# AI unsupervised approach

## 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. Results were not satisfactory and given the high number of features it took too long to justify

## 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 tehat by testing with and without, seeing that without the clusters formed better.

I then removed most features except cic features that I decided upon.

This lead to high silhouette of around 0.8 (fig 1), I did port analysis, number of attacks per group, and visually the clusters where good.

A graph of a plot

AI-generated content may be incorrect.

Fig 1 silhouette score of hdbscans

# AI semi supervised approach

### Model trained on the cic and predicting on omni port

## 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:

python

CopyEdit

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 simulate the OmniPort dataset more realistically, I reduced the number of classes to 6. This improved both confidence calibration and class balance 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.A screenshot of a computer screen

AI-generated content may be incorrect.

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.

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.

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 handled either in the same manner but with more refined models or simply ignored, or with knn or other methods that work well when a lot of confident labelled data is in place and few unlabelled data.