Scuola universitaria professionale della Svizzera italiana
Dipartimento tecnologie innovative

SUPSI

Clustering and classification of cyberattacks

Student
- Luca Lucchina

Main supervisor
- Angelo Consoli

Degree course
Data Science and Artificial Intelligence

Year
- 2025

Table of Contents

Abstract

Cyberattacks have become highly sophisticated mechanisms used daily to halt the normal functioning of electronic systems in all sectors of the economy. The primary objective of this project was to analyse and classify a large set of unlabelled cyberattacks, leveraging payload characteristics to determine their type and key attributes. One of the main challenges was that unsupervised learning approaches proved ineffective due to the absence of well-defined cluster boundaries in labelled reference datasets. This limitation motivated the adoption of a semi-supervised learning strategy, which yielded improved classification performance, but was still constrained by the lack of ground-truth labels and the necessity of relying on an external labelled dataset introducing additional uncertainty.

To enhance attack understanding and dataset analysis, the study examined multiple features beyond payload content, including attack timings, geographic origins, ASNs, and port identifiers. By integrating these features, I developed a robust pseudo-labelling methodology that, despite inherent uncertainty, provides sufficiently reliable labels to identify and isolate attacks. This could allow automatic alerts to security administrators and the activation of appropriate defensive protocols upon the detection of new attacks.

ITALIANO

Gli attacchi informatici rappresentano oggi strumenti di crescente sofisticazione, impiegati sistematicamente per compromettere il regolare funzionamento dei sistemi informativi in una vasta gamma di contesti economici. Il presente lavoro si propone di analizzare e classificare un ampio vasto insieme di attacchi a honeypots, attraverso l'analisi delle caratteristiche del payload al fine di determinarne la tipologia e gli attributi distintivi. Una delle principali criticità emerse nel corso dell'analisi riguarda la limitata efficacia degli approcci di apprendimento non supervisionato, dovuta all'assenza di cluster pre-definiti in labeled dataset di riferimento. Tale problematica ha motivato l'adozione di un paradigma di apprendimento semi-supervisionato, che ha permesso di raggiungere classificazioni superiori rispetto ai metodi puramente non supervisionati. Questo approccio resta comunque limitato e ha portato alla ricerca di una soluzione tramite etichette di riferimento e labeled dataset esterni (in particolare, provenienti dalla piattaforma Kaggle), che però hanno implkicato un ulteriore margine di incertezza. Al fine di migliorare la qualità dell'analisi e lo studio degli attacci, è sono stati studiati aspetti alternativi al contenuto del payload, quali ad esempio la temporalità degli attacchi, le coordinate geografiche degli attaccanti, gli Autonomous System Numbers (ASN) e la natura dei servizi (porte di comunicazione) sotto attacco. L'integrazione di tali caratteristiche ha consentito lo sviluppo di una metodologia robusta di pseudo-labeling, che, pur in presenza di un certo grado di incertezza intrinseca, ha permesso di generare etichette (labels) sufficientemente affidabili per l'identificazione, lo studio e la classificazione degli attacchi. Questo approccio apre prospettive promettenti per l'automatizzazione dei sistemi di rilevamento e clusterizzazione, rendendo potenzialmente possibile l'invio tempestivo di notifiche agli amministratori dei sistemi e anche l'attuazione di procedure difensive adeguate in risposta alla rilevazione cyber-attacchi.

Chapter 1: Omnipot Data Exploration

The Omnipot dataset, provided by my supervisor Angelo Consoli, consists of 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 “decoy system or network designed to attract and trap cyber attackers, diverting them from real targets and allowing you to study their methods and gather valuable threat intelligence” (What Are Honeypots (Computing)?, 2025).

1.1 Geographical features

The data included the IP of the source and the destination, the country and city for both as well as the coordinates, any reference of IP or coordinates from the dataset are hidden/removed as the honeypot system is still in use. Looking at the map shown in figure shows that there is little observable trend in the distribution of attack locations.

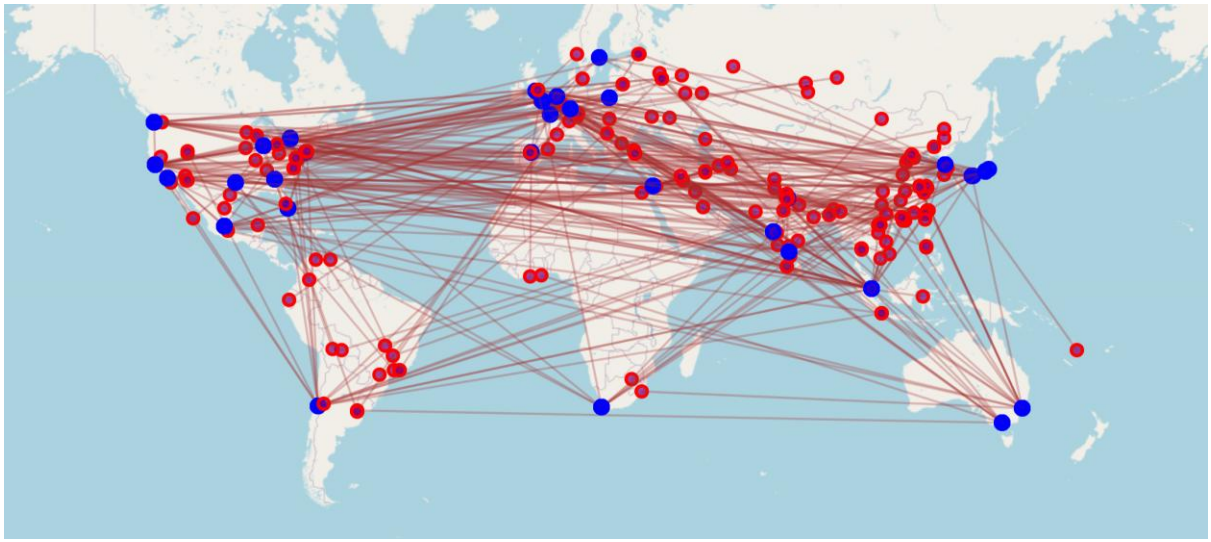

Fig 1: red dots being attackers and blue dots the honeypot servers.

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

1.1.1 Location based patterns

China and the USA are the most frequently targeted countries, with India and the Netherlands not far behind. This is due to the high number of servers in them leading to, naturally, more attacks, as shown in Figures 2 and 3.

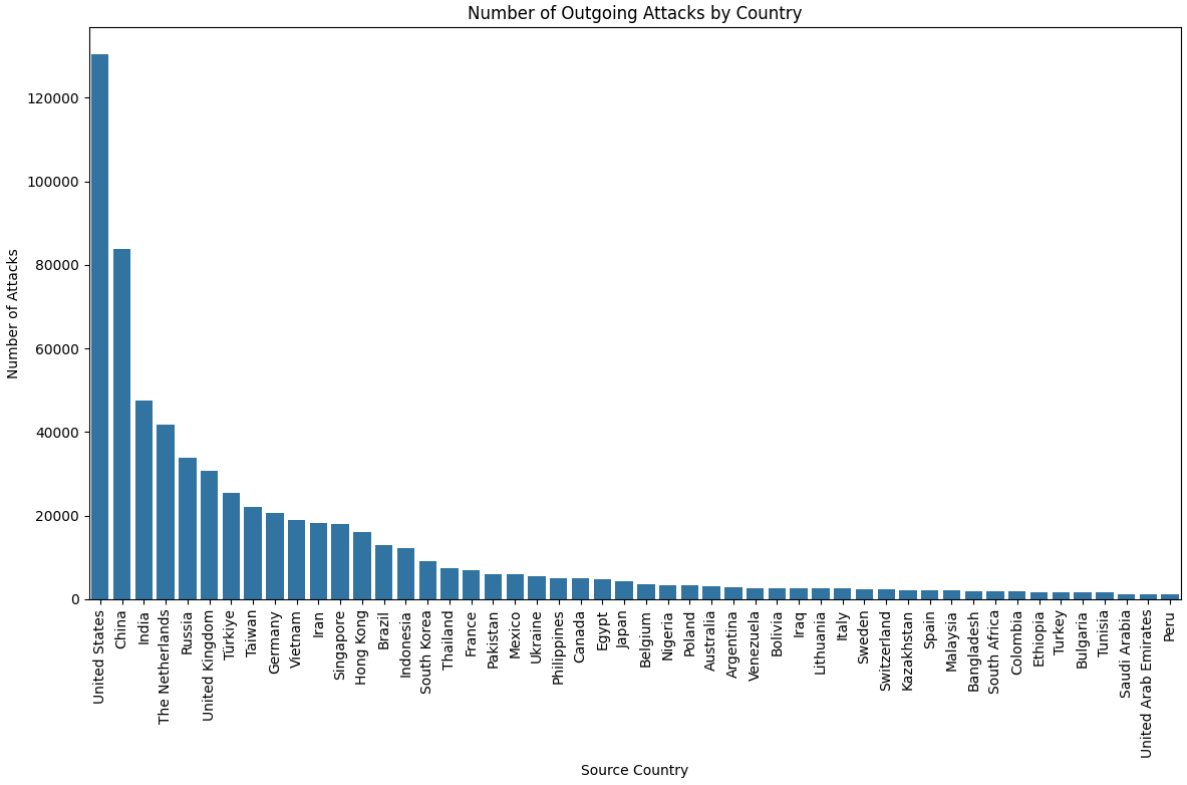

Fig 2: Graph of outgoing attacks, showing high number of outgoing attacks for USA and China mainly.

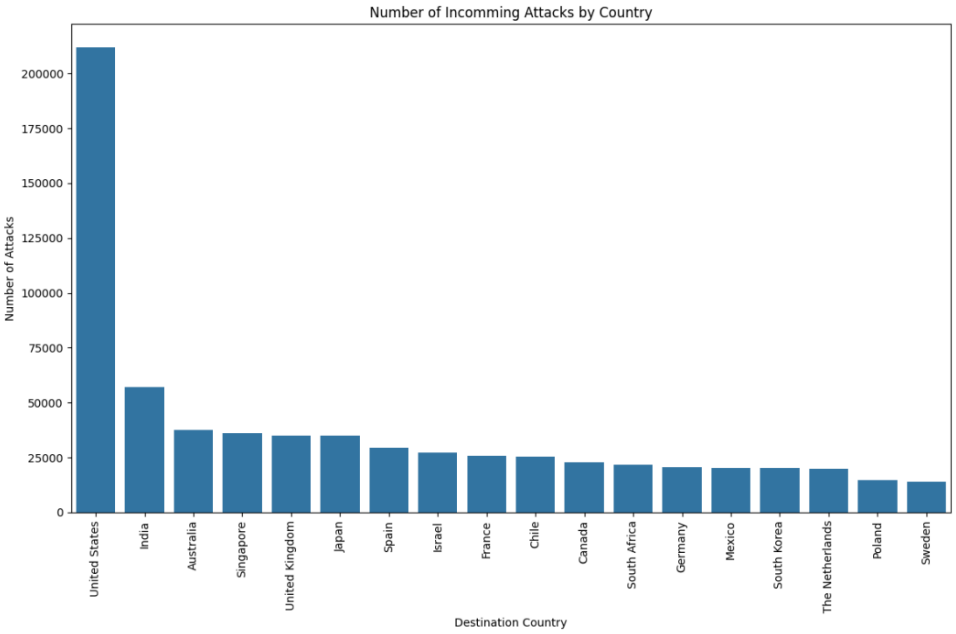

Fig 3: graph of incoming attacks, showcasing high number of incoming attacks for USA and India

1.2 Time based features

Time can be a crucial aspect in the prediction of the type of attack. Humans inherently have patterns and preferences whilst bot network do not. Unfortunately, I found that there are no major patterns when looking at UTC times. I thought there would be patterns with local time of attack: at night or non-workable days but looking at figure 4 illustrates that such patterns are only very slightly present. The timing for cyber-attacks is evenly spread across the week.

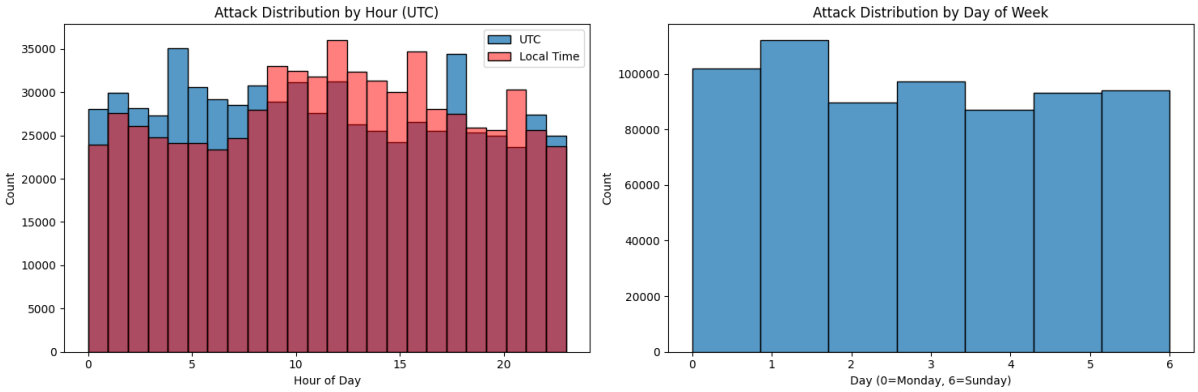

Fig 4: distribution of attacks by hour of day and day of the week

Bot networks are programmed to set intervals between cyber-attacks, compared to human attack timing which can vary drastically. Therefore, I looked at the time between individual attacks separating it the following classes: sub 1 second, 1-60 second, 1-60 minutes, 1-24 hours, >24 hours and unknown (occurring when only one interaction with this IP was recorded in the dataset). The median time and class distribution for the top ten 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: extremely high frequency, certainly machine-generated traffic.
- 1-60 seconds
    - Attacks occurring between 1-60 seconds apart often indicates automated scanning or systematic probing, likely machine-generated traffic, could be human.
- 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 indicating 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.

