BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

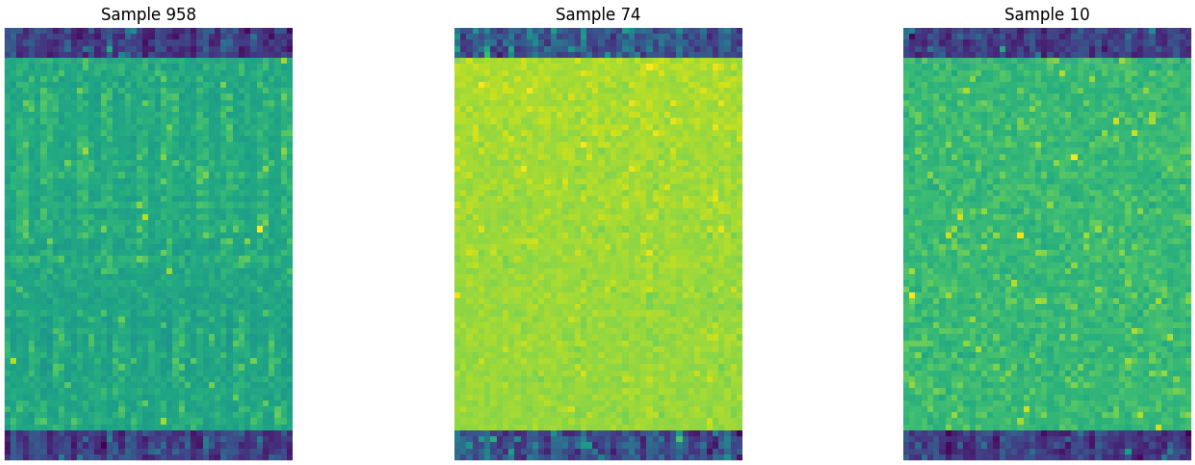
isep
Paris Institute of Digital Technology

*Electronics and Communication Technologies*

*2024/2025*

# Machine Learning Fundamentals

## Classification 5G base stations




**Guillaume Cappe de Baillon - Mailley Charles**

## Summary



## Table of figures



## GitHub link

https://github.com/Deathvanos/MPA_MLF-Classification_5G_base_stations

# Introduction

Mobile communication is a fundamental part of modern everyday life, enabling simple communication with friends and family as well as access to a great deal of information. The connection relies primarily on cellular networks and base stations, such as eNodeBs (in 4G/LTE) or gNodeBs (in 5G), which manage communication between the user equipment and the core network. However, the open radio interface exposes these networks to security vulnerabilities. They can exploit these loopholes for their benefit by employing home-made hardware and software for the creation of False Base Stations (FBS) commonly referred to as Rogue Base Stations (RBS) or IMSI Catchers. They will then replicate actual base stations and deceive mobile terminals to connect to them and hence facilitate information interception, user tracking, or injection of false messages. For an FBS to function, it must be capable of mimicking the 'messages' of a real base station, i.e., necessary synchronization sequences.

# Project description

This project addresses the problem of finding such illegal transmissions in a 5G cellular environment. The primary objective is to design and evaluate a machine learning-based classification model for determining a signal radiated from a valid T-Mobile gNodeB compared to one radiated from a possible attacker's FBS located nearby.

The classification task is based on the examination of channel frequency response values derived from captured primary (PSS) and secondary (SSS) synchronization sequences – integral components of the 4G/LTE and 5G signal frame format. The data set consists of samples in the form of 2D arrays, with each sample having 72 rows (subcarriers) and 48 columns (iterations of channel frequency responses). Each sample must be classified into one of three classes:

- Class 0: Signal from a legitimate T-Mobile gNodeB in an adjacent building.
- Class 1: Signal from an FBS (attacker) at a first possible location.
- Class 2: Signal from an FBS (attacker) at a second possible location.

# I. Data exploration

We begin with the investigation of the training data set that was used to train the model. It contains 1,491 samples, each of which had been stored as a NumPy array. The images are monochrome (1 channel), though matplotlib may render them in artificial color maps for contrast.

Each image dimension is 72 × 48 pixels (height × width), so they are low-resolution images.

The class distribution is clearly imbalanced:

- Class 0: 1,209 samples
- Class 1: 141 samples
- Class 2: 141 samples

This imbalance would bias the model towards Class 0 predictions unless steps to counter them (e.g. oversample or class weights) are taken.

