Electrosense OpenAPI

Yago Lizarribar: yago.lizarribar@imdea.org

Alessio Scalingi: alessio.scalingi@imdea.org

Developing the Science of Networks

Outline

1. Basic Usage (Yago Lizarribar)
   a. General facts
   b. Main functions
   c. Examples
2. Use cases (Yago Lizarribar)
   a. FM channels detection
   b. Anomalies in Spectrum
3. Technology Classification (Alessio Scalingi)

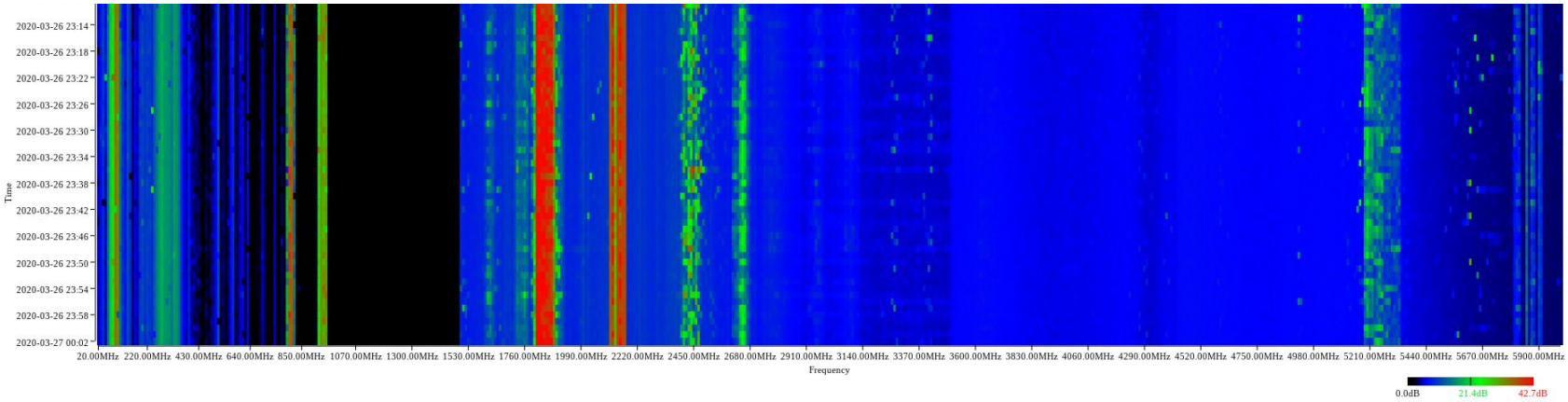
OpenAPI purpose

Electrosense API provides spectrum power measurements:

- Over time
- Over different countries/regions

Different **Spectrum Applications** can then use this data

OpenAPI purpose



uc3m

www.networks.imdea.org

Electrosense OpenAPI

Basic Usage

institute IMDEA networks

Some facts

- API Specification:
  - https://electrosense.org/api-spec
- Latest stable release: August 8, 2019
- Base URL:
  - https://electrosense.org/api…
- 4 different endpoints:
  - Sensor List
  - Sensor details
  - Aggregated data
  - Raw FFT measurements

Examples of usage

- Github repository with examples (python):
  - https://github.com/electrosense/api-examples
- These examples include:
  - Map of sensors (Europe)
  - Plotting the spectrum in different parts of Europe
  - Waterfall plots with different aggregations

Example query (with curl)

If we want to obtain the list of sensors:

```
curl --user username:password https://electrosense.org/api/sensor/list
```

Sensor List

- Query:
  - <base>/sensor/list
- Output is a JSON array:
  - Sensing Status (ON/OFF)
  - Model
  - Firmware version
  - Location


```
{
  "id": 55,
  "type": "esraspi2psdr",
  "serial": 202481589733661,
  "name": "Noelia",
  "uid": null,
  "operator": null,
  "address": null,
  "country": "Spain#",
  "contact": null,
  "position": {
    "longitude": -3.76501052880473,
    "latitude": 40.33400364111561 5,
    "altitude": 670.6424560546875,
    "indoor": true
  },
  "deployed": 1482307392,
  "model": null,
  "frontend": "rtl-sdr2",
  "sensing": true,
  "lastConnectionEvent": 1586490519,
  "anonymized": true,
  "antenna": null
},
{
  "id": 62,
  "type": "esraspi2psdr",
  "serial": 202481596393530,
  "name": "madrid_city_1",
  "uid": null,
  "operator": null,
  "address": null,
  "country": "Spain",
  "contact": null,
  "position": {
    "longitude": -3.712304188242629,
    "latitude": 40.40046818831868,
    "altitude": 579.0170288085938,
    "indoor": true
  },
  "deployed": 1484214412,
  "model": null,
  "frontend": "rtl-sdr2",
  "sensing": true,
  "lastConnectionEvent": 1586801469,
  "anonymized": true,
  "antenna": null
},
```

Map of Sensors

- Use of the Sensor List endpoint
- Map:
  - Green: Sensing sensors
  - Red: Offline sensors

Sensor details

- Query:
  - <base>/sensor/details/<serial>?<query_parameters>
- Parameters:
  - **serial** (required): Serial number of the sensor
  - **time** (optional): Time at which to retrieve data (Unix timestamp)
- Output is a single JSON (same data as before)