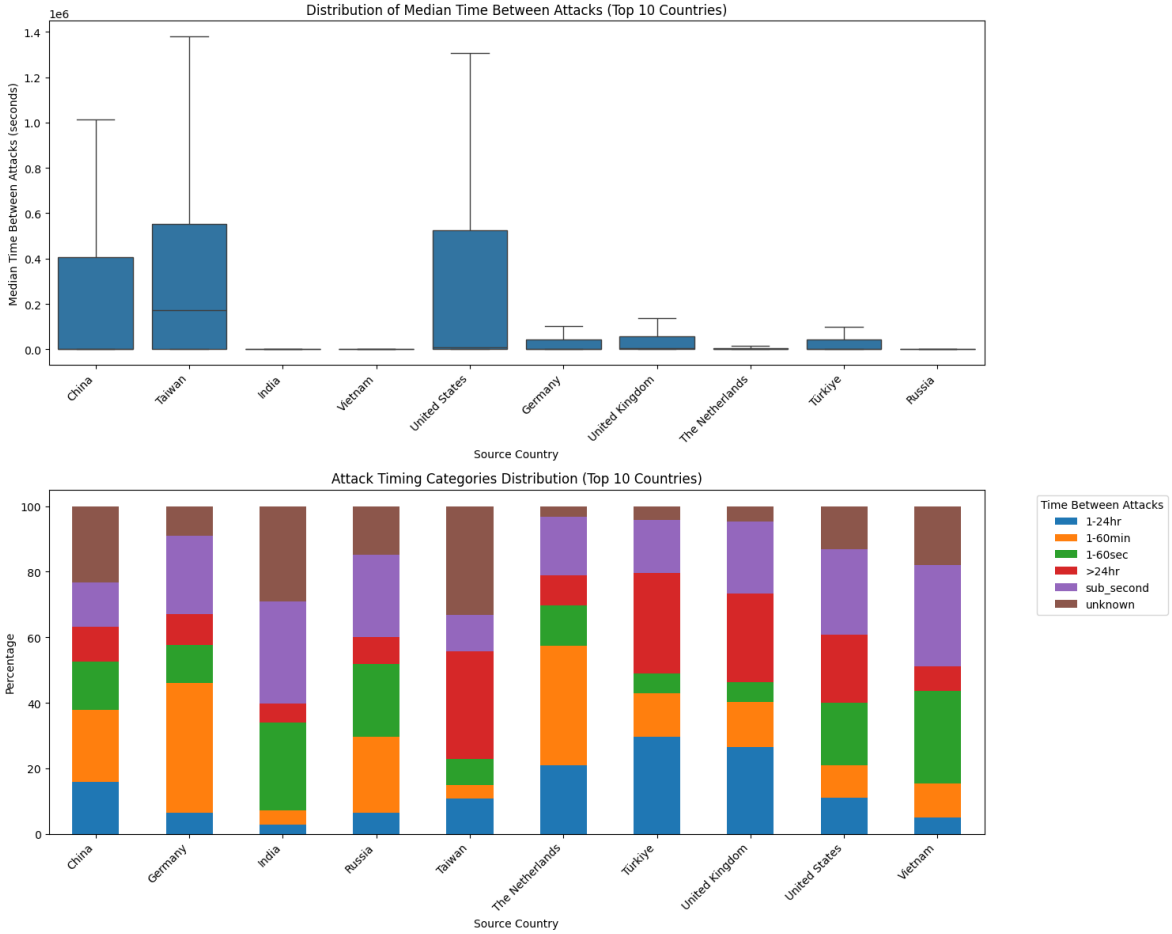

Fig 5: attack timings for the top ten countries in the dataset

1.3 Payload Features

The payload was managed in two ways: feature extraction from the encoded payload, and analysis of command occurrences. For encoded features, I extracted payload length (pl_length), unique bytes (pl_unique_bytes), entropy (pl_entropy), average byte value (pl_mean_byte), and byte standard deviation (pl_std_byte).

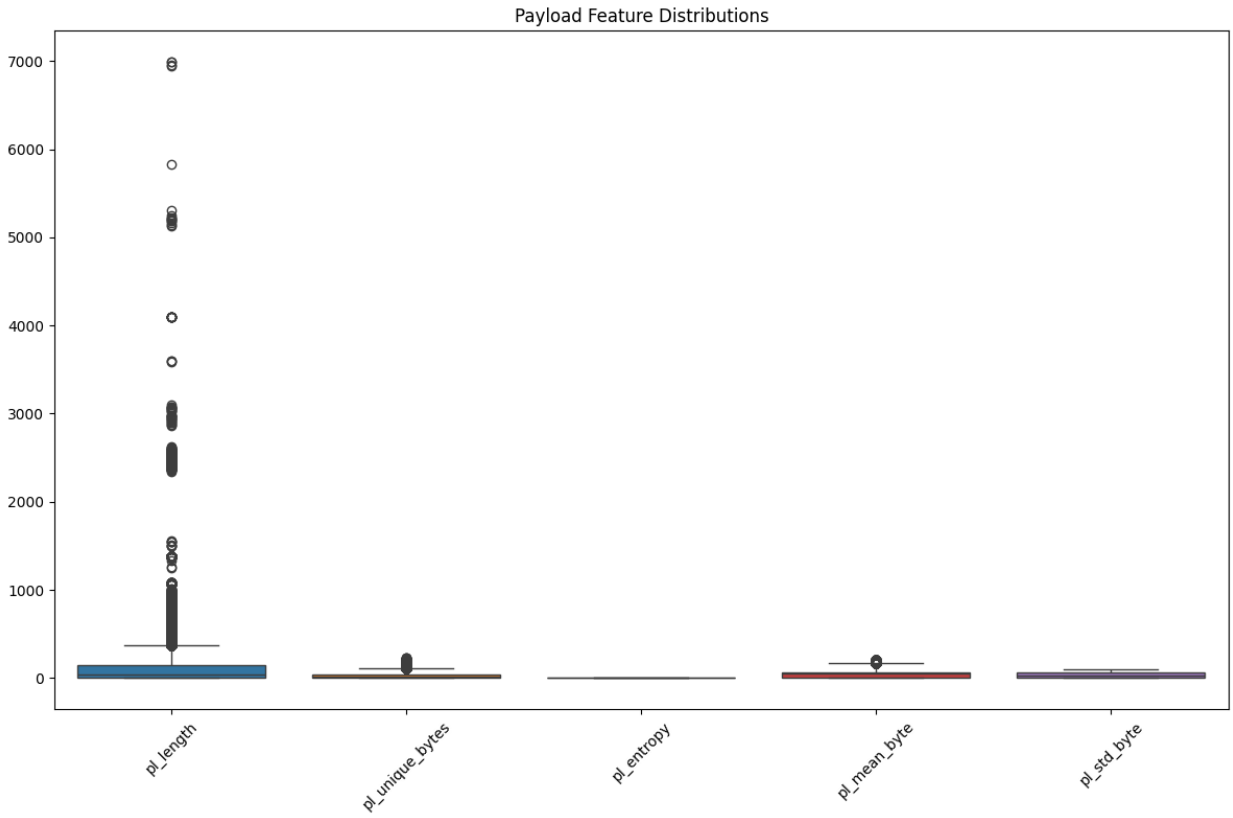

Fig 6: Box plot distribution of payload features

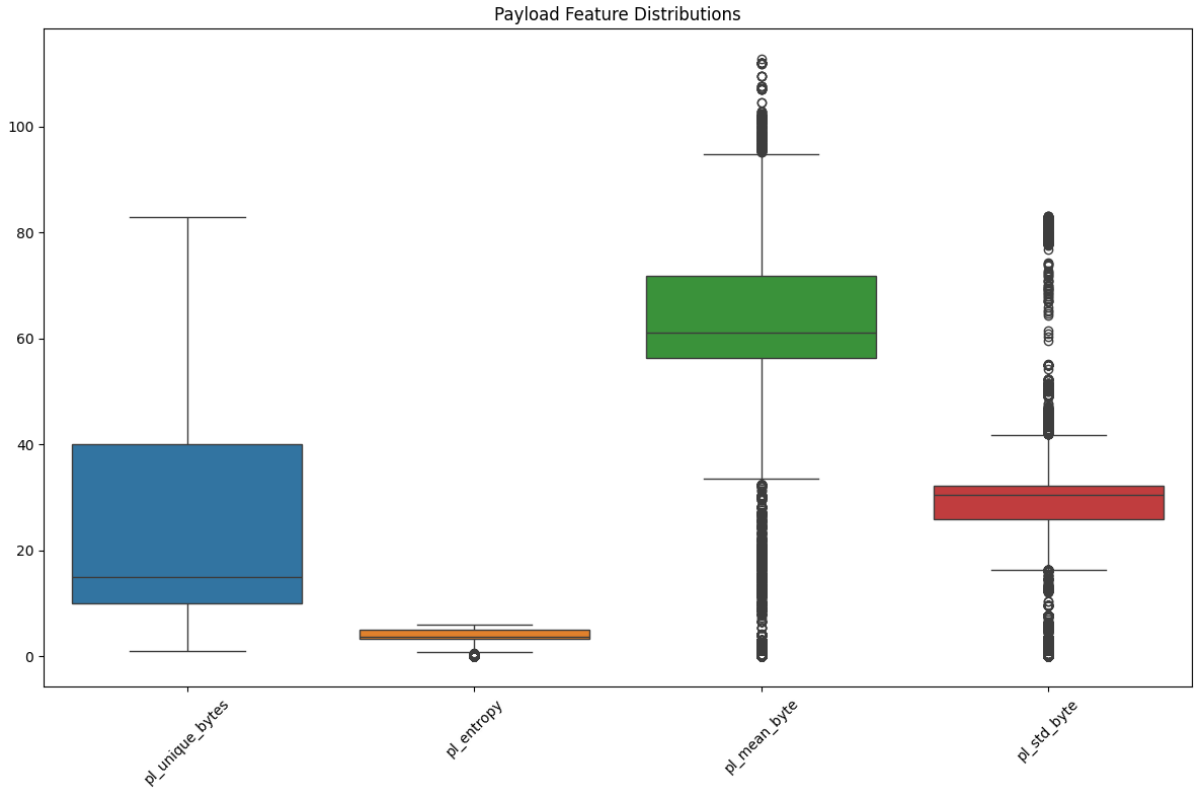

Fig 7: Payload feature distribution removing the large length feature

As shown in Figure 6, the payload length has large payloads (up to 7,000 bytes), which compresses the distribution of all other features. For this reason, a second plot was generated excluding pl_length (Figure 7). This allows a clearer view of the distributions of the remaining features. The payloads number of unique bytes shows a widespread, reflecting payloads with highly variable byte diversity. Payload entropy remains tightly concentrated around lower values, suggesting limited randomness in most payloads. In contrast, the mean number of bytes and the standard deviation exhibit broader distributions, highlighting variability in byte composition across payloads.

While the payload length is excluded from the main comparative visualization, it remains a valuable feature for differentiating between small payload, probing requests and unusually large, potentially file-transfer or binary-oriented payloads.

