non-workable days but looking at figure 4 we can see that such patterns aren’t present and there is a rather even distribution of attacks per hours and day of the week.

A close-up of a graph

AI-generated content may be incorrect.

Fig 4: distribution of attacks by hour of day and day of the week

Still thinking about the bot nets and normal human attack timing I looked at the time between attacks separating it between sub 1 second, 1-60 second, 1-60 minutes, 1-24 hours , >24 hours and unknown being only one interaction with this IP was recorded in the dataset. The median time and class distribution for the top 10 countries can be seen in figure 5.

* Sub-second (<1 second)
  + Attacks occurring less than 1 second apart usually indicates automated/bot attacks or DDoS attempts: very high frequency, likely machine-generated traffic
* 1-60 seconds
  + Attacks occurring between 1-60 seconds apart often indicates automated scanning or systematic probing probably too fast for human operation
* 1-60 minutes
  + Attacks occurring between 1-60 minutes apart could be both automated (with delays) or human-driven may indicate more sophisticated scanning with rate limiting
* 1-24 hours
  + Attacks occurring between 1-24 hours apart more likely to be human-driven or scheduled automated tasks could indicate daily routines or scheduled scans
* >24 hours
  + Attacks occurring more than 24 hours apart usually indicates sporadic attempts or long-term reconnaissance more likely to be human-driven or opportunistic attacks

A graph of different colored bars

AI-generated content may be incorrect.

Fig 5: attack timings for the top 10 countries in the dataset.

## Payload Features

## 1.4 Text embedding and Categorical encoding

# Chapter 2: AI unsupervised approach

## 2.1 Full dataset

Initial approach was doing clustering after understanding the dataset I encoded the features in this manner:

1. Separate the ip by quartets
2. Extract sin and cos of the lat and long
3. Time based attack analysis
   1. Extracting local time based on the latitude
   2. Added temportal features (hour of day, day of week, is weekend bool
   3. Extracted time between attacks and understood the distribution
4. Analyzed port scanning patterns by looking at:
   1. Sequential port access
   2. Port range coverage
   3. Standard deviation of port numbers
5. Added protocol categories (one hot encoded)
   1. Looked at the burstiness and periodicity of the attacks to see trends
6. Decoded the payload
   1. Analysed payload features (length, size in bytes, entropy)
   2. Analysed commands trend
7. Text embedding for the country and city location
   1. Pca variance optimization
8. Categorical feature encoding
   1. Autonomous system number
   2. True False

After this I took all the features, scaled them and clustered with k means and hdbscans evaluating the clusters with the silhouette score and visually with PCA and UMAP reduction. The results were unsatisfactory, and the high dimensionality led to considerably increased computation time, hampering effective clustering.

## 2.2 Grouped dataset

Seeing as each datapoint was a single attack (packet sent) I decided to group the attacks based on the dst and src ip and the dst port to have one row containing multiple packets sent overall being one cyber-attack over multiple packets. This allowed me to add features based on the group of attacks including idle time, number of packets per second number of packets over the duration, duration of the attack.

This reduced the data a bit helping with the extensive computation time of hdbscan and silhouette score. After testing I realized that the embeddings and onehot encoded protocol where useless and simply increased computational time so I removed them, I decided that by testing with and without, seeing that without the clusters being formed it would be better removing the occasions where the clusters would be formed based on the location of the attack, or time based whilst still keeping relevant features.

This lead to high silhouette up to 0.9 averaging 0.75 based on the initialization (fig 1), I did port analysis, number of attacks per group, and visually the clusters where good and well separated.

A graph of a plot

AI-generated content may be incorrect.

Fig 1 silhouette score of hdbscans

In results refer to this as even though the clusters where good comparing to other datasets that where labeled they where too different and not matched the labelled attacks.

# Chapter 3: AI semi supervised approach

### 3.1 Model trained on the cic and predicting on omniport

Initially I wanted to do direct predictions from cic to omniport. To be able to do that I needed to adapt the omniport dataset to cic-ids. The cic dataset was made from grouping a series of forward payloads, the grouped dataset is in that format, and it was made partly for that reason seen in 1.2 grouped dataset.

This method did not persist as the predictions where mostly on the same majority classes.

## 3.2 Pseudo-Labeling configuration on CIC IDS 2017

To simulate the pseudo-labeling pipeline needed for the OmniPort dataset provided by Consoli, I used a subset of the CIC-IDS 2017 dataset. From the roughly 600,000 available samples, I selected 10,000–20,000 labeled samples for training and treated the rest as unlabeled.

**🔹 Initial Approach**

My first attempt involved training a model on the labelled set, predicting on the unlabeled pool, and selecting high-confidence predictions using the softmax maximum. These pseudo-labeled samples were then added to the training set to retrain a final model. However, the results were disappointing — likely due to noisy pseudo-labels and the lack of any regularization or structure in the learning process.