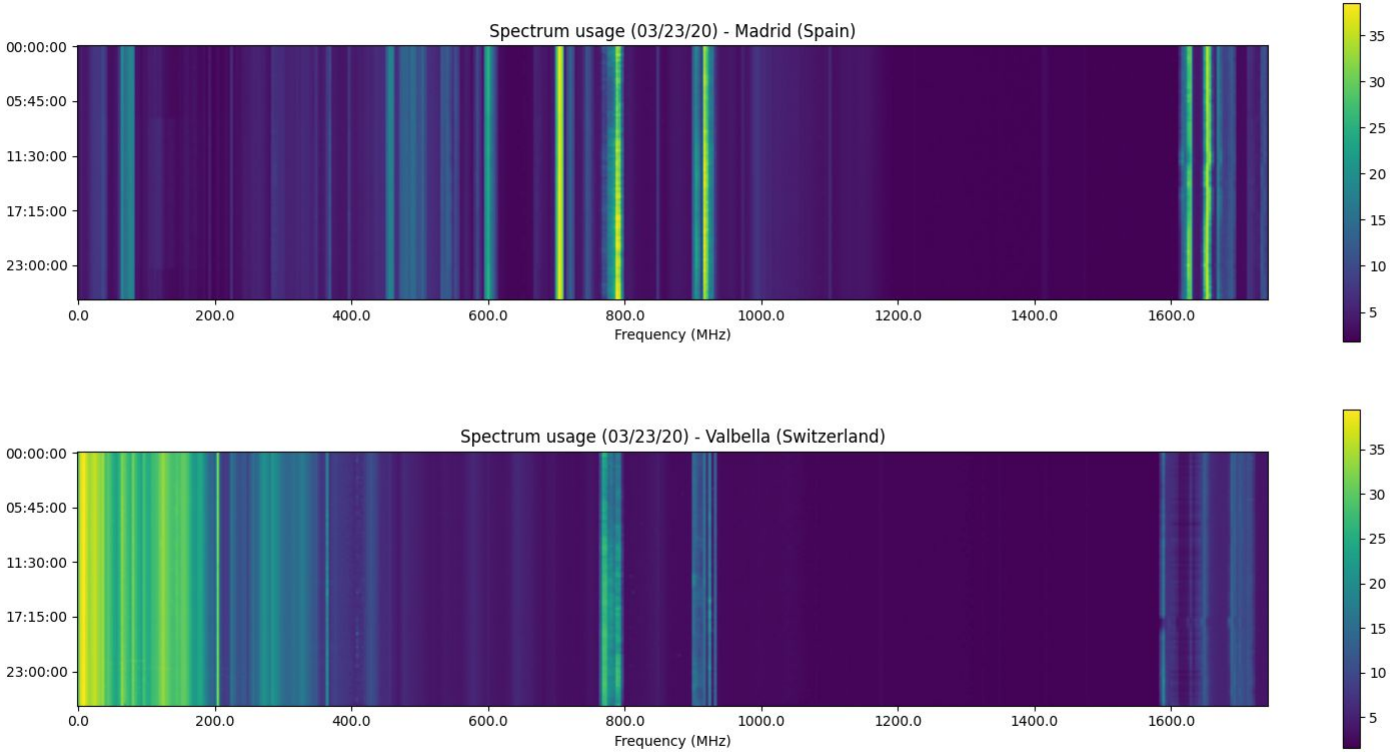
Aggregated data

- Obtaining aggregated data (in time & frequency)
  - No restrictions for own device
  - For the rest (60s and 100kHz)
  - Cannot ask for more than 2 million values
- Query:
  - <base>/spectrum/aggregated?<query_parameters>
- Output is JSON:
  - Parameters of the search
  - Arrays with data for each time value