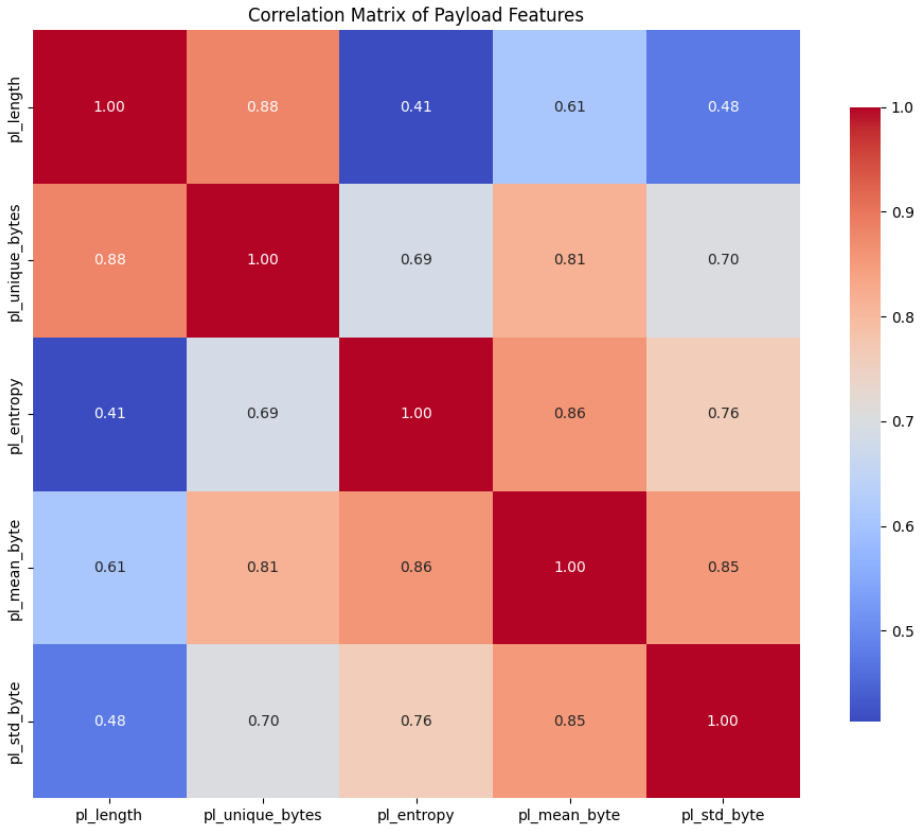

Fig 8: correlation matrix of payload features

The correlation matrix in Figure 8 confirms that the extracted payload features are strongly correlated, which is expected given their shared dependence on payload size and composition. For example, longer payloads naturally contain more unique bytes, and higher mean byte values are associated with greater variability (std). Entropy also aligns positively with other variability measures, reflecting its role in capturing randomness in the byte distribution. These correlations just confirm that the payloads are coherent rather than providing new information.

1.4 Text embedding

The dataset contained two primary sources of text prone to embedding: the decoded payloads and the location-based attributes (city, country, and region names). Embedding the location features was done with a generic model. It was done as language model (LLM) discern non-spatial patterns and enrich the data with additional contextual details.

On the other hand, embedding the payload was more of a challenge due to a combination of factors, including unreadable and empty payloads, as well as the need for an LLM embedding method tailored to the language.

CodeBERT is a pre-trained model on Python, Java, JavaScript, PHP, Ruby, and Go. Which seemed the most suitable given it has an understanding of coding synthetics compared to a general-purpose model. However, this method was sparingly used in the payload analysis due to the prohibitively long inference time required for embedding, which outweighed its potential benefits for this project. Companies such as CISCO have implemented and are using LLMs fine-tuned on cyber-attacks as primary protection. This will be further analysed in Chapter 4.

1.5 Categorical encoding

The Autonomous System Number (ASN), represented by 'src_as', was encoded using a frequency-based approach given to its potential implication. The number of occurrences of each ASN was mapped to a new feature, 'src_as_encoded', to capture the prevalence of specific ASNs in the dataset.

Chapter 2: AI unsupervised approach

2.1 Full dataset Clustering

For the clustering process I used both K-Means and HDBSCAN algorithms. The quality of the clusters was evaluated using the Silhouette score and visual inspection of dimensionality-reduced data using PCA and UMAP. “The silhouette value is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation).” (Davies, *KMEANS clustering, 2024)*

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

$a(i) =$ average distance between point i and all other points in the same cluster (cohesion)

$b(i) =$ average distance between point i and all points in the nearest cluster (separation)

The initial clusters seemed cluttered and not well defined. K-Means seemed only to be effective in identifying two to four clusters (Figure 9), which is not many if the distinction should be made by the type of attack.

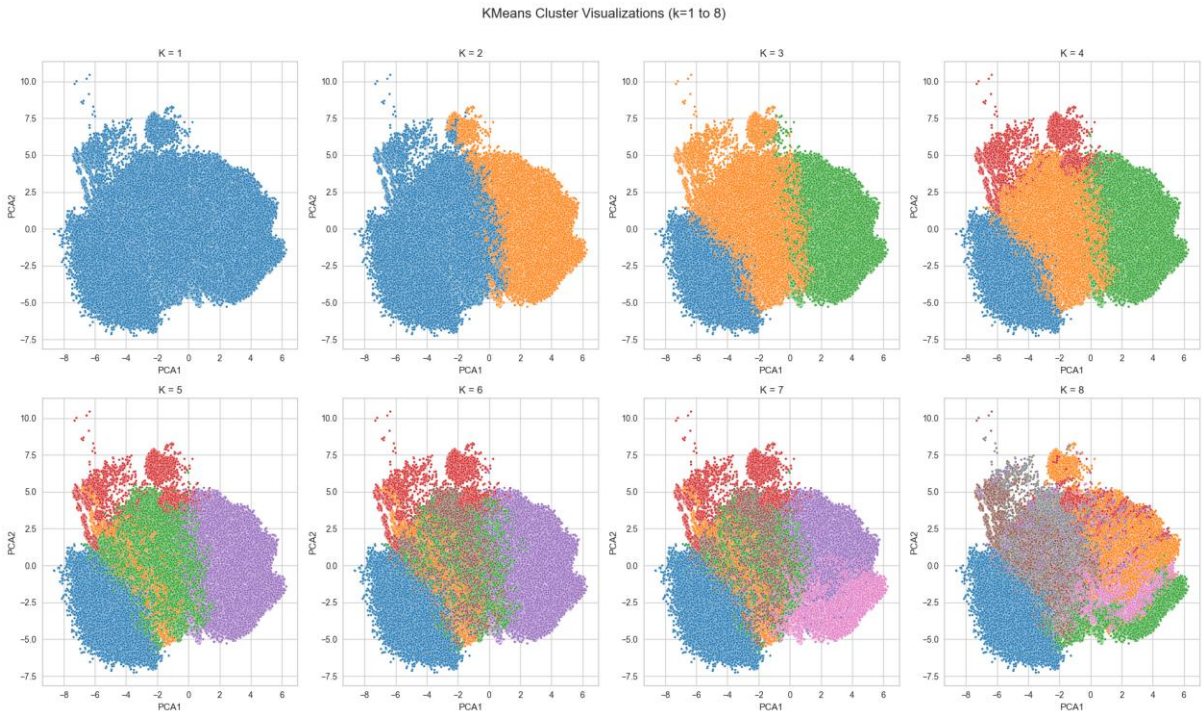

Fig 9: K-Means cluster visualization

HDBSCAN is a hierarchical clustering algorithm, it uses density-based connectivity to identify clusters of varying densities, providing a more flexible approach than traditional clustering methods. The ideal number of clusters for the data is determined by the algorithm, unlike k-means which can be regulated.

This method is a significant help to make very efficient and realistic clusters based on data. In my case the algorithm selected sixty-one clusters, which is considerably more than the simpler K-means algorithm, this is because it analyses in more depth the data. It suggested that the data was too complex, and it was not clustering based on the type of attack but other patterns. This makes sense as I had around three hundred features, a vast majority being text embeddings. The silhouette score for these clusters was extremely low if not negative on some occasions, being visually understandable seeing the cluster visualization in figure 10 below.

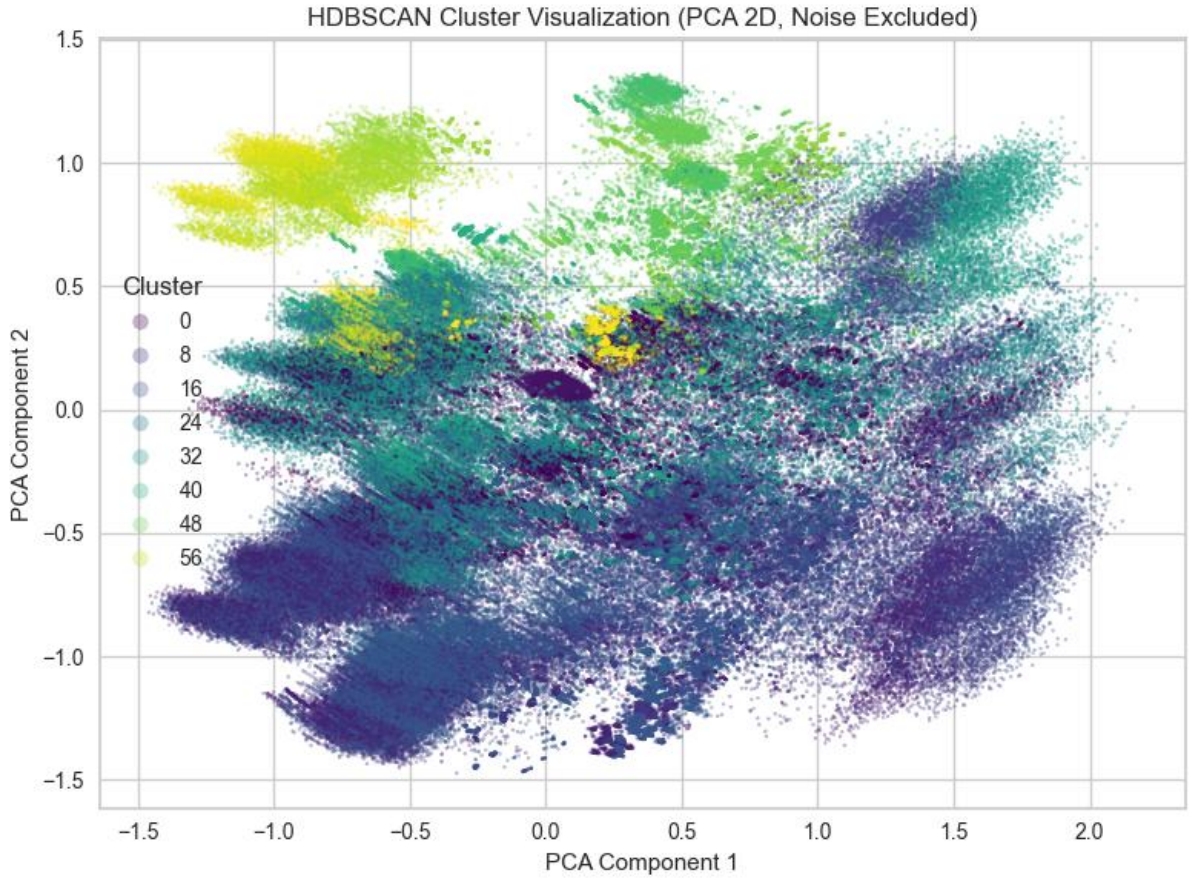

Fig 10: HDBScan PCA reduction visualization

2.2 Grouped dataset

The main issue with the above method was the high computational time due to high dimensions of the dataset. To remove that problem I decided to group the attacks based on the source and destination IP, where each package sent needs to have at maximum a 2-minute interval with the previous package. This reduced the number of samples, grouping them by burst of attacks. It also grouped the probing or mistakes in the attack process (noisy samples as well). Also allowing the addition of group-based features: time between attack, byte/s, duration of the attack, average packet length, number of packets during the attack as the main ones.

2.2.1 Feature selection and test

After testing I realized that the embeddings and one hot encoded protocols were simply increasing the complexity, computational time and blurred the clusters together, so I removed them. This modification led to high silhouette up to 0.9 averaging 0.75 based on the initialization (figure 11).

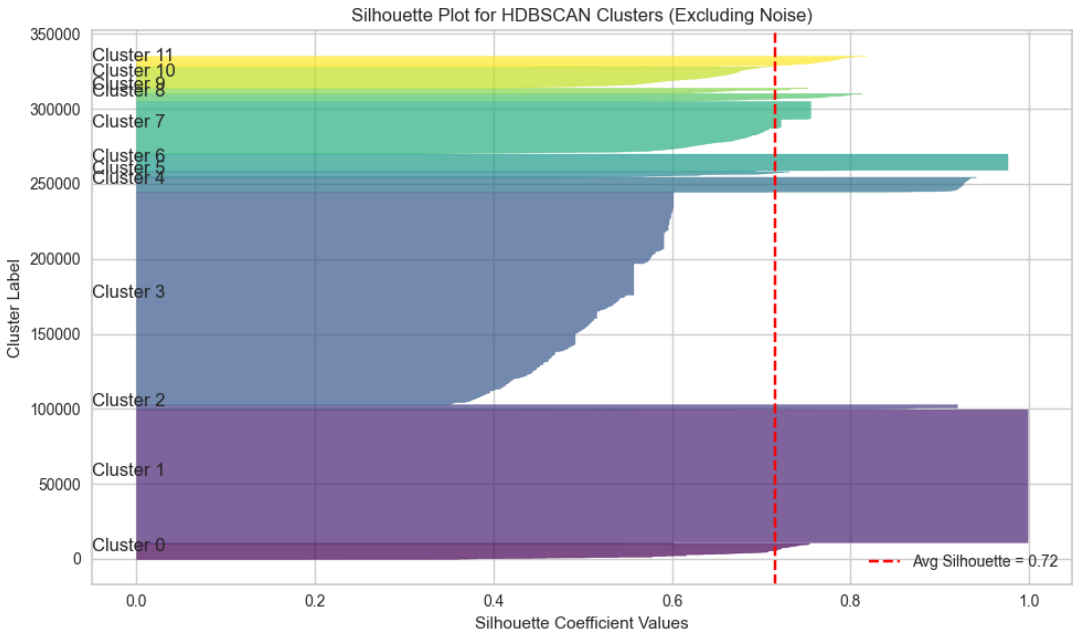

Fig 11: silhouette score of hdbscans

The dimensionality reduction to graph the samples led to more defined groups of data highlighting better cluster visually, figure 12 below.

