University of Padova

Department of Information Engineering

Master Thesis in Ict For Internet and Multimedia

Physical Layer Jamming detection: a Machine Learning Approach

Supervisor
Prof. Stefano Tomasin
University of Padova

Co-supervisor
Prof. Stefan Valentin

Master Candidate
Matteo Varotto

Student ID
2037034

Academic Year
2022-2023

"Milano non è Milan, Italia è Milan"
— Zlatan Ibrahimović

Abstract

This paper aims to illustrate the laboratory experience carried out during March-July 2023 at Hochschule Darmstadt having as its goal the writing of a master's thesis.
The initial goal of the project was to use machine learning techniques to analyze the physical characteristics (i.e:ISO/OSI layer 1) of a wireless cellular channel in order to detect the presence of an attacker.
Thus, the expected outcome of the project is to construct a binary classifier, which takes in input information from the wireless channel and outputs the state of the channel through a binary classification: that is, whether the channel is in a state recognized as normal or whether it has been corrupted by the presence of an attacker.
Lab experiences were carried out using software to implement SDR, both user-side and attacker-side. Therefore, the methodologies used to conduct these experiments will be explained, specifying the theoretical background and commenting from a technical point of view on the results obtained.

Contents

Listing of figures

Listing of tables

Listing of acronyms

SDR . . . . . Software Defined Radio

IoT . . . . . Internet of Things

AI . . . . . Artificial Intelligence

ML . . . . . Machine Learning

NN . . . . . Neural Network

CNN . . . . . Convolutional Neural Network

AE . . . . . Auto Encoder

SVM . . . . . Support Vector Machine

MSE . . . . . Mean Squared Error

CAE . . . . . Convolutional Auto Encoder

1 Introduction

Cellular networks have become an essential part of modern society, transforming the way we communicate and so access information.
Given the mass use of smartphones and other mobile devices, these networks have revolutionized the way people interact and connect with the world around them.
Suffice it to say that until 30 years ago the idea of communicating remotely with other people via a connection that was not wired, such as landline telephones or early Internet connections via Ethernet, was unthinkable for an ordinary citizen.
The arrival of these devices in the early 1990s thus marked a turning point in the way people interfaced with the world, allowing them to access more and more information.
Over the years and with the advancement of information technology, these devices have become more and more intelligent and capable of storing more and more information within them. Suffice it to say that in this decade a person through their smartwatch, a device about 5 centimeters in diameter, is able to receive calls, text messages, and pay at the supermarket.
Given the enormous potential of these technologies to transmit or receive information, they are not only used for civilian use, but also for industrial use.
The advent of so-called IoT has made it possible to radically change the way factories are designed and conceived, making the production of them more efficient.
The market therefore for these technologies has been growing steadily in recent years, in fact:

- Experts expect the global IoT in manufacturing market size to grow from USD 33.2 billion in 2020 to USD 53.8 billion in 2025 at a Compound Annual Growth Rate of

10.1%. [1].

- Experts highlight that discrete Manufacturing, Transportation & Logistics, and Utilities industries will spend $40B each on IoT platforms, systems, and services.[2]

The advent, therefore, of these technologies in areas considered critical to a business or a governament has triggered the emergence of new attacks aimed at compromising the integrity of the proper functioning of these technologies.
Some examples of such attacks are:

- **Jamming**: is a tool used to prevent wireless devices from receiving or transmitting radio information.
  Jammers block the use of devices by sending jamming radio waves on the same band used to transmit information. This, for example, causes interference that inhibits communication between cell phones and repeater towers, paralyzing all telephone activity within its range. On most cellular phones what appears during such jamming is simply a no-network signal.
  In fact, the smartphone interprets the incapability to transmit information as the absence of a cellular network.

- **Covert channel**: is defined as any communication method used to communicate and/or transfer information in a covert and stealthy manner. The primary purpose in using a covert channel is to overcome the security policies of systems and organizations. There are multiple types of cyber threats that can affect the multilevel security (MLS) of ICT infrastructure and systems, and they are increasing daily at an impressive rate. Any shared resource as a bandwidth of a spectrum can potentially be used as a covert channel and this makes everything more difficult, in other words, covert channels are not everywhere but can be anywhere.