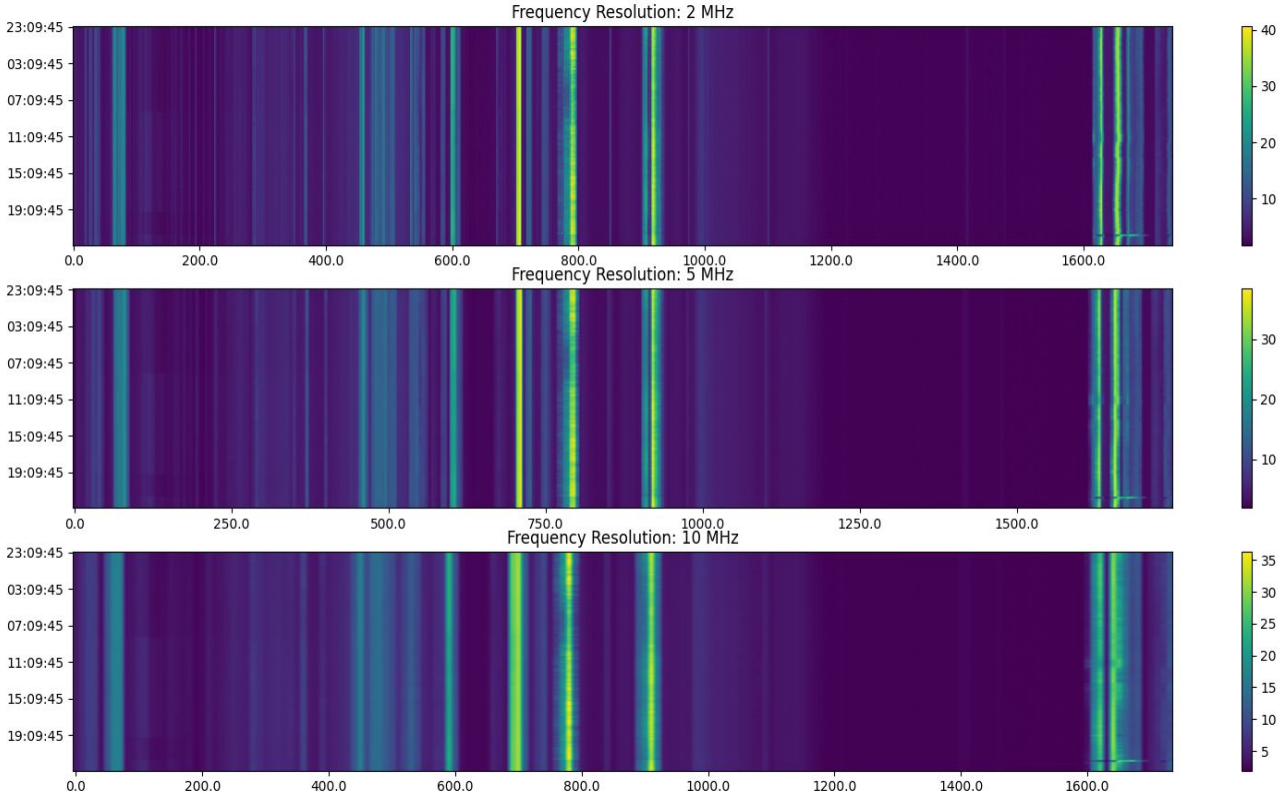
Aggregated data

Name	Required	Description
sensor	Yes	Sensor serial number
timeBegin	Yes	Start time in seconds (Unix timestamp)
timeEnd	Yes	End time in seconds (Unix timestamp)
freqMin	Yes	Minimum frequency in Hz
freqMax	Yes	Maximum frequency in Hz
aggFreq	Yes	Frequency resolution in Hz
aggTime	Yes	Time resolution in seconds
aggFun	No	Aggregating function (MAX, AVG)

Spectrum in different parts of Europe



Aggregated data for one sensor



Raw FFT data

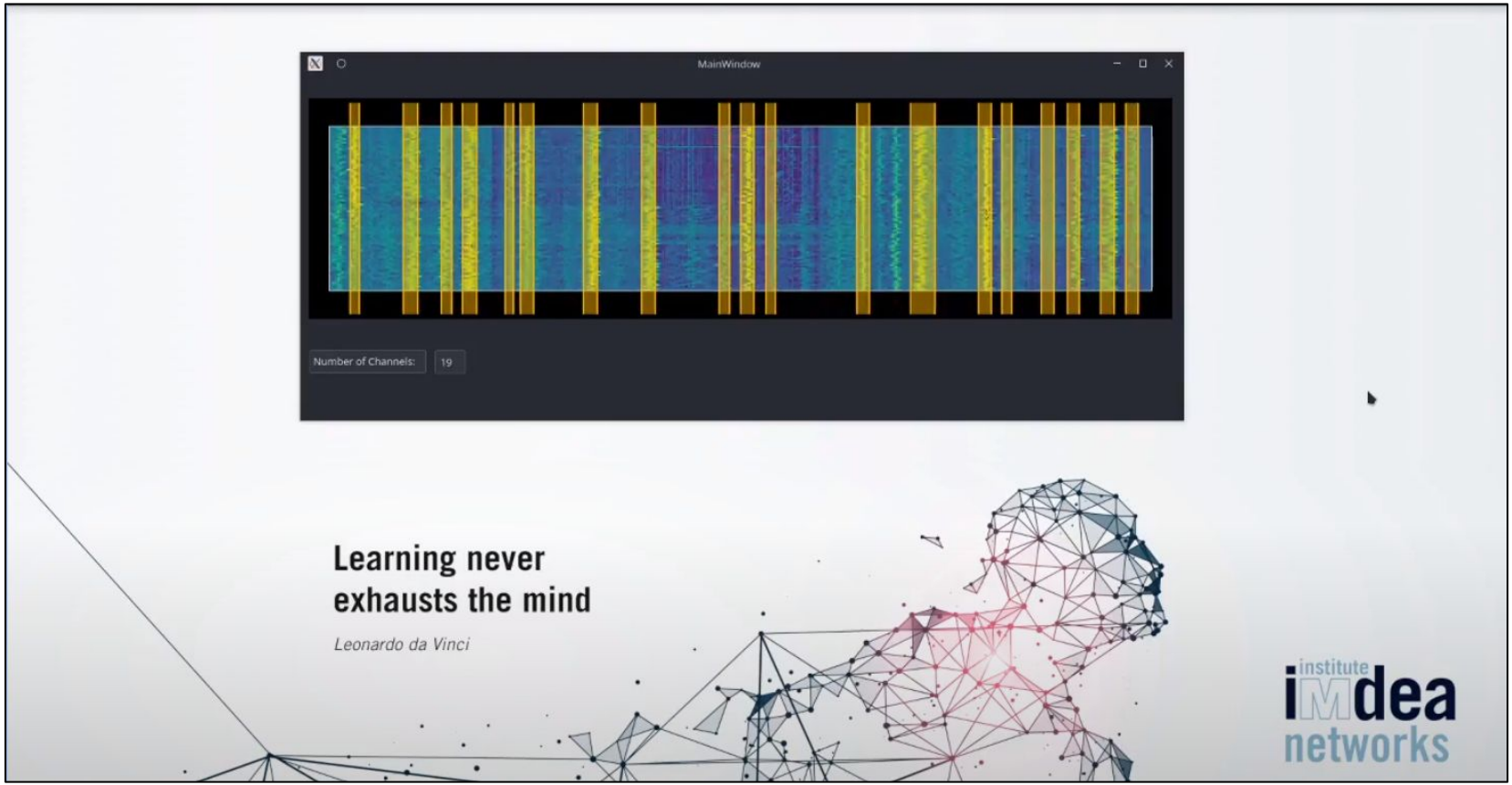
- Power measurements as received by the sensor
- Restrictions:
  - None for own device
  - No access to any other sensor
  - Cannot ask for more than 3 minutes of data
- Query:
  - <base>spectrum/fft?<query_parameters>
- Parameters:
  - **sensor** (required)
  - **timeBegin** (required): Start time in seconds (Unix timestamp)
  - **timeEnd** (required): End time in seconds (Unix timestamp)