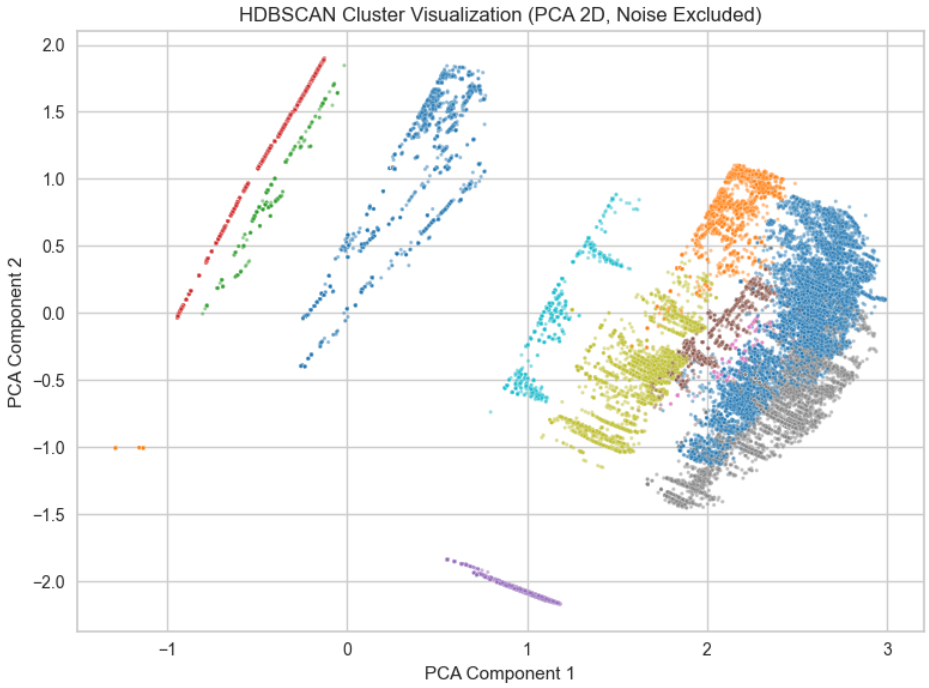

Fig 12: PCA reduced HDBScan cluster

Chapter 3: AI semi supervised approach

3.1 Model trained on the CIC and predicting on Omnipot

The initial approach consisted of training a supervised model on the CIC-IDS 2017 dataset and subsequently applying it to the Omnipot dataset. In order to achieve this objective, the Omnipot dataset was adapted to align with CIC-IDS by aggregating forward packet streams into bursts and extracting equivalent statistical features, as described in Section 2.2.

The CIC-IDS dataset is a benchmark intrusion detection dataset generated in a controlled testbed, where researchers simulated both normal network traffic and a variety of cyberattacks. The controlled and artificial nature of its design ensures precise labelling and reproducibility but also means it may not fully capture the complexity and unpredictability of real-world network environments.

However, the straightforward method exhibited significant bias toward the majority class, indicating a lack of generalization beyond CIC-specific traffic patterns. The crux of the issue pertained to the disparity in characteristics between the CIC and Omnipot datasets, which hindered the efficacy of the fundamental approach in facilitating accurate identification.

In light of this limitation, the focus was shifted toward semi-supervised learning, commencing with pseudo-labelling on CIC-IDS 2017 (see Section 3.2).

3.2 Pseudo-Labelling configuration on CIC IDS 2017

Pseudo-labelling is a method of assigning labels to samples without ground-truth guarantees. The main challenge lies in generating labels with the highest possible confidence.

3.2.1 Initial Approach

The design of the pipeline was initiated with a comprehensive review of extant literature on the subject of pseudo-labelling in the context of cybersecurity. The present study was informed by two papers: CRUPL (http://arxiv.org/abs/2503.00358) introduced a curriculum-based refinement strategy for pseudo-labelling, and Cyber Attack Detection with Semi-Supervised Learning (http://arxiv.org/abs/2001.06309v1) delved into more concrete labelled cyber attack detection. These works highlighted techniques and steps that should be taken to ensure efficacy of predicting, pseudo-labelling if labels are not available.

The initial setup of the CIC-IDS 2017 dataset served as a foundation for the subsequent analysis. In a sample size of approximately 600,000, a subset of 10,000–20,000 was selected for labelling, while the remaining data was designated as unlabelled. This enabled the simulation of a realistic semi-supervised scenario, where the acquisition of labelled attack traffic is costly and time-consuming, but unlabelled traffic logs are abundant. The overall pipeline can be summarized as follows:

1. **Train initial model**
2. **Predict on unlabelled data**
3. **Select confident samples** – Select high confidence samples where the maximum softmax probability exceeds τ, threshold set manually.
4. **Expand training set** – Add these samples to the training set.
5. **Retrain model** – Train a new model on the expanded dataset and sample the rest of the data

3.2.2 Consistency Regularization with selective noise Augmentation

During the training process, Gaussian noise was introduced to the pseudo-labelled samples, while the labelled samples remained untouched. This approach encouraged the model to generalize rather than memorize noisy labels, ensuring it learned more robust patterns instead of overfitting.

Building on this idea, consistency regularization, a common semi-supervised learning technique, was employed to further improve model robustness. By constraining the model to produce stable predictions under small input perturbations, consistency regularization complements the noise injection strategy. In the present implementation, this concept was extended through the integration of a selective consistency model (SelectiveConsistencyModel), which explicitly differentiates between labelled and pseudo-labelled samples.

The model is designed to minimize a combined loss consisting of:

1. **Supervised loss:** standard cross-entropy on labelled samples, weighted by sample confidence and class balance.
2. **Pseudo-label loss:** a KL-divergence term on pseudo-labelled samples, weighted lower to reflect uncertainty.
3. **Consistency loss:** the mean squared difference between predictions on original and perturbed pseudo-labelled inputs, encouraging stable predictions under input perturbations.

The total training loss is computed as:

$$L\_total = L\_supervised + \lambda\_consistency \cdot L\_consistency,$$

Where $\lambda_{consistency} = 0.5$ balances the contribution of the consistency term. This approach guarantees that pseudo-labelled data contributes to learning in a controlled manner, thereby enhancing robustness, and mitigating the risk of reinforcing erroneous pseudo-labels.

3.2.3 Curriculum Learning

The application of noise to uncertain samples, in conjunction with pseudo-label weighting, facilitates the model's learning of a more stable decision boundary. This, in turn, enables the gradual incorporation of harder pseudo-labelled examples in a curriculum-like fashion.

Inspired by the mechanism described in *CRUPL: A Semi-Supervised Cyber Attack Detection with Consistency Regularization and Uncertainty-aware Pseudo-Labelling in Smart Grid*, I implemented a similar process for learning.

The central idea is to gradually introduce pseudo-labelled samples into training, starting with the most confident predictions and progressively including harder (less confident) ones. This staged approach reduces the risk of reinforcing incorrect pseudo-labels and allows the model to learn from easier examples first, stabilizing the decision boundary.

In practice, this was realized by performing multiple training cycles with a confidence threshold schedule: high-confidence pseudo-labels are selected in early cycles, while lower-confidence ones are incorporated in later cycles. To further control the influence of potentially noisy pseudo-labels, a sample-weighting mechanism was applied:

- **Labelled samples** were assigned full weight (1.0) to fully contribute to the loss.
- **Pseudo-labelled samples** were assigned reduced weight (0.3–0.5), with the exact value reflecting both their confidence and stage in the curriculum.

Furthermore, class imbalance was addressed by the calculation of balanced sample weights derived from the target labels.

This mechanism ensures that the model utilizes the most reliable pseudo-labels first and progressively expands its supervision to examples with less certainty. The model integrates curriculum learning with selective weighting, thereby facilitating a controlled, progressive semi-supervised learning process. This approach reduces errors from incorrect pseudo-labels while still making full use of the unlabelled data.

3.2.4 High-Confidence Sample Selection

Initially, all pseudo-labelled predictions above a fixed confidence threshold τ were included in the curriculum. That is, for a predicted probability vector $\hat{y}$ over the available classes, a sample was selected if

$$max(\hat{y}) \geq \tau$$

However, this approach occasionally allowed overconfident and wrong predictions, particularly in imbalanced class scenarios, if the model is unsure about the class the sample should belong in. This led to the model overfitting to majority classes, which was not solvable by class balancing. After just four curriculums steps the majority of the samples would be in two to three classes.

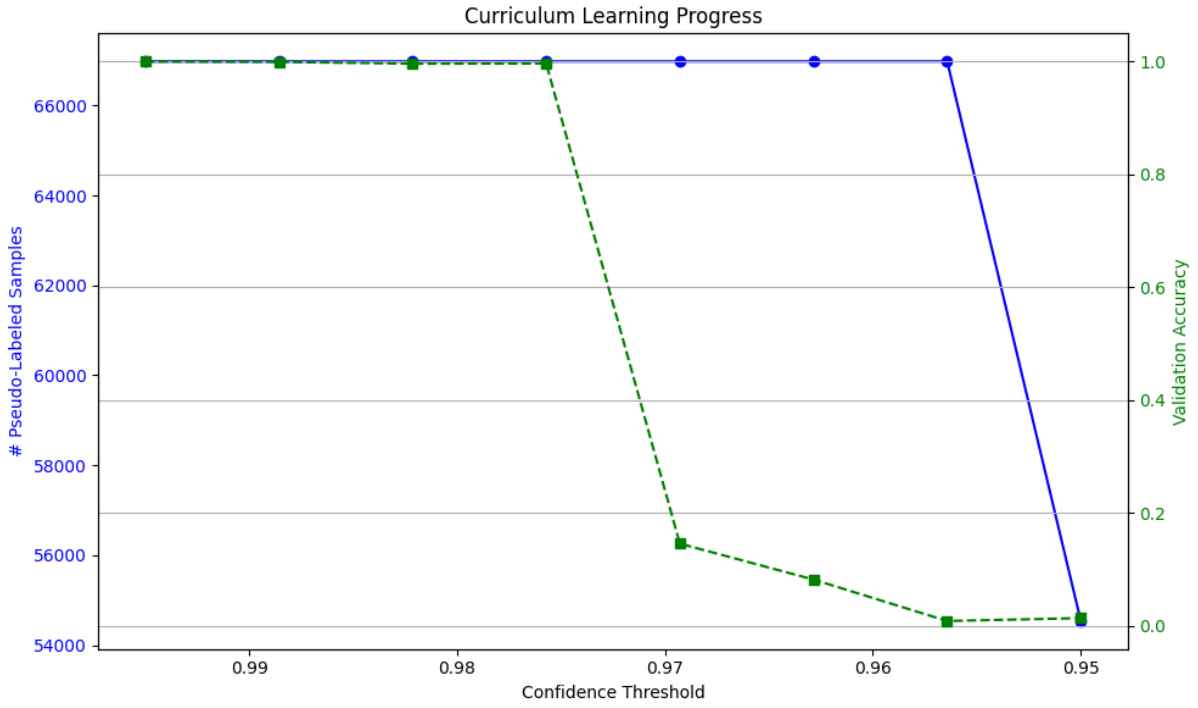

Fig 13: curriculum learning progress showing spike down after 4 steps as the majority samples where “used up”

In an attempt to solve this over the subsequent steps two additional criteria were put in place.

1. **Entropy filtering** measures the uncertainty of the prediction. For a probability ŷ, the entropy is computed as:

$$H(\hat{y}) = -\sum_{c=1}^{C} \hat{y_c} \log \hat{y_c}$$

Lower entropy indicates that the prediction is sharp and unambiguous. Only samples with entropy below a threshold H_{max} were retained, reducing the likelihood of including ambiguous or noisy pseudo-labels. In our experiments, H_{max} was set between 0.2 - 0 .4, corresponding to high-confidence, low-uncertainty predictions.

2. **Confidence margin filtering** examines the difference between the highest and second-highest Softmax probabilities:

$$\Delta_{\text{margin}} = \widehat{y^{(1)}} - \widehat{y^{(2)}}$$

where $\widehat{y^{(1)}}$ and $\widehat{y^{(2)}}$ are the top two predicted probabilities. Only samples with $\Delta_{\text{margin}} \geq \delta$ were selected, where δ controls how distinct the model’s top prediction must be relative to alternatives. In effect, this ensures that pseudo-labels are not only high probability but also clearly separated from other classes. I used δ = 0.3 testing different values when experimenting.

Finally, a cap on the number of pseudo-labelled samples per curriculum step was introduced to prevent overwhelming the model with too many new samples at once. This staged addition

ensures a gradual, controlled learning process, combining confidence, uncertainty, and relative margin to improve the quality of pseudo-labels while mitigating bias and error propagation.