To informally get some intuition for the data, some sample images for each class were manually inspected. Although visually similar at first consideration, some common patterns were observed:

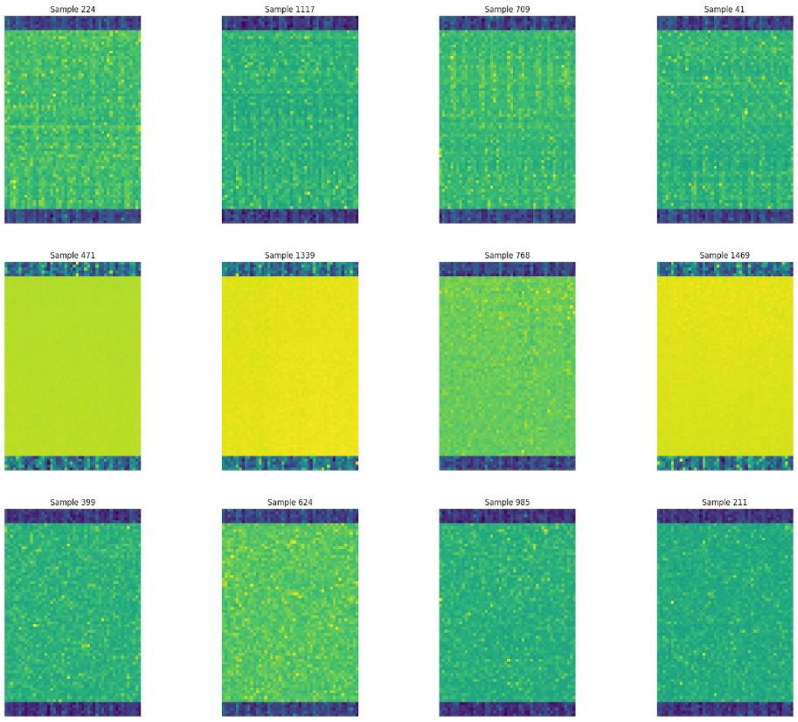



*Figure 1 - Class 0, 1 & 2 (from top to bottom)*

- Class 0: Most regular pattern is demonstrated in this class. Most images typically contain three regions of varying patterns:
    - The top and bottom regions consist of vertical line patterns,
    - The middle region consists of horizontal line patterns. These regular patterns are visually prominent and can be learnt by a CNN.
- Class 1: The images in this class tend to be brighter as a whole, with a steadier gray tone across the entire image. No local patterns are dominant, making this class more difficult to isolate based on texture alone.
- Class 2: This class is characterized by the lack of any perceivable structure. The intensities of pixels appear to be randomly distributed, and the image appears to be noise.

Based on these results, our aim is to develop a model that is able not only to identify overt visual patterns (such as Class 0) but also subtle statistical differences (between Classes 1 and 2). The model needs to be sensitive enough to exploit the repeated structures of Class 0 but learn to recognize weaker signals in the more ambiguous classes.

# II. Model 1: Simple CNN Architecture

We started with the simplest sequential CNN. Its architecture was as follows:

Model: "Chagui_Team_model"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_8 (Conv2D) | (None, 70, 46, 32) | 320 |
| max_pooling2d_8 (MaxPooling2D) | (None, 35, 23, 32) | 0 |
| conv2d_9 (Conv2D) | (None, 33, 21, 64) | 18,496 |
| max_pooling2d_9 (MaxPooling2D) | (None, 16, 10, 64) | 0 |
| flatten_4 (Flatten) | (None, 10240) | 0 |
| dense_8 (Dense) | (None, 128) | 1,310,848 |
| dropout_4 (Dropout) | (None, 128) | 0 |
| dense_9 (Dense) | (None, 3) | 387 |

Total params: 1,330,051 (5.07 MB)
Trainable params: 1,330,051 (5.07 MB)
Non-trainable params: 0 (0.00 B)

*Figure 2 – Our CNN model Architecture*

This model was compiled with Adam optimizer and categorical crossentropy loss. It was trained directly on the originally split training data (X_train, y_train) for 10 epochs, validated against the corresponding test split (X_test, y_test).