Electrosense OpenAPI

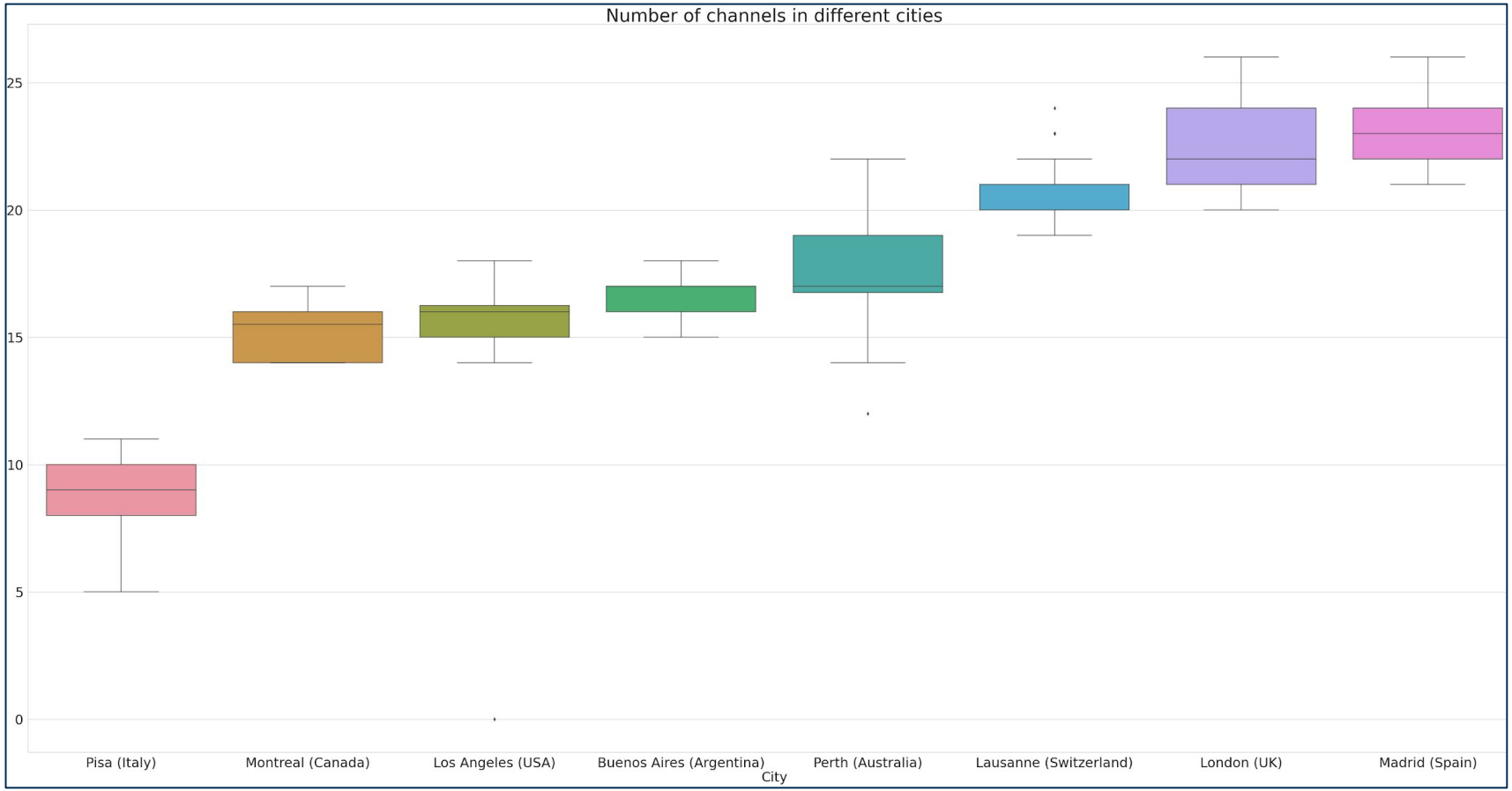
Use cases

FM channels detection

- We want to detect the number of transmissions present
- For this example we focus on the FM band
- We can make use of the API:
  - To get the sensor list
  - Select the sensors we want to observe
- Estimate the number of transmissions over a period of time


MainWindow
Number of Channels:
19
Learning never exhausts the mind
Leonardo da Vinci
institute
iMdea
networks