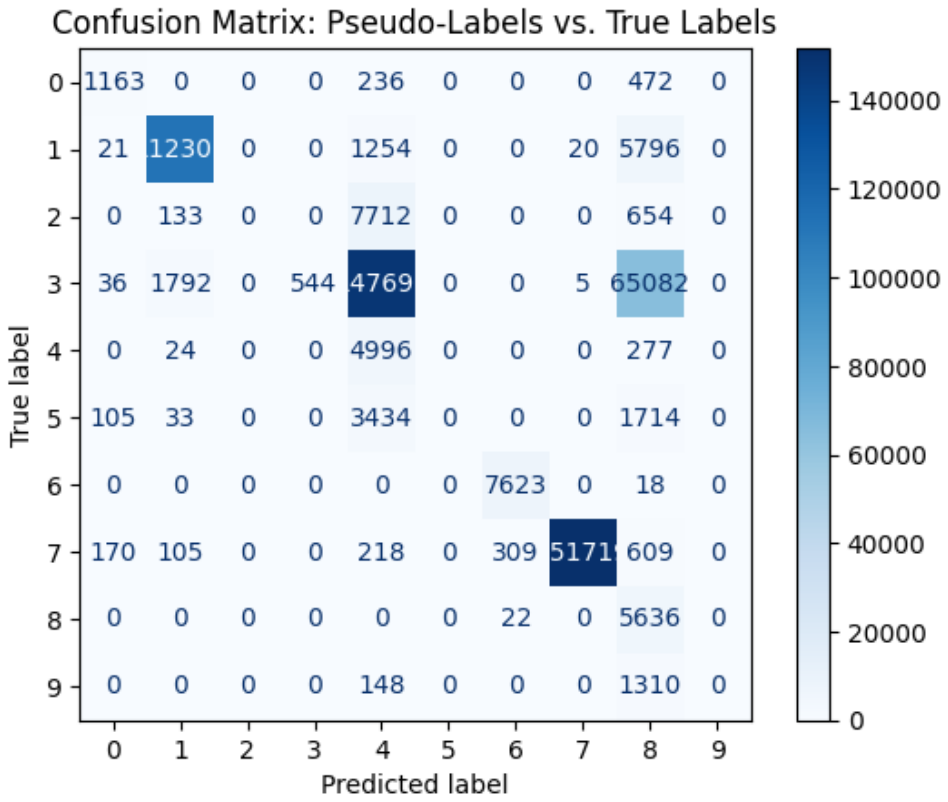

Fig 14: confusion matrix of pseudo labels vs true labels

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

3.2.5 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 15, 16, 17 show the results after applying this fix.

# label_encoded	# count
3	8177
7	5785
1	4643
2	383
6	286
8	209
4	205
5	190
0	65
9	57

Fig 15: label distribution

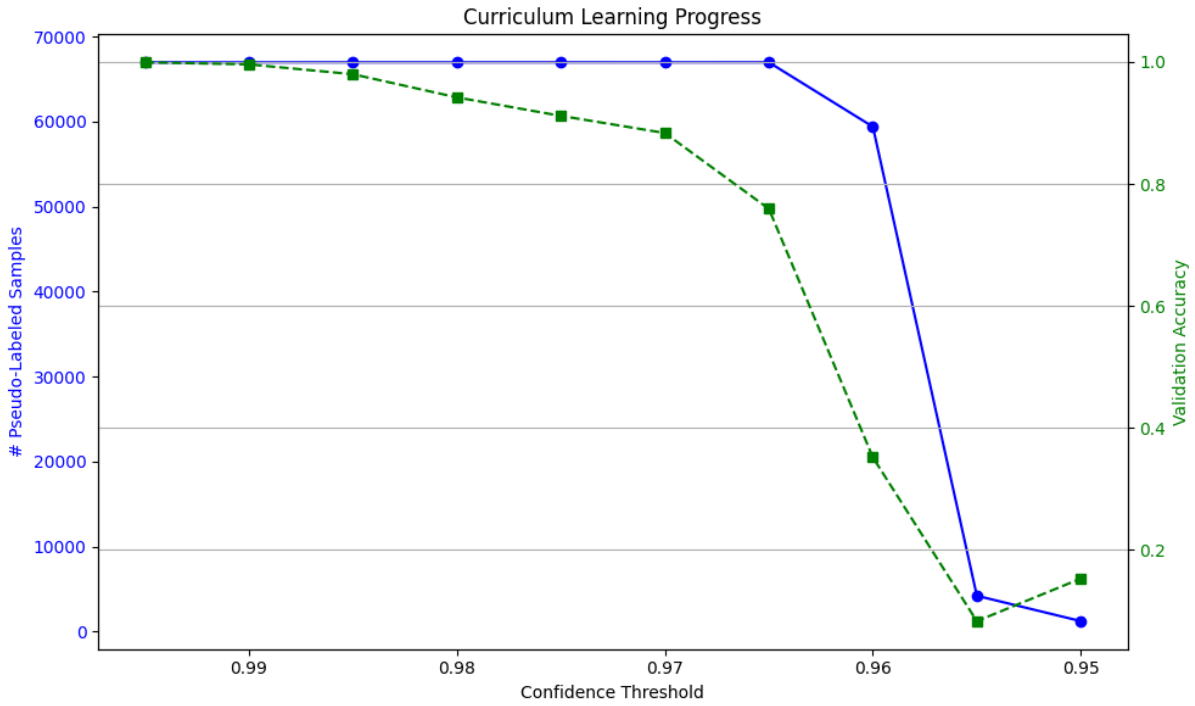

Fig 16: curriculum learning process tracked

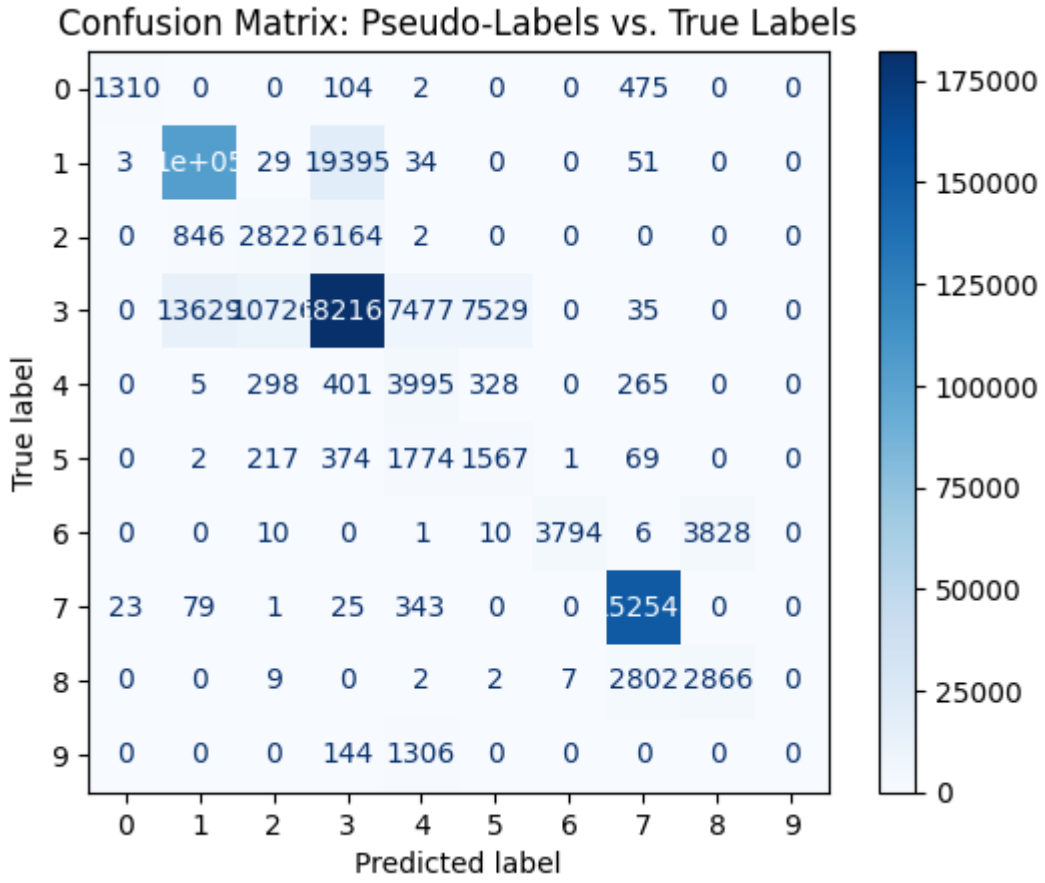

Fig 17: confusion matrix at the end of the process

Accuracy improved to **0.80** with **F1 score at 0.60**, though the minority classes were still underrepresented.

3.2.6 Reducing Number of Classes

To simplify the prediction task, I reduced the number of target classes to the six most represented, removing the least frequent ones. While this reduced granularity between attack types, it significantly improved model performance and confidence in pseudo-labels.

Results:

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

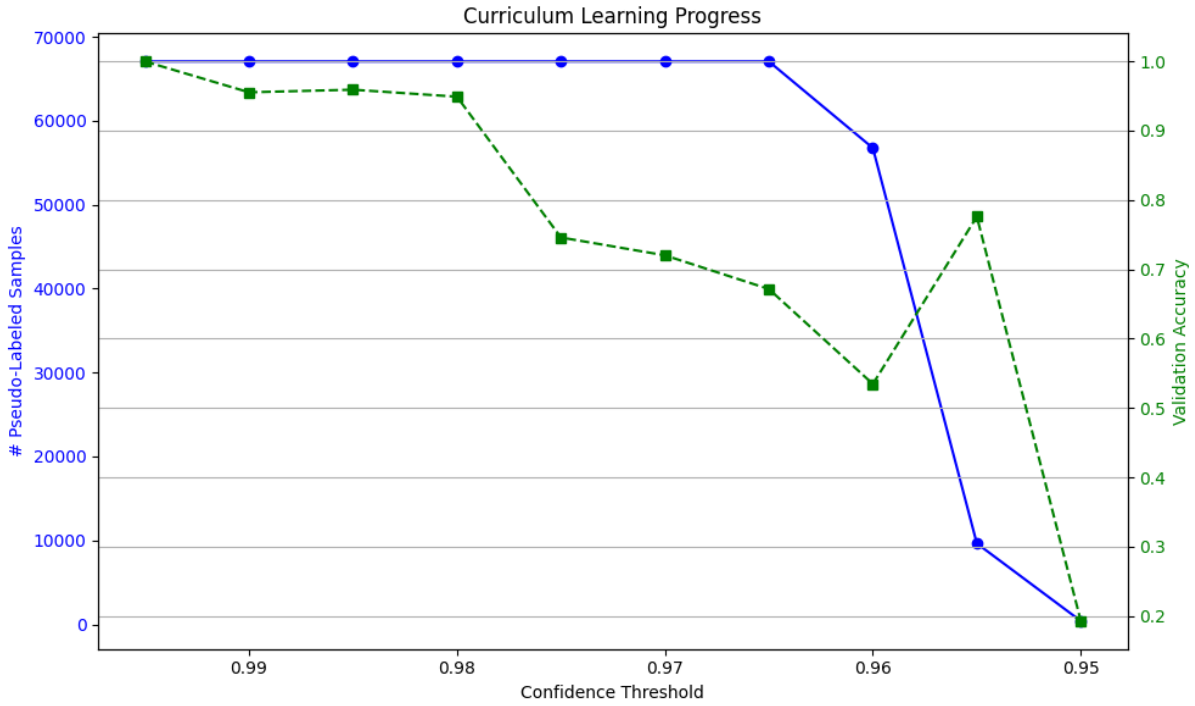

Fig 18: curriculum learning process number of labels selected, accuracy after the selection and confidence threshold as the steps advance