After reviewing the pseudo-labeling paper [CRUPL (arXiv:2503.00358)](https://arxiv.org/pdf/2503.00358), I implemented key components it proposed, beginning with consistency regularization.

**🔹 Consistency Regularization**

Consistency regularization encourages models to produce stable predictions even under input perturbations. I added Gaussian noise to inputs for pseudo-labelled samples and trained the model to minimize the difference between predictions on clean vs. noisy inputs. This helped the model become more robust and improved performance marginally.

**🔹 Curriculum Learning**

Instead of pseudo-labelling everything at once, I introduced curriculum learning — running multiple training cycles with a gradually decreasing confidence threshold. This way, the model starts with the easiest (most confident) pseudo-labels and slowly includes harder ones.

To reduce the influence of potentially incorrect pseudo-labels, I used down-weighting:

labeled\_mask = np.concatenate([  
 np.ones(len(x\_labeled)),  
 np.full(len(x\_train\_curr) - len(x\_labeled), 0.3)  
])

This ensured that pseudo-labels contributed less to the total loss, especially in early steps.

Slight noise was also added to pseudo-labeled inputs to introduce variability, helping the model maintain stable predictions under perturbations.

**🔹 High-Confidence Sample Selection**

Initially, I selected all predictions above a confidence threshold. However, this often included overconfident but incorrect predictions, especially in imbalanced settings. The model, after a few steps, became heavily biased toward majority classes, ignoring minority ones. Fig 1shows this effect clearly.

A graph with green and blue lines

AI-generated content may be incorrect.

Fig 1 curriculum learning progress showing spike down after 4 steps

To improve this, I added entropy and confidence margin filters — only selecting samples with low entropy and large margins between the top softmax scores. I also limited how many samples could be added per step to prevent overwhelming the model.

As pseudo-labeling progressed, the model increasingly favored majority classes. After a few curriculum steps, pseudo-labeled data became dominated by only a few classes, making it harder to learn the minority ones.

Fig 2 illustrates this skew.

A graph with numbers and a chart with blue squares

AI-generated content may be incorrect.

Fig 2 confusion matrix of pseudo labels vs true labels

The accuracy plateaued around **0.4–0.6**, with **F1 scores around 0.3**.

**🔹 Fixing Class Imbalance in Pseudo-Labels**

I introduced per-step class balancing: at each step, the selected pseudo-labels had to include samples from all classes. This curbed the feedback loop where majority classes were reinforced each round.

Fig 3, 4, 5 show the results after applying this fix.

A black and white rectangular object with white text

AI-generated content may be incorrect.

Fig 3

A graph with green and blue lines

AI-generated content may be incorrect.

Fig 4

A graph with numbers and labels

AI-generated content may be incorrect.

Fig 5

Accuracy improved to **0.80** with **F1 score at 0.60**, though some very small classes were still underrepresented.

**🔹 Reducing Number of Classes**

To make the prediction easier, I decided to reduce the number of classes, this meant that there would be less distinction between types of attacks but increasing drastically the accuracy of the model. I decided to reduced the number of classes to the six most represented classes and remove the ones that had less than 200 samples. This improved both confidence calibration and class imbalance in pseudo-labels.

**Results:**

* Accuracy: **0.82**
* F1 Score: **0.77**

A graph showing the growth of a curriculum

AI-generated content may be incorrect.

Fig 6 curriculum learning process with only 6 classes

However, even in this simpler setup, the model showed a tendency to favor the majority class (e.g., class 5).

A graph with numbers and labels

AI-generated content may be incorrect.

Fig 7 confusion matrix labelling on 6 classes

**🔹 KL Divergence for Pseudo-Labeled Samples**

Later in the pipeline, I refined the loss function to treat labeled and pseudo-labeled data differently:

* **Labeled** samples use sparse\_categorical\_crossentropy
* **Pseudo-labeled** samples use **KL divergence** against soft pseudo-labels

This change allowed the model to better capture uncertainty and avoid overfitting to possibly incorrect one-hot targets. KL divergence aligns the output distribution to the pseudo-label distribution more softly, leading to improved stability in later curriculum stages.

This helped but overall the model was still overfitting after the first few steps which lead me to reduce the model’s dimensionality which stabilized a lot the learning in early stages as seen in figure below and then decrease in performance in later stages.

A graph showing the growth of a curriculum

AI-generated content may be incorrect.

Fig 8

A screenshot of a computer screen

AI-generated content may be incorrect.

Fig 9

This led me to think more in the curriculum learning formatting where in this case too many samples would be pseudo-labelled early on which cluttered the learning process early leading to some inaccuracies and as soon as some inaccuracies are introduced the errors build up. For that I limited the number of samples that can be selected greatly:

Initially:

* I would limit the selection to 1/8 of the total unlabelled dataset every step
* Have the previous per class distribution constraint

After:

* Over the first 6 steps the limit slowly grows to reach that 1/8 of total unlabelled dataset, as seen in figure below.