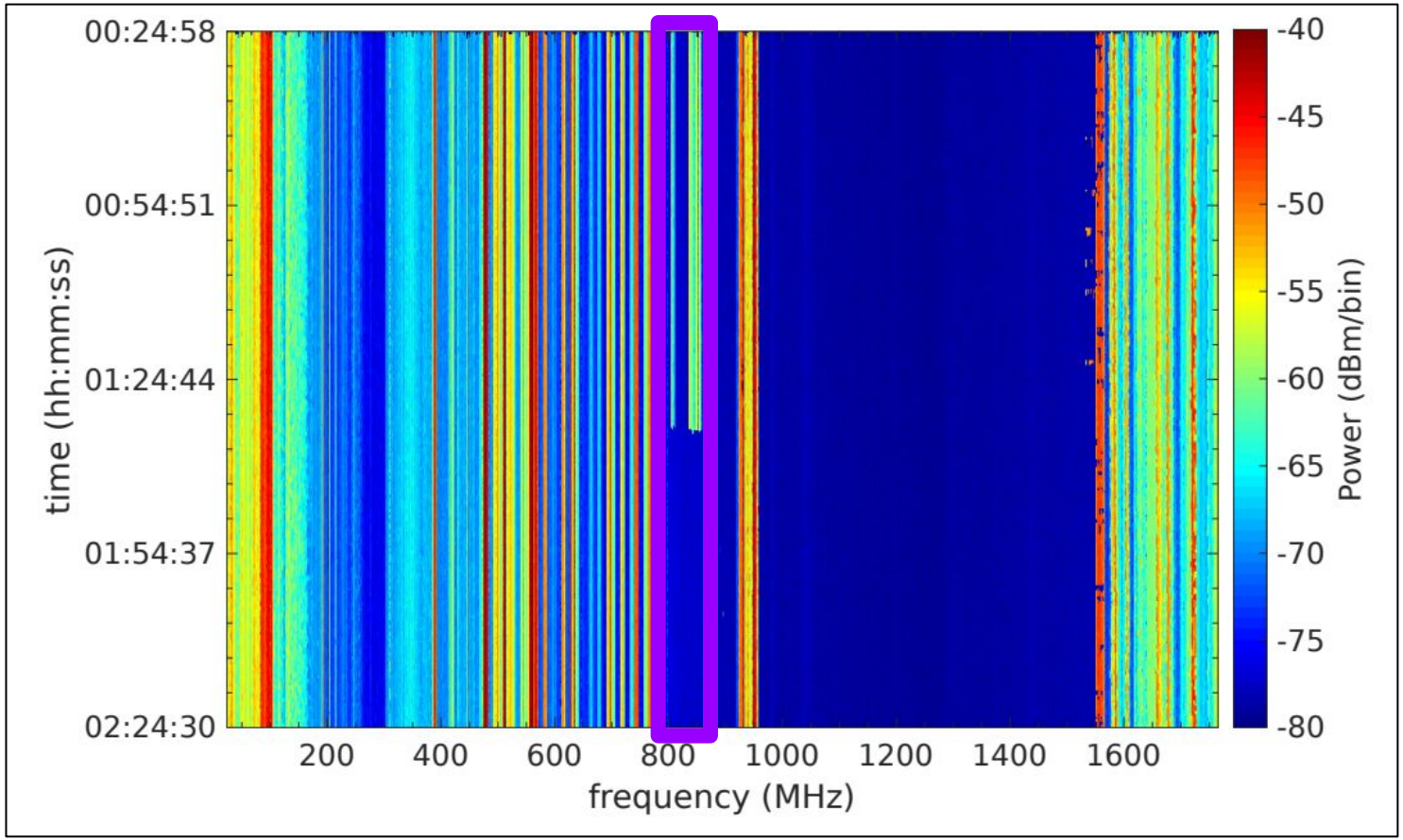
FM channels detection



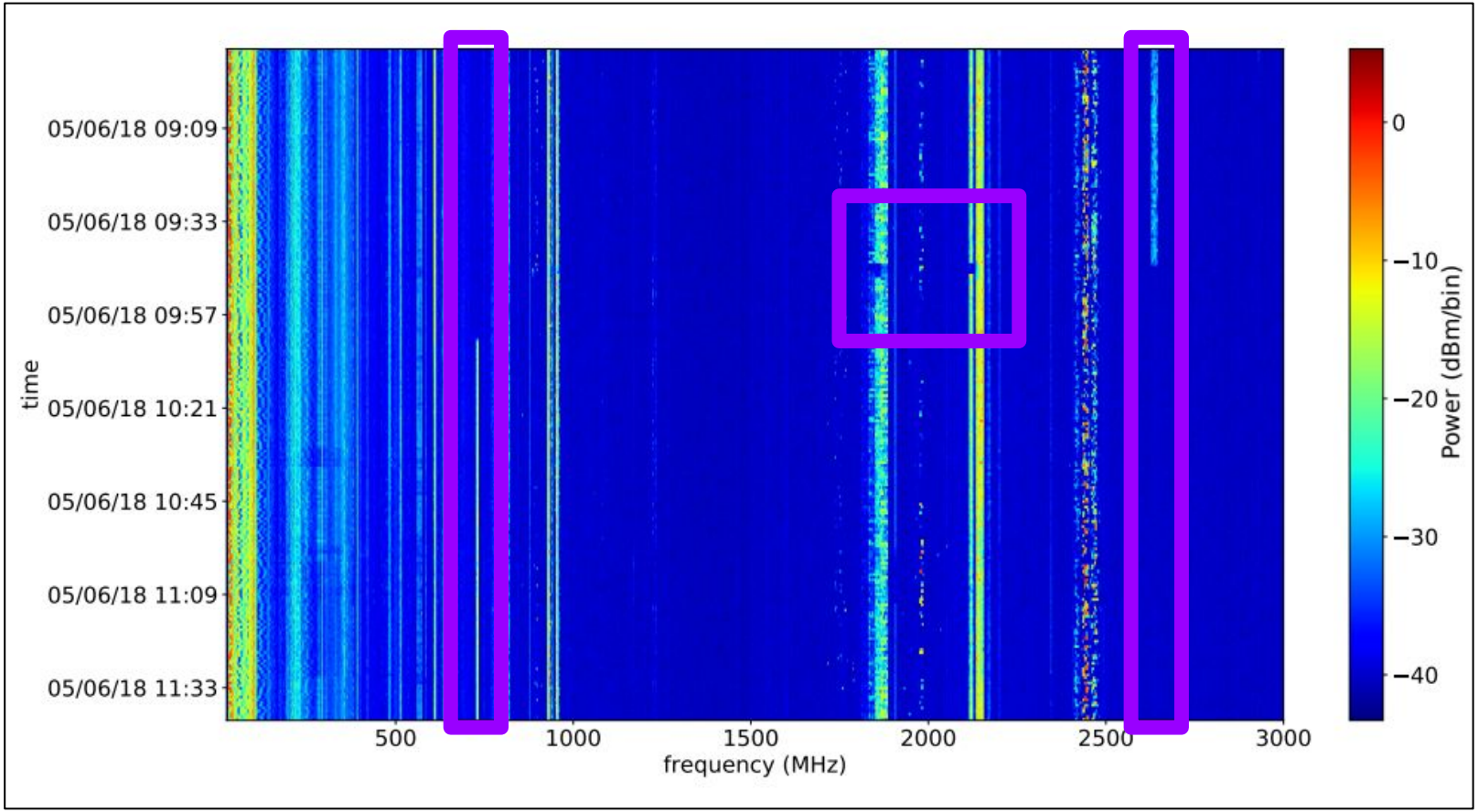
Occupancy over time

- Spectrum occupancy might vary at different times of the year
- We can detect anomalies:
  - Changes in EM spectrum usage
- With the Electrosense API we can obtain power data for long periods of time:
  - Map the evolution of occupancy at different bands

Occupancy over time (DVB-T and LTE)