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

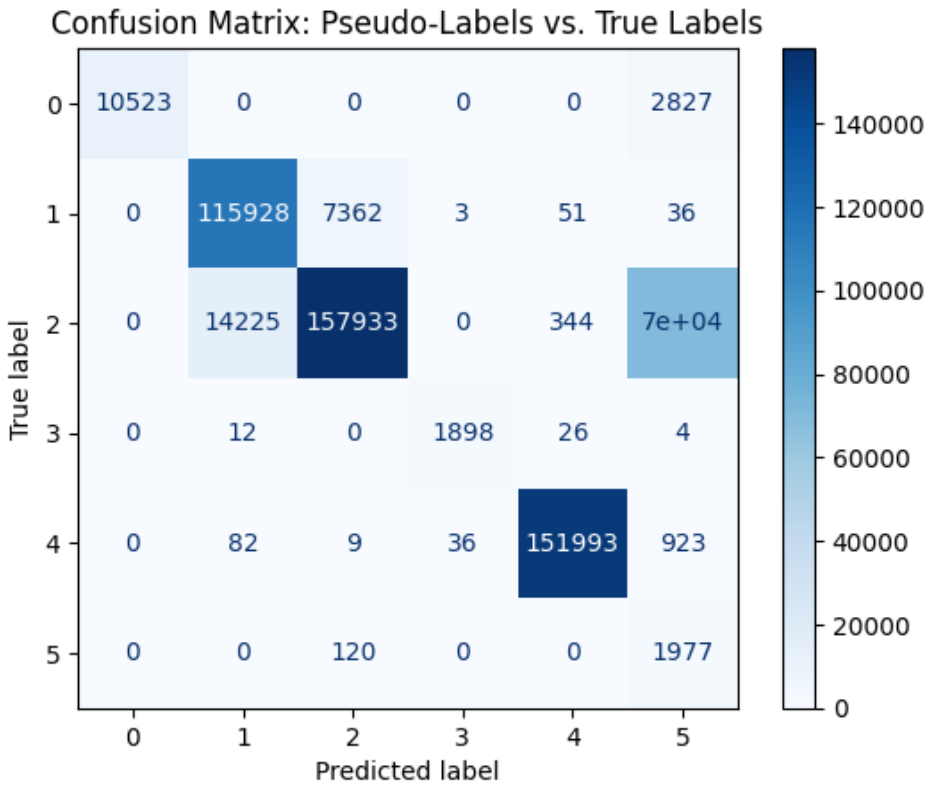

Fig 19: confusion matrix labelling on six classes

3.2.7 KL Divergence for Pseudo-Labelled Samples

To train the model with both labelled and unlabelled data, the standard categorical cross-entropy loss is applied to enforce alignment with ground-truth class labels. For unlabelled samples, a KL divergence term is introduced, which encourages the model's predictions to remain close to the pseudo-label distributions. The overall training objective is therefore defined as:

$$\mathcal{L}_{\mathrm{C}} = -\sum_{c=1}^{C} y_c \log Q_{\theta}(c \mid x)$$

$\mathcal{L}_{\mathrm{C}}$ is the *categorical cross-entropy* loss computed on labelled samples, where y_c is the one-hot encoded ground-truth label for class c, $\mathrm{Q}\theta(\mathrm{c} \mid \mathrm{x})$ is the model's predicted probability for class c, and C is the total number of classes.

$$\mathcal{L}_{\mathrm{K}} = \sum_{c=1}^{C} P_{\mathrm{pseudo}}(c \mid x) \log \frac{P_{\mathrm{pseudo}}(c \mid x)}{Q_{\theta}(c \mid x)}$$

$\mathcal{L}_{\mathrm{K}}$ is the KL divergence*, Kullback–Leibler divergence,* applied to unlabelled samples, where $\mathrm{Ppseudo}(\mathrm{c} \mid \mathrm{x})$ is the pseudo-label distribution (generated during the curriculum process) and $\mathrm{Q}_{\theta}(\mathrm{c} \mid \mathrm{x})$ is the model's predicted probability distribution. This term penalizes divergence between the model and the pseudo-labels.

$$\mathcal{L} = \mathcal{L}_{\mathrm{C}} + \lambda \mathcal{L}_{\mathrm{K}}$$

The final loss $\mathcal{L}$ combines supervised and semi-supervised objectives. The first term $\mathcal{L}_{\mathrm{C}}$ ensures learning from labelled data, while the second term $\mathcal{L}_{\mathrm{K}}$ leverages unlabelled data. The hyperparameter λ controls the relative importance of the semi-supervised signal, which varied between 0.2 to 0.5 during the project.

Using KL divergence, in the method explained above, allowed the model to capture uncertainty in pseudo-labels more effectively, reducing the risk of overfitting to potentially incorrect predictions. This soft alignment stabilized learning in later stages of the curriculum.

Despite this improvement, overfitting was observed in the early steps. To mitigate it, the model's dimensionality was reduced, which helped stabilize early-stage training (Fig 20, Fig 21). Performance only declined when the number of selected pseudo-labelled samples dropped below 6,000, indicating that most high-confidence samples were correctly labelled at that stage.

This highlighted a key insight: adding too many pseudo-labelled samples too early can clutter the learning process and introduce compounding errors.

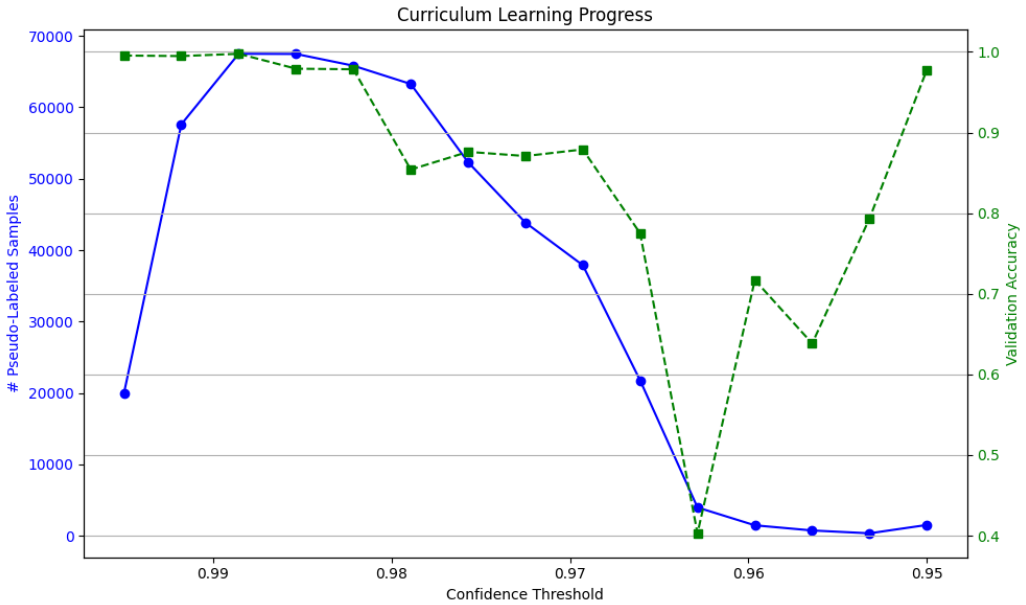

Fig 20: curriculum learning process number of labels selected, accuracy after the selection and confidence threshold as the steps advance

	precision	recall	f1-score	support
0	0.34	1.00	0.51	13568
1	1.00	0.97	0.98	125151
2	0.98	0.85	0.91	207126
3	0.36	0.64	0.46	1963
4	1.00	1.00	1.00	155741
5	0.26	0.84	0.39	2118
accuracy			0.93	505667
macro avg	0.66	0.88	0.71	505667
weighted avg	0.97	0.93	0.94	505667

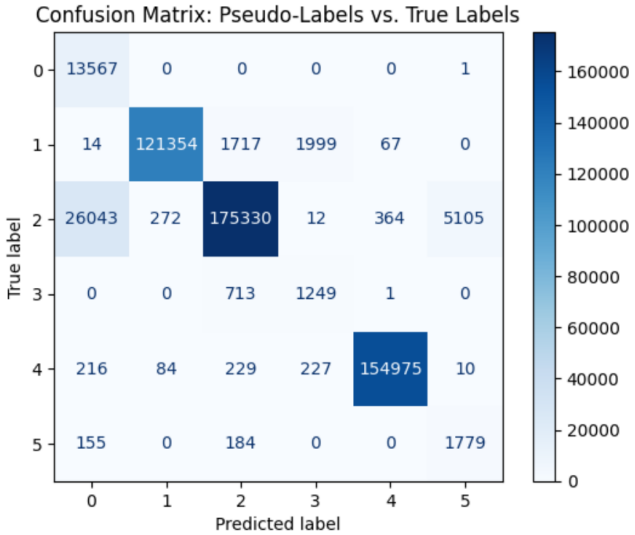

Fig 21: results of the process with added modifications

3.2.8 Upgrades

To address early-stage cluttering, the pseudo-labelling selection strategy was revised with curriculum principles in mind. Initially, too many samples were selected too early, which led to error accumulation once some inaccurate pseudo-labels were introduced.

Original selection strategy:

- Limit selection to 1/8 of the unlabelled dataset per step

Revised strategy:

- Gradually increase the selection limit over the first six steps until reaching 1/8 of the dataset (Fig 22)
- Enforcing class-based sampling

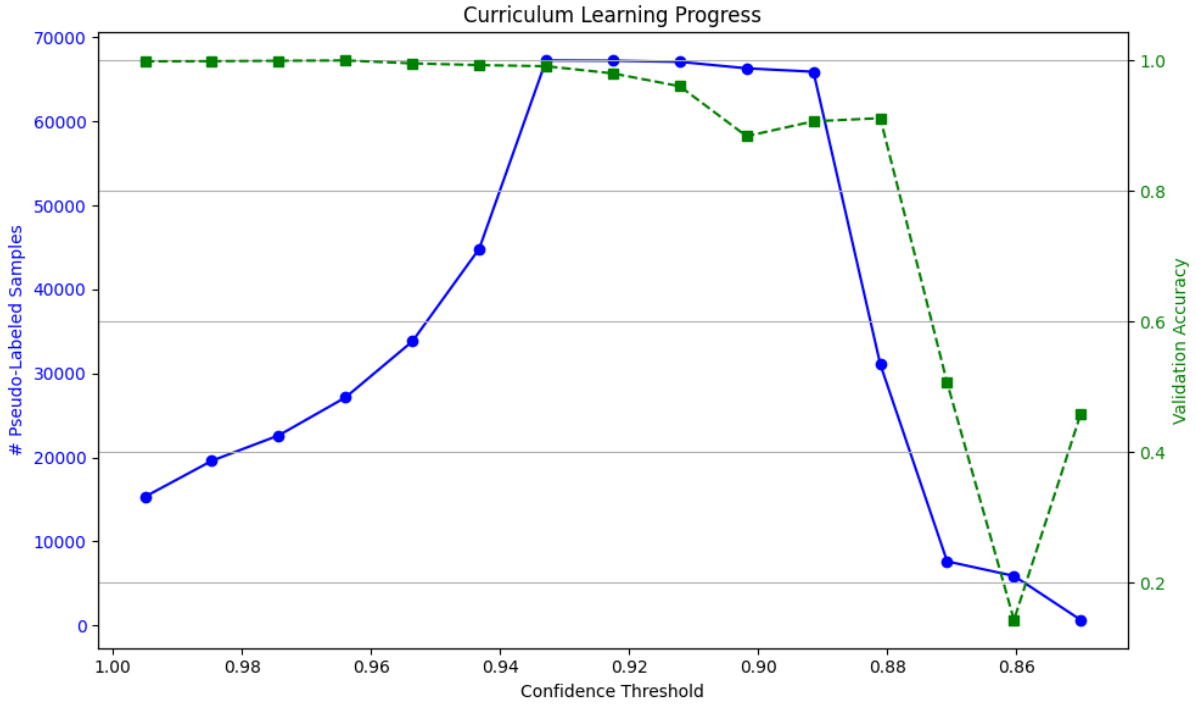

Fig 22: curriculum learning process number of labels selected, accuracy after the selection and confidence threshold as the steps advance

This gradual increase kept early-step accuracy high. Accuracy drops only occurred when the number of selected samples fell below 10,000, corresponding to the most challenging samples. Overall performance remained comparable, but robustness improved. Testing across different initial labelled subsets showed slightly higher average F1 scores, and the first nine steps no longer exhibited the sharp accuracy drop seen after steps five to six. Accuracy stayed in the 90s rather than falling into the 80s.

	precision	recall	f1-score	support
0	0.76	0.95	0.84	13589
1	0.99	0.85	0.92	125682
2	0.93	0.97	0.95	241334
3	0.33	0.99	0.50	1967
4	1.00	0.97	0.98	155909
5	0.28	0.84	0.42	2131
accuracy			0.94	540612
macro avg	0.71	0.93	0.77	540612
weighted avg	0.95	0.94	0.94	540612

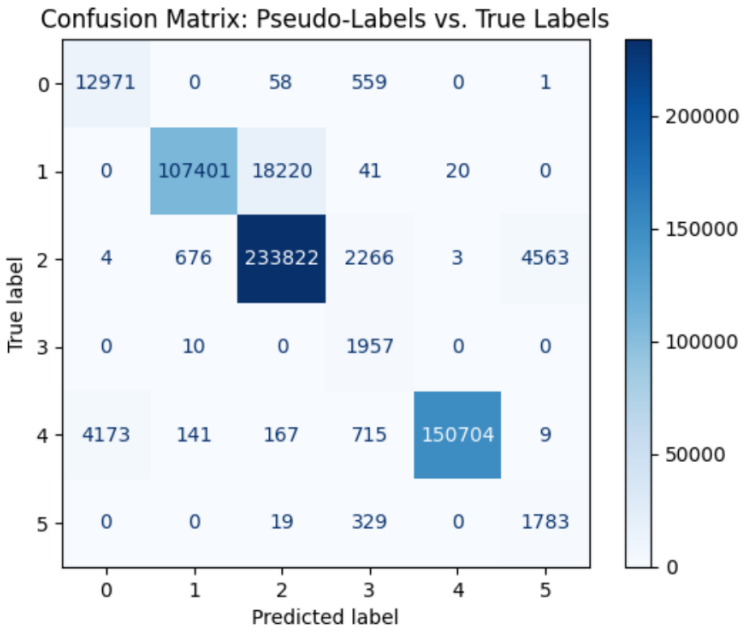

Fig 23: results of the process with newly added modifications

For further robustness, a stopping criterion can be introduced to halt curriculum learning when the number of selected pseudo-labelled samples falls below a threshold. This ensures that only high-confidence samples are retained while difficult-to-predict samples are ignored, preventing early-stage instability.