As a result, we observed in the evaluation that this simple model did not perform well. The performance measures and confusion matrix pointed towards the model continuing to classify the majority class (Class 0) to all test set instances consistently. The validation accuracy leveled off quite quickly at about 78%, which is a reflection of Class 0 test split ratio, suggesting the model failed to recognize distinguishing features of the minority classes (Class 1 and Class 2). This failure was a result of the severe class imbalance in the training data, whereby the model was minimizing loss by defaulting to the most common class.

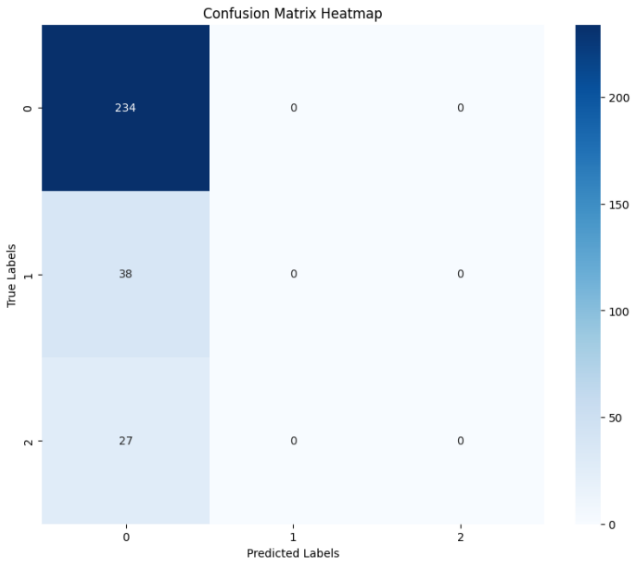



*Figure 3 - Confusion matrix of ours first model*

# III. Model 2: Complexified CNN Architecture

Due to the limitation encountered with the first CNN, we employed a variety of methods to allow the model to learn meaningful features within the minority classes.

One of the initial works was to reshape how data was divided into training and validation sets by adding the “stratify” parameter. Instead of a random split, we employed stratified sampling so that all of the classes were represented proportionally in both sets. This was required to preserve the inherent nature of the original class imbalance while still enabling the model to learn from each class as it trained.

We also added fixed training callbacks for the purpose of enhancing stability as well as to prevent overfitting. There was an “EarlyStopping” mechanism implemented to halt the training when validation loss stopped improving for a limited number of epochs. Additionally, a “ModelCheckpoint” was used that automatically saved the best version of the model from training based on validation accuracy.

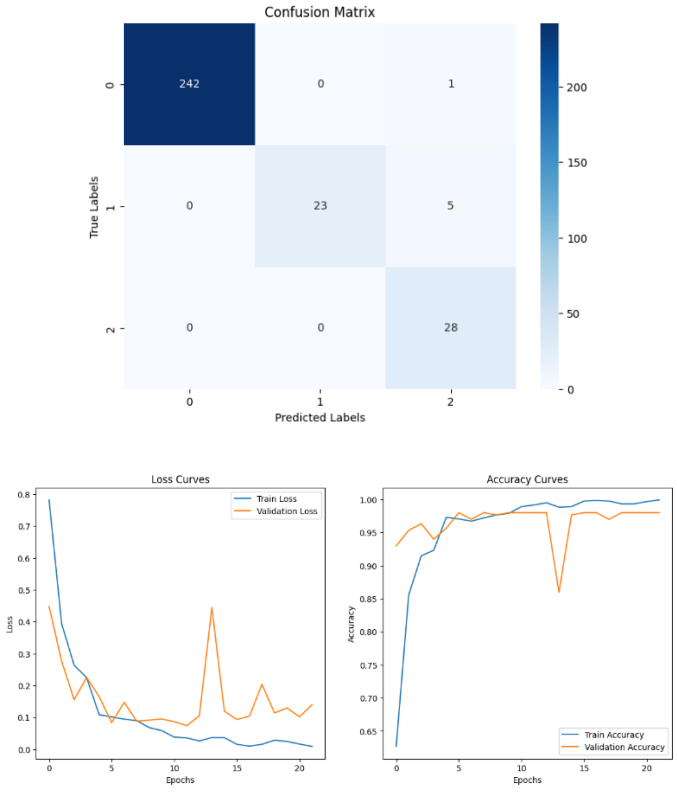



*Figure 4 - Confusion matrix of ours second model & Loss + Accuracy Curves*

On the Kaggle public leaderboard, this version of the model achieved a score of 0.95833, confirming that the optimizations were effective, though the model still faced difficulties in fully capturing the more subtle visual cues present in classes 1 and 2.