Occupancy over time (DVB-T and LTE)



uc3m

www.networks.imdea.org

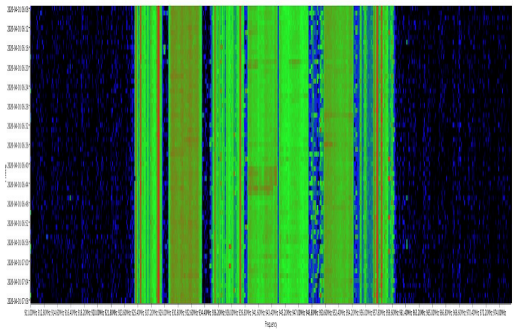
Electrosense OpenAPI

Technology Classification

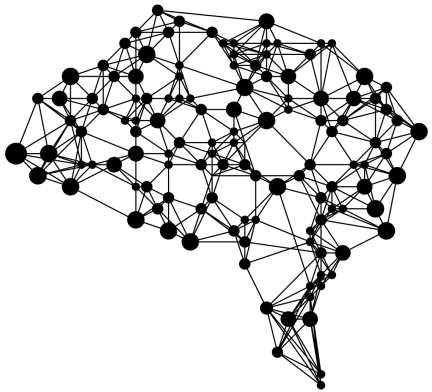
institute imdea networks

Technology Classification

- Build an automatic radio technology classification using aggregated spectrum data from Electrosense collected by low-cost spectrum sensors.
- Given aggregate spectrum data: what is the technology used for?
- We rely on deep learning approach to tackle this problem



INPUT



OUTPUT

FM ?

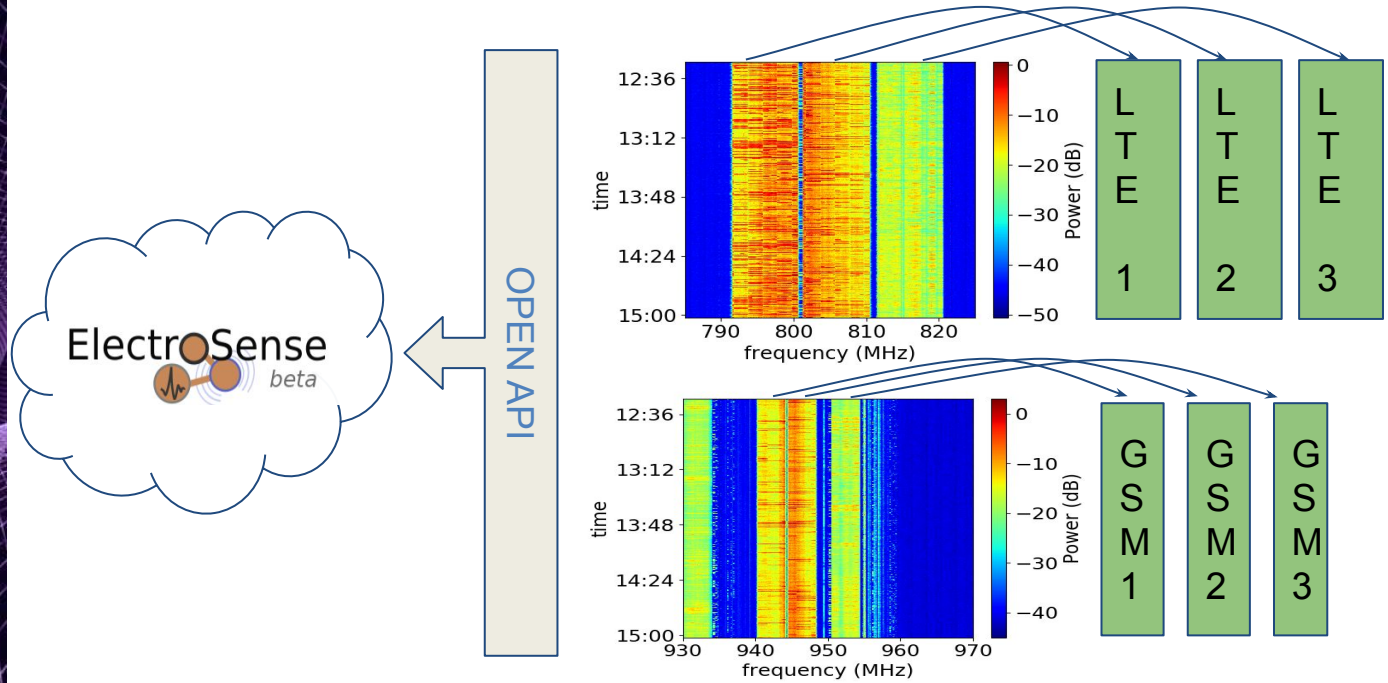
GSM ?

LTE ?

DVB-T ?

Technology Classification

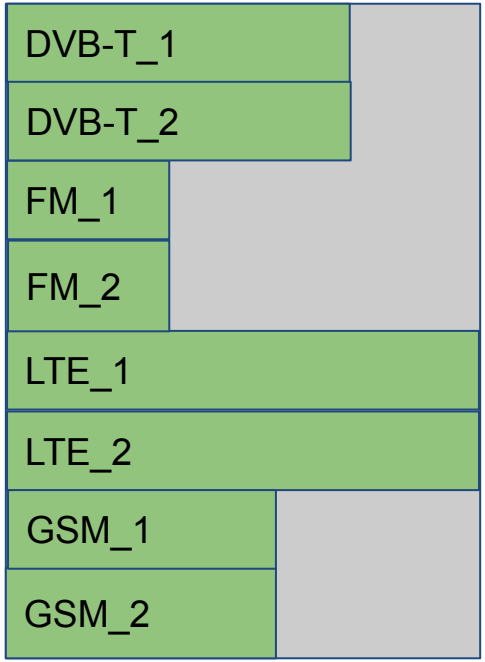
- Spectrum data extraction from Electrosense with Open API
  - Single channel extraction for each Technology