A graph showing the growth of a curvy line

AI-generated content may be incorrect.

Fig 10

Seeing this we can see the early step accuracy is very high and the accuracy drops down only when the number of selected samples become <10000 meaning that the last 3 steps the remaining samples are the hardest to learn, which makes sense that the accuracy lowers. This results in similar over all accuracy in pseudo-labelling, but the method is more robust. I tested with different initial labelled samples and I think the f1 score seems to be slightly better on average, the accuracy is about the same.

A screenshot of a graph

AI-generated content may be incorrect.

Fig 11

To increase f1 and more importantly robustness a parameter could be set to stop the curriculum learning when the number of selected samples spike down below a certain threshold (assuring that it doesn’t stop at the first few steps when the cap is low). This would keep confidence of the pseudo-labels high and the harder to predict samples could be simply ignored.

## 3.3 Omniport Pseudo-Labeling Implementation

Seeing as this worked, I attempted to work on the main problem which is labelling and identifying the type of attack of the Omniport dataset. Initially there was a chance to get a very small amount of data as labelled when that plan fell through, I had to rely on the cic dataset, a labelled dataset from Kaggle and clustering intuition and analysis to evaluate the labelling of the data as well as payload analysis.

This work was still usable as curriculum learning is still an efficient method of making predictions and robust models for label creation. For evaluating purposes, will be discussed later, I started by splitting the cic dataset into train test set, trained a curriculum learning model with all the added improvements seen above.

Omniport data with these I analysed the labels with respect of the whole of the Omniport set and compared to the the cic set.

From the whole of the labelled cic dataset I selected high confidence predictions as shown earlier. The result to some of this is the following:  
A chart of a diagram

AI-generated content may be incorrect.

Fig 12

A graph of blue and orange dots

AI-generated content may be incorrect.

Fig13

A screenshot of a graph

AI-generated content may be incorrect.

Fig 14

I looked at both the pure predicted labels clusters and the similarity of these values, labels to the cic data set. Noticeably the cic and Omniport datasets are not 100% the same but that was discussed earlier. However, I thought that they filled the gaps from each other. Looking at the predictions the label clusters are quite similar and much more promising than any unsupervised labelling method that was attempted earlier with dbscan and kmeans using different feature configuration.

This process was performed multiple times with different parameters to see the consistency of the predicted label. This test showed that regardless of the hyper parameters the labels would be very similar amongst each other meaning that the labels have quite a high confidence.

The visual method of evaluation was a good start additionally I thought of other methods to have a better evaluation of it:

1. A model trained on labeled data was used to generate pseudo-labels for the unlabeled set; these pseudo-labels were then used to train a new model, which was evaluated on the labeled data to assess how well the pseudo-labels capture the true class structure.
2. A model trained on the pseudo-labeled data was then used to predict on the unlabeled set, just to see how much its outputs lined up with the original pseudo-labels and whether they made any sense.

# Chapter 4: Payload analysis

The payload is a key component of a cyber attack meaning being able to analyse the payload in detail will lead to being able to detect the type of attack. A quick method for that would be any of the open-source models for cyber security, which take payload and other information to classify the attack. To analyse the payloads in detail, I explored open-source models for cybersecurity such as Cisco's Foundation AI. Most available payloads were corrupted and unreadable, significantly reducing the number of usable samples. Out of 470,000 valid payloads (those that were non-empty and non-duplicate), only about 170,000 were usable. Most of the usable payloads consisted of simple SSH or HTTP requests with minimal or empty content, such as “GET /hello HTTP/1.1”.

I did try to use cisco’s open-source model nonetheless to see the kind of results I could obtain. I got different results on the same query that I tried.

One of the payloads that seemed a bit more complex was the following. They were others with a full payload that where interpretable other than this one.

Payload: GET /libs/js/iframe.js HTTP/1.0   
User-Agent: xfa1   
Accept: \*/\*   
Host: 216.238.86.69

This returned different responses based on the model and the initialization:

“Scan  
The request for "iframe.js" in the /libs/js/ directory, combined with a non-standard User-Agent ("xfa1"), suggests a potential probe for known vulnerabilities or common file locations. This is characteristic of a scanning attempt.”

“AttackClass: WebAttack  
Explanation: The payload requests a JavaScript file (iframe.js) using HTTP/1.0 with a suspicious user-agent (xfa1). The filename and path (/libs/js/iframe.js) are commonly associated with malicious iframes used in drive-by download or redirection attacks, which are typical of web-based exploitation attempts.”

<https://huggingface.co/fdtn-ai/Foundation-Sec-8B>

It also had empty responses which I must have made a mistaken in the initialization of the model.