3.2.9 Parent confirmation

To improve the robustness of the semi-supervised learning process, an additional layer of verification is introduced for pseudo-labelled samples in the implementation of a parent model, which serves as an independent evaluator of the main model's predictions.

The parent model is distinct from the main model, differing in architecture and training configuration, ensuring that its assessments are not biased by the student's learning dynamics. During the curriculum, the parent model reviews unlabelled data and provides confirmation for pseudo-labels before they are incorporated into training. Only samples for which both the student and parent models agree are used with high confidence, reducing the risk of error propagation and supporting more reliable model updates.

By acting as an external check, the parent model strengthens the overall learning pipeline, adding a layer of reliability to the pseudo-labelling process and enhancing the stability of subsequent training iterations.

3.3 Omnipot Pseudo-Labelling Implementation

With the CIC-IDS 2017 pipeline working, I moved on to the main problem: labelling and identifying the type of attacks in the Omnipot dataset. Initially, there was a possibility of obtaining a small set of labelled Omnipot data, which I could use to imitate the above process, but the labelling of the Omnipot was not possible which meant I had to rely on the CIC-IDS 2017 dataset and clustering-based intuition and payload analysis to evaluate potential labels.

Although it was not directly mappable the curriculum learning approach developed in Section 3.2 remained applicable for generating high-quality pseudo-labels. For evaluation purposes (discussed later), I kept a subset separate to not lead to contaminated evaluation values.

Using this trained model, I applied the pseudo-labelling process to the Omnipot data. I then analysed the resulting labels in the context of the entire Omnipot dataset and compared their distribution to that of the CIC dataset. From the CIC-trained model's predictions, I selected only the high-confidence samples, as described in earlier sections. The results of this process are shown below.

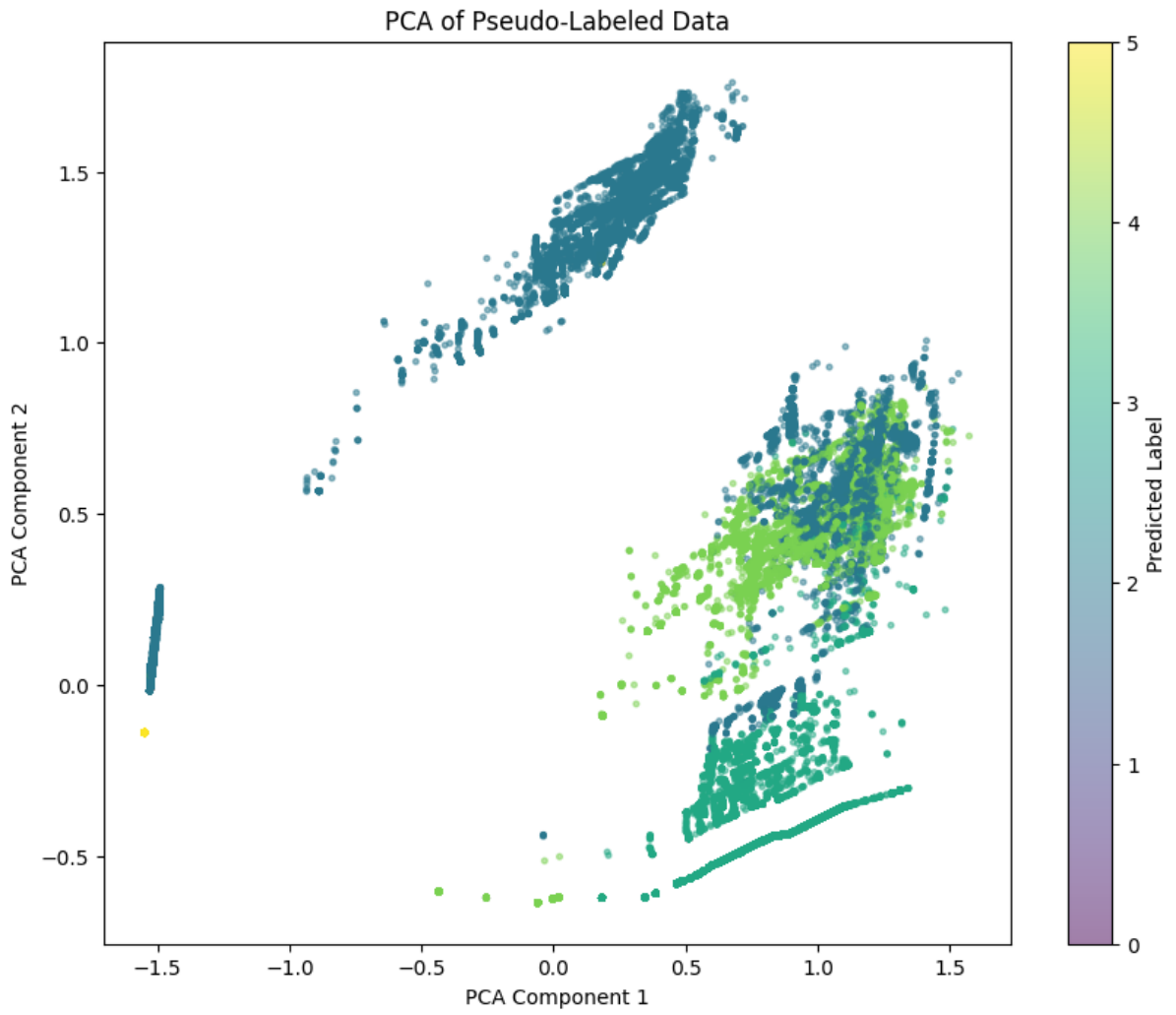

Fig 24: Pseudo-labelled omnipot data

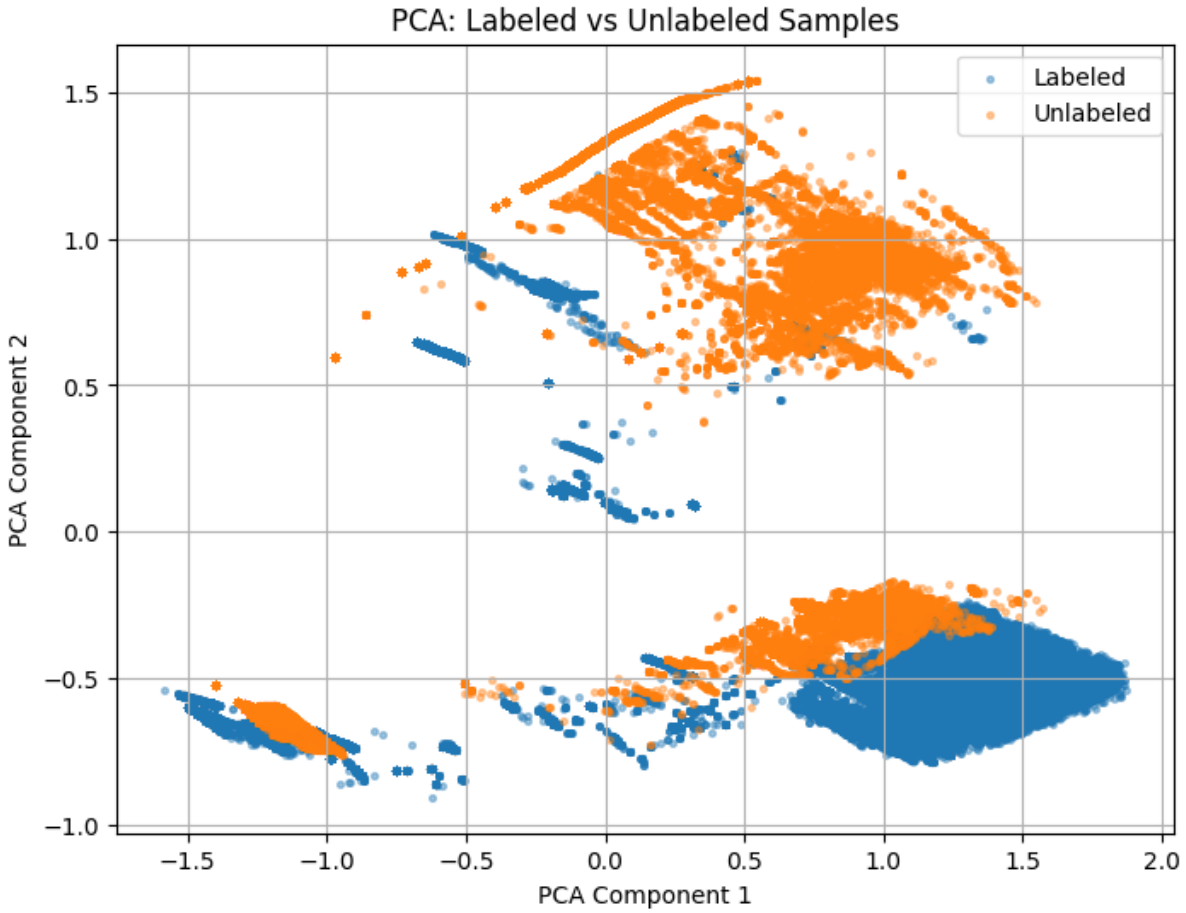

Fig 25: CIC vs Omnipot data

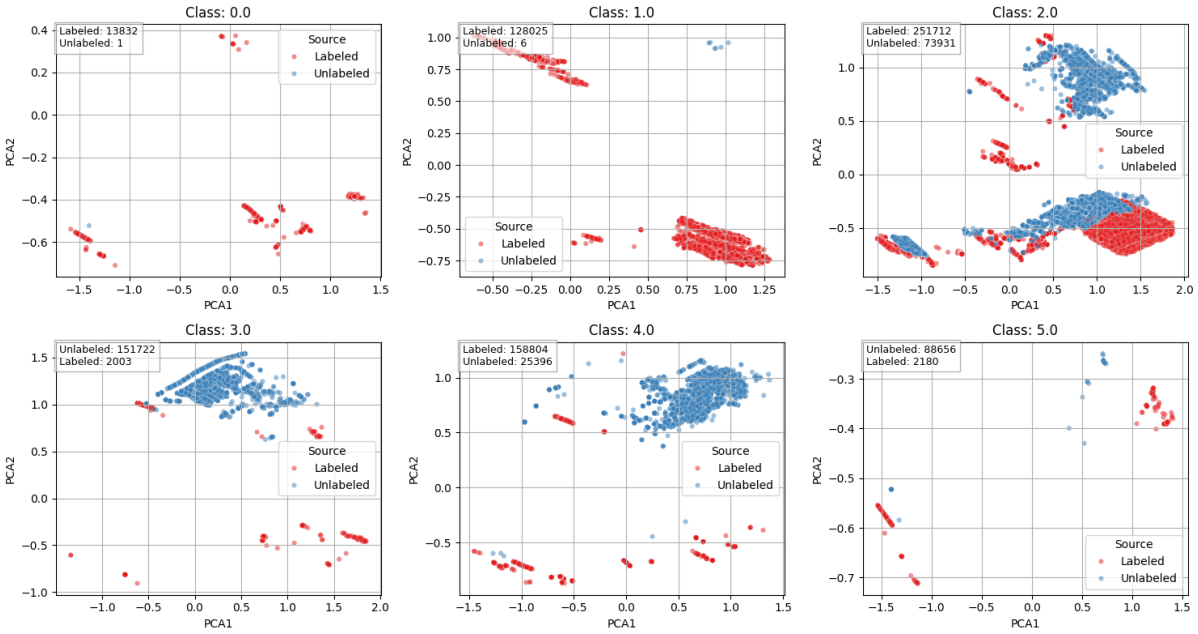

Fig 26: per class labelled vs unlabelled distribution

As discussed earlier, the CIC and Omnipot datasets differ in distribution. However, they also complement each other in certain regions of feature space. In Figure 25, the CIC samples in the bottom right are complemented by the Omnipot samples, which show slight overlap but also extend the distribution toward the left.

When observing the clusters formed from the pseudo-labelled predictions, I found both strong similarities and meaningful differences between the datasets. This outcome appears far more promising than any unsupervised labelling attempts using HDBSCAN or k-means.

To check the stability of these results and ensure they were not a one-off coincidence, I repeated the process multiple times with different hyperparameters. Across these runs, the resulting labels were highly consistent, which supports the conclusion that the pseudo-labels have relatively high confidence.

3.3.1 Evaluation method

The visual evaluation provided a good starting point, but I also explored additional methods to assess pseudo-label quality:

1. **Reverse evaluation:** A model trained on the labelled data was used to generate pseudo-labels for the unlabelled set. These pseudo-labels were then used to train a new model, which was evaluated on the labelled data. This helped measure how well the pseudo-labels preserved the true class structure.
2. **Self-consistency check:** A model trained solely on the pseudo-labelled data was applied to the unlabelled set again, to see how closely its predictions aligned with the original pseudo-labels and whether the outputs were coherent.