Extraction

Technology Classification

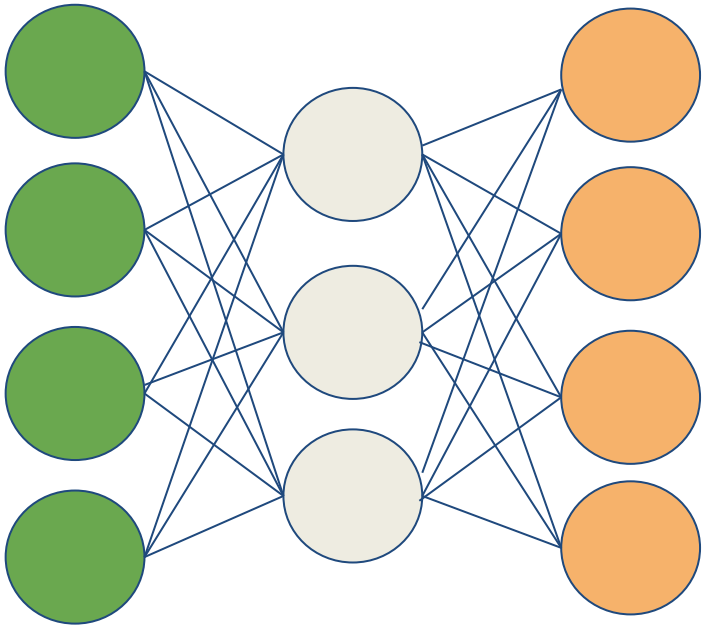
- Spectrum data extraction from Electrosense with Open API
- Labelling data
- Prepare data to be a consistent input of the model



Labelling & Preparation

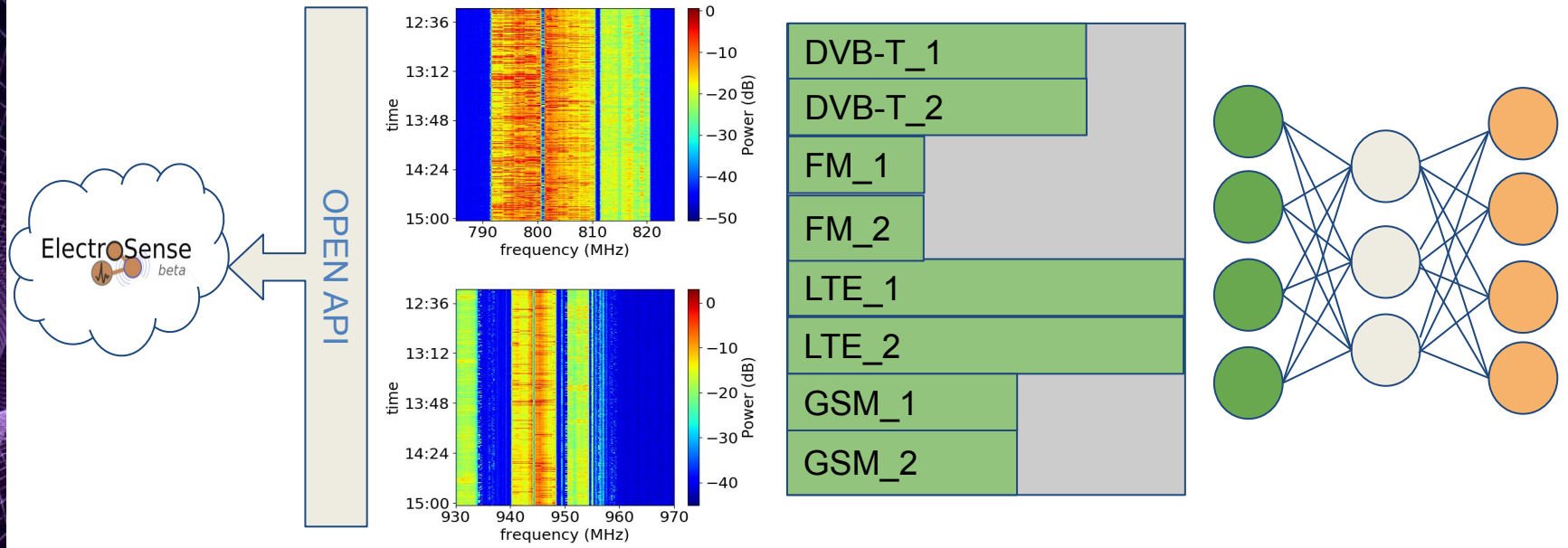
Technology Classification

- Spectrum data extraction from Electrosense with Open API
- Labelling data
- Prepare data to be a consistent input of the model
- Train the DL model with the labeled data