# IV. Model 3: Optimized CNN Architecture

To further address the remaining limitations and improve class sensitivity, we developed a more advanced training strategy based on balanced batch generation. This method constructs each training batch with a fair representation of all classes, achieved through class-wise oversampling. The goal was to ensure the model received equal learning opportunities from each class during every training iteration.

In parallel, we incorporated light data augmentation, including random brightness changes and horizontal flips. These small variations helped increase the model's robustness by exposing it to slightly altered versions of the same inputs, improving its ability to generalize.

This final version of the model demonstrated excellent performance during evaluation. The confusion matrix showed strong recognition across all three classes, and the model no longer defaulted to predicting only the majority class.

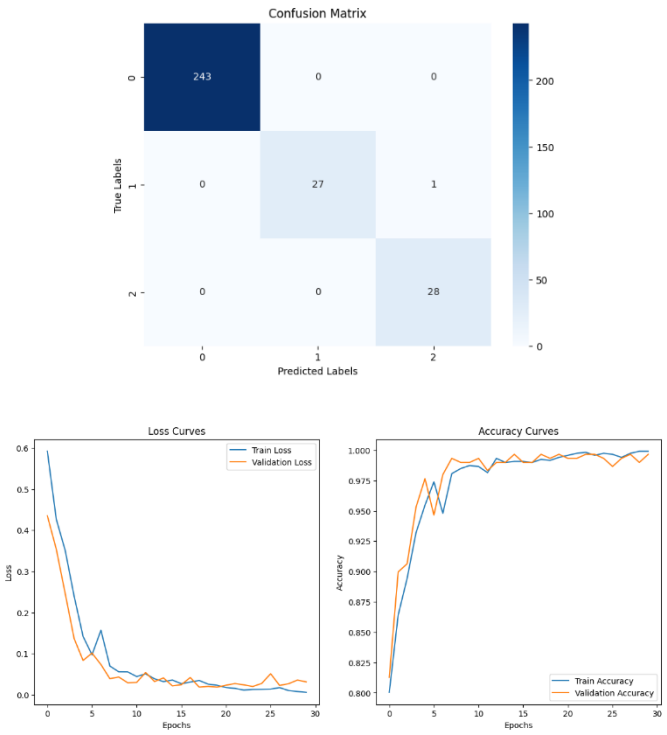



*Figure 5 - Confusion matrix of ours last model & Loss + Accuracy Curves*

Most notably, the model achieved a score of 0.99166 on the Kaggle public leaderboard, confirming that the combination of class balancing and data augmentation was highly effective. Although it did not reach 100%, such a result is a strong indicator of robustness. As perfect classification is rarely achievable in real-world data scenarios, this performance reflects a model that is nearly optimal.

# Conclusion

The objective of this project was to create a classification model to distinguish a legal base station signal from a FBS. We were allowed to choose any kind of ML model and use any techniques to tune it. The data exploration part enlightened us to choose a convolutional neural networks (CNN) architecture to classify images in grayscale with three visually distinguishable classes. We began with a minimal sequential architecture and gradually moved through stratified sampling of data, performance-based callbacks, and finally a custom data batch generator which was able to include class balancing during augmenting data.

For the first model, while we had a good architecture, it was unable to generalize due to severe class imbalance with the training data; this was solved by stratified sampling and checkpointing to maintain performance and stabilize learning for underrepresented images.

For the second model, we changed the data splitting to get a better balance in the training and testing data. We also improved the training (callbacks, EarlyStopping and ModelCheckpoint) to enhance precision and prevent overfitting. Through experimentation, we observed a distinct stabilization in our model's confidence reflected by an internal Kaggle Score of 0.95833.

For our final model, we trained on a specialized data generator that could create balanced batches of data with minimal augmentations. This allowed our network to learn the relative importance of logical representations in order to separate classes. This resulted in a final Kaggle score of 0.99166 - not perfect, but one of high scoring excellence which demonstrates extreme generalizability, as well as one which guaranteed our training pipeline guaranteed optimal model generalizability.

In conclusion, we have provided an explanation on how comparatively simple architecture of CNN is capable of achieving high performance by proper data preparation and meticulous model training, along with designated strategic optimizations. While theoretically 100% is never achievable in the real-world situation, our model is very close, offering precision and consistency in multi-class image classification.