Results will be discussed in the results chapter.

Chapter 4: Payload analysis

Payloads are the most direct evidence of what an attacker is attempting to do. While network metadata (e.g., ports, timing, flow features) reveals patterns of communication, the payload provides the actual content exchanged, which can expose reconnaissance attempts, scanning behaviour, or exploitation of vulnerabilities. Analysing payloads is therefore a crucial step in understanding the nature of attacks in the dataset.

4.1 Dataset Overview and Limitations

The raw dataset contained roughly 470,000 non-empty payloads. However, many of these could not be meaningfully decoded or interpreted in human-readable form, since they consisted of protocol-level commands (e.g., SMB negotiation strings) rather than application-layer content. An example is shown below:

T SMBr (/K ^ 1 LANMAN1.0 LM1.2X002 NT LANMAN 1.0 NT LM 0.12

Fig 27: unreadable payload

Such payloads are valid network exchanges but do not expose attack intent in a straightforward textual format. After filtering for readability and removing duplicates, only about 170,000 payloads were suitable for higher-level analysis.

4.2 Probes, Scans, and Grouping

Grouping traffic by temporal bursts and communication pairs revealed that what appeared as individual probes were in fact coordinated scans. For example, repeated pings or GET / requests arriving at short intervals represent systematic reconnaissance of open services. Classifying them as scans rather than attacks in isolation provides a more realistic understanding of adversarial behaviour. This also explains why the dataset contained disproportionately high volumes of such traffic compared to actual exploit attempts.

4.3 Payload Inspection and Open-Source Models

To further investigate the payloads, I tested several open-source cybersecurity models that leverage payload and contextual information to classify attacks. Cisco's **Foundation-Sec-8B** model was of particular interest.

Despite the dataset limitations, it was possible to analyse the outputs for a subset of payloads. For example, one of the more complex and interpretable payloads was:

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

The model returned different responses depending on the run:

Response 1:

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.

Response 2:

Attack Class: 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.

This inconsistency illustrates a broader challenge: single payloads often lack sufficient context for reliable classification.

4.4 Reflections on Payload-Only Analysis

From these experiments, several conclusions emerged:

- The dataset was dominated by simple probes, often in the form of trivial HTTP requests.
- Payload-only analysis is limited. Single packets cannot reliably distinguish between benign probes, reconnaissance, and true exploitation attempts.
- Contextual aggregation, combining multiple packets into flows or sessions, and adding metadata (protocol, frequency, inter-arrival patterns), is essential for robust classification.
- Open-source AI models like Foundation-Sec-8B show potential but also highlight the ambiguity of short, minimal payloads.

In summary, payload analysis confirms that the dataset reflects attacker reconnaissance at scale, with fewer examples of exploitation. Probes such as GET / HTTP/1.1 dominate the traffic, underscoring the importance of grouping and context. Payloads alone, while informative, cannot fully characterise attack types without being combined with protocol-level and temporal information.

Chapter 5: Results

5.1 Structure of the Results Section

This chapter presents the outcomes of the methods described in the previous sections, focusing on both quantitative performance metrics and qualitative observations.

1. **Quantitative evaluation**: summarising the performance of unsupervised clustering, curriculum-based pseudo-labelling, and cross-dataset evaluation.
2. **Qualitative analysis**: including stability of pseudo-labels, visual inspection of clusters, and payload-based plausibility checks.

The evaluation reflects the central challenge of this project: the absence of true labels in the Omnipot dataset. As a result, some metrics are from labelled datasets (CIC IDS 2017) and others are indirect or qualitative.

5.2 Results Table

Experiments	Dataset	Metrics	Value	Notes
Ungrouped cluster	Omnipot	Silhouette	0.3	Poor separation; high compute
Grouped cluster	Omnipot	Silhouette	0.77	Grouping improved scores but clusters ≠ attack types
Curriculum pseudo-labelling	CIC IDS (train/test)	Accuracy / F1	0.80 / 0.60	With class balancing and KL loss
Curriculum pseudo-labelling (6 classes)	CIC IDS 2017	Accuracy / F1	0.82 / 0.77	Reduced classes → higher accuracy
train on Omnipot df → eval on CIC	Omnipot → CICsubset	Accuracy / F1	0.89 / 0.57	Measure cross-dataset alignment
Trained on cic df → eval on Ominpot label	CICsubset → Omnipot	Accuracy / F1	0.22 / 0.12	Measure cross-dataset alignment
Self-consistency evaluation	Omnipot → omnipot	Accuracy / F1	0.99 / 0.80	Measure cross-dataset alignment

5.3 Step-by-Step Evaluation

Step 1: Unsupervised Baselines

To establish an initial reference, I compared the semi-supervised pipeline against unsupervised clustering using the same feature space.

- K-Means produced a best silhouette score of 0.30, indicating weak separation between clusters. Visual inspection confirmed that clusters overlapped heavily, and cluster membership did not correspond to attack categories.
- HDBSCAN improved separation in some parameter settings, yielding silhouette scores up to 0.77, but the resulting groups were not semantically meaningful (e.g., clusters grouped flows by traffic intensity rather than attack type).

This highlighted a key limitation: although clustering found structure, it did not align with the labels of interest (attack categories). In other words, good geometry (high silhouette) ≠ good semantics (useful attack classes).

Step 2: Baseline on CIC IDS 2017

As a supervised reference, I trained the curriculum learning model on the labelled CIC IDS 2017 dataset.

- On the full class set, the model achieved Accuracy = 0.80 and Macro F1 = 0.60. Accuracy was high, but F1 showed imbalance, performance was dominated by majority classes.
- Reducing the task to 6 merged classes increased balance, giving Accuracy = 0.82 and F1 = 0.77.

This confirms two tendencies:

1. Curriculum + balancing mitigates skew.
2. Reducing the number of classes improves per-class performance by reducing confusion and data sparsity.

These baselines serve as an upper bound for what the semi-supervised method could achieve.

Step 3: Pseudo-labelling Omnipot visual evaluation

A visual analysis of the pseudo-label distributions revealed partial separation among classes, but they did not clearly match the expected attack types. This reflects the domain shift between Omnipot and CIC IDS 2017: although the model's features transferred effectively, the actual attack patterns and their behaviour differed between the two datasets.

Nonetheless, variability across different runs indicated that noisy or ambiguous samples continued to propagate, particularly in later iterations.

Step 4: Cross-dataset Evaluation

To evaluate whether pseudo-labels carried meaningful structure, I trained a model solely on pseudo-labelled Omnipot data and evaluated it on a labelled CIC IDS subset:

- The pseudo-trained model achieved Accuracy = 0.89, but F1 = 0.57. High accuracy but low F1, struggling to correctly predict the minority classes.
- In the reverse setup (trained on CIC subset → evaluated on Omnipot pseudo-labels), performance collapsed to Accuracy = 0.22 / F1 = 0.12. This revealed that pseudo-labels deviated significantly from the original label structure, either due to class mismatch or accumulated noise.

Step 5: Self-consistency of pseudo-labels

Finally, I evaluated the model trained and tested purely on the pseudo-labelled Omnipot dataset (i.e., internal self-consistency).

- The model achieved near-perfect performance with Accuracy = 0.99 and F1 = 0.99.
- However, the reliability of the macro F1 was affected by severe class imbalance: two classes had only a small number of samples (max double digits). When these rare samples appeared in the test set, the F1 dropped sharply, pulling down the macro average despite otherwise near-perfect performance.

Interpretation

- The Unsupervised methods (Step 1) gave unsure results even with a high silhouette score (0.77) because the models found other patterns and not meaningful attack clusters.
- The supervised baseline (Step 2) showed that the complete process and elements added to the curriculum learning where important to the robustness of the models and confident predictions.
- Pseudo-labels (Step 3) showed partial class separation but inconsistent alignment with true attack patterns, reflecting domain differences and persistent noise in later iterations.
- Cross-dataset tests (Step 4) The low performance in the second step of the evaluation is not encouraging but the fact that the model trained on Omnipot performed reasonably well when tested on CIC data means that the Omni dataset captures general attack patterns that CIC also has. The lower accuracy on step 2 suggests that the CIC dataset does not represent the diversity in Omni. Meaning that Omnipot has a broader range of variability given it is a real sample of attacks collected from a honey pot contrarily to the CIC dataset that is artificially generated.
- Self-consistency evaluation yielded near-perfect accuracy and F1, confirming that pseudo-labelling generated a coherent internal structure. However, macro F1 was unstable due to extreme class imbalance: minority classes contained only a handful of samples, so their occasional misclassification disproportionately reduced the score.

Overall, pseudo-labelling proved sensitive to label quality but still added meaningful value by supplying more representative training examples. The method improved generalization on CIC-like data, though it remained vulnerable to class skew and domain mismatch.

Conclusion

This project addressed the challenge of analysing and classifying a large, unlabelled cyberattack dataset (Omnipot) by integrating advanced feature engineering with semi-supervised learning. Initial experiments with unsupervised clustering failed to recover meaningful attack categories, motivating the adoption of a curriculum-based pseudo-labelling strategy.

This strategy, applied first to CIC IDS 2017, achieved perceived performance improvements compared to naive pseudo-labelling. Transferring the trained model to OmniPort yielded stable and reproducible pseudo-labels that retained predictive power, indicating meaningful encoded patterns.

Payload-focused classification was limited by incomplete payloads, reinforcing the value of combining payload analysis with contextual metadata. The results demonstrate that curriculum-driven pseudo-labelling is a viable method for generating high-confidence labels in large, unlabelled cyberattack datasets, offering a foundation for automated threat identification.

Further development

If more time and resources where available, I would expand the analysis and development of the project by making a more defined model with more classes and making it explainable for deployment. The main task would be having true labels of the dataset to rely on; it would probably be a process partly made by hand as after grouping the bursts (2.2) there would be an analysis of payloads to be done and then assigning a class to that burst. When a reasonable set of data labelled with human decision across multiple classes with peer validation, etc. then the curriculum process could be done based on those labels and have a data set that is much more accurate. Additionally, the dataset would not need to be confined to the CIC IDS features but to the broader available information.

This would also allow more features than the ones selected as the dataset would not be restricted to the features in the CIC IDS dataset that where doable from the Omnipot. An easy example of this would be the payload embeddings. With this base a domain specific model could be fine-tuned on bash script and cyber attack scripts instead of using CodeBERT.

This could be done in two main ways, the man analysed method as stated before or if available some bigger computer processing method thought the LLM path of foundation sec or other LLM that are able to determine the source of attack based on a series of payloads.

Citations

Silhouette (clustering). (2025). Retrieved from Wikepedia: https://en.wikipedia.org/wiki/Silhouette_(clustering)

What Are Honeypots (Computing)? (2025). Retrieved from Fortinet: https://www.fortinet.com/resources/cyberglossary/what-is-honeypot#:~:text=A%20honeypot%20is%20a%20decoy,and%20gather%20valuable%20threat%20intelligence.

Dash, S. P., Khandeparkar, K. V., & Agrawal, N. (2025, March 1). *CRUPL: A semi-supervised cyber attack detection with consistency regularization and uncertainty-aware pseudo-labeling in smart grid*. arXiv.org. https://arxiv.org/abs/2503.00358

Delplace, A., Hermoso, S., & Anandita, K. (2019, May 17). *Cyber Attack Detection thanks to Machine Learning Algorithms*. https://arxiv.org. https://arxiv.org/pdf/2001.06309v1

Berthelot, D., Carlini, N., Goodfellow, I., Oliver, A., Papernot, N., & Raffel, C. (2019, October 23). *Mixmatch: A holistic approach to semi-supervised learning*. arxiv.org. https://arxiv.org/pdf/1905.02249

Yang, X., Song, Z., & King, I. (2021, August 23). *A survey on deep semi-supervised learning*. arxiv.org. https://arxiv.org/pdf/2103.00550

Alturki, B., & Alsulami, A. A. (2025, June 19). *Semi-supervised learning with entropy filtering for intrusion detection in asymmetrical IOT Systems*. MDPI. https://www.mdpi.com/2073-8994/17/6/973

Smithie, J. (2023, October 7). *Intrusion detection in cybersecurity*. Advances in Engineering Innovation. https://www.ewadirect.com/journal/aei/article/view/4474

Karbasi, A., & Team at Foundation AI — Cisco. (2025, April 28). *FDTN-ai/foundation-SEC-8B · hugging face*. fdtn-ai/Foundation-Sec-8B · Hugging Face. https://huggingface.co/fdtn-ai/Foundation-Sec-8B

Davies, J. (2024, September 23). *KMEANS clustering: Calculating the optimal number of clusters*. Medium. https://medium.com/@AI_MLwithJonny/kmeans-clustering-calculating-the-optimal-number-of-clusters-e3a708ce7349

Datasets:
Search UNB. University of New Brunswick est.1785. (2017, July). https://www.unb.ca/cic/datasets/ids-2017.html