Training

Technology Classification



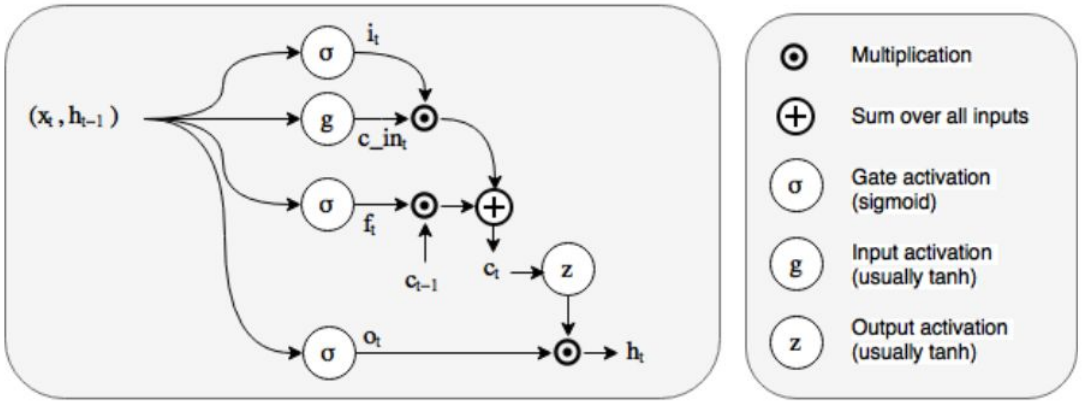
Extraction

Labelling & Preparation

Training

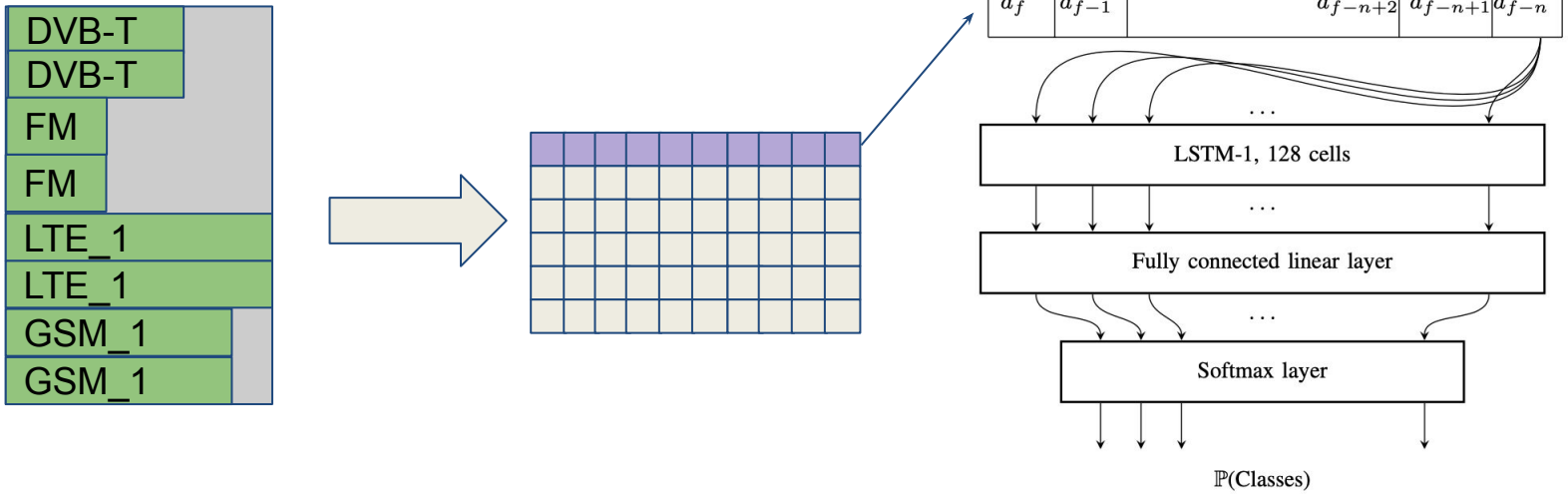
Technology Classification

- Long-Short Term Memory (LSTM) is a kind of recurrent neural network capables to learn long term correlation through the inputs.
- Each cells has an internal memory or state (C_t)
- The output of each cell is a function of the current input x_t and the history h_{t-1}
- Gating mechanism helps to store information into the cells.
- Model learn the gate weights based on:
  - Previous state c_{t-1}
  - The current input



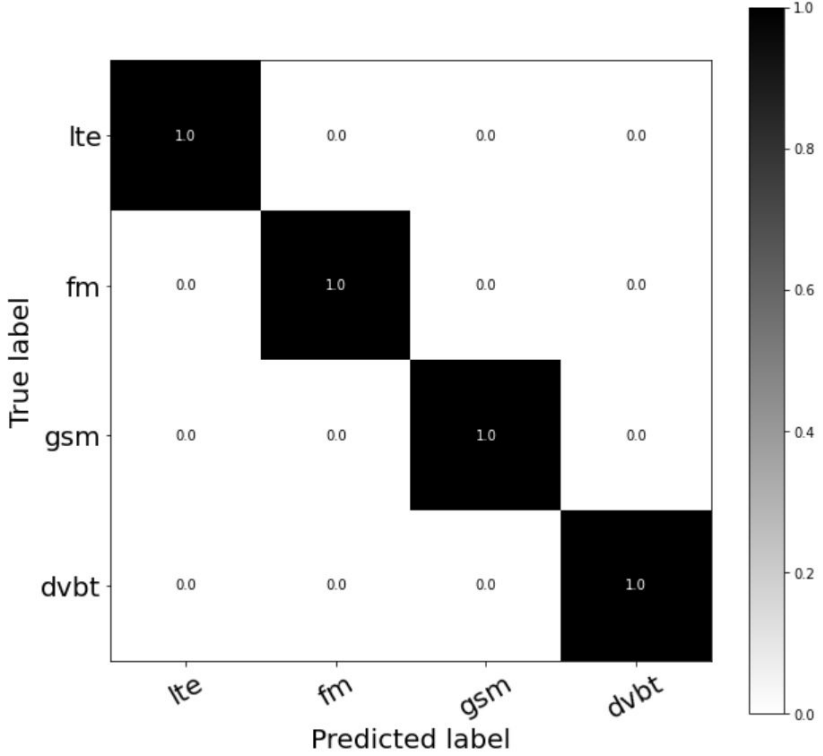
Technology Classification

- Aggregate spectrum data become a matrix.
- Row by row, each column are fed to all cells of the LSTM layer as 1-D vector.
- Output LSTM layer is a vector of dimension 128 that is fed to the last dense layer of the model.
- Softmax layer which maps the input to one of the 4 output classes representing the technology



Technology Classification

- Dataset Size:
  - 4200 samples for each technology
  - 16800 Total samples
- LSTM Hyperparameters:
  - Training data input size: 8200
  - Epochs: 100
  - Learning Rate: 0,001
- Training Time
  - GeForce RTX 2080 Ti
  - ~40 Minutes



Technology Classification

- Open API can support many applications in radio environment.
- High accuracy with spectrum data collected by low-cost software defined radio (RTL-SDR)
- Technology classification can help identify interferences or suspicious transmissions that can be detected

Electrosense OpenAPI

Yago Lizarribar: yago.lizarribar@imdea.org

Alessio Scalingi: alessio.scalingi@imdea.org

Developing the Science of Networks