After briefly describing the possible attacks that can be carried out, it is easy to deduce that all companies that want to remain competitive in the market need to develop prevention systems for these types of attacks, so that a malicious attacker cannot jeopardize, for example, the continuity of a factory's production.
This aspect turns out to be very important when entering the economic-business world, since in the event of a stop in production, the damage done to the company itself can be considerable. Without going into technical-economic details, it should be considered useful to show a case of this kind, in order to get familiar to the reason why security solutions will take hold on the market more and more in the next years:

- In March 2022 a Toyota facility in Japan was cyber-attacked and the production was affected for more than a day and influenced about 10000 vehicles, which is equivalent to 5% of the production of a month of the group in Japan.[3]

This then primarily explains the reason why the previously described project took place in the first place, which is to try to study more robust solutions than those currently in the literature that can go about detecting possible attacks and intruders present in a wireless cellular network.

2

Wireless channel

2.1 Jamming on a wireless channel

3
Machine Learning

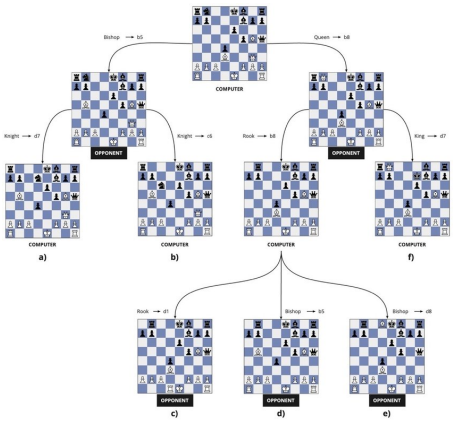
3.1 Introduction

The term Artificial Intelligence (henceforth AI) was invented by John McCarthy in 1956, at a two-month seminar (which he organized at Dartmouth College in Hanover, New Hampshire, USA) that had the merit of acquainting 10 U.S. scholars (on automata theory, neural networks and intelligence) with each other, and of giving the imprimatur to the term "Artificial Intelligence" as the official name of the new field of research.[4]
Since then, AI has established itself and evolved; today it is recognized as a branch autonomous, although connected to computer science, mathematics, cognitive science, neurobiology and philosophy.
Artificial Intelligence therefore represents a field of research and development that aims to create systems that are able to emulate and automate some human cognitive functions.
AI then is the ability of a machine to mimic some of the human cognitive functions, including machine learning, reasoning, planning, sensory perception, natural language understanding, and social interaction. AI can be divided into two main categories: **weak AI**, which focuses on specific and limited tasks, and **strong AI**, which aims to develop a machine with general intelligence comparable to human intelligence.
These goals nowadays are reached in two ways:

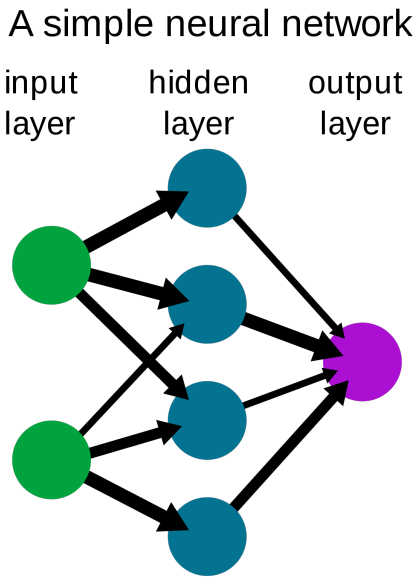
- As a first approach, AI can be created by telling to a model the rules to follow to solve

problems or to take decisions.
This is the case with the implementation of a machine attempting to play chess: here the chessboard is modeled and each possible move in the subsequent rounds is ranked according to its quality (via trees for example; see figure below) by predetermined heuristic algorithms, and the machine is required to choose the move that is rated most convenient, so as to maximize the probability of victory.
It is clear then that only simple-modelled problems can be solved by setting some set of rules in order to make the machine behave as a human; when it comes to deal with more complex structures (such as images, videos or texts) this approach shows its limits.

- The most common way nowadays to deal with more complex data structures is the so called **Machine Learning** which is a branch of AI that allows machines to learn from data without being explicitly programmed to solve that specific problem, thus making a model trainable to solve multiple problems.
  It is easy to see that ML radically changed the approach to solve problems: whereas before we tended to study a problem to find the rules for solving it, now we tend to create a model to which we feed solved examples of that problem (so-called supervised learning) so that it learns to solve that problem on its own.
  The machine learning literature nowadays is widely developed and includes various approaches, ranging from a simple linear separator such as the perceptron to the more modern and now commonplace artificial neural networks, which allow extremely complex data structures to be handled and have extremely high performance.
  Without going into details (which will be explained later), it is possible to describe NNs as models that try to replicate the behavior of the human brain: a set of neurons (main elements of the network) are connected in order to take a datum as input and make a prediction or a decision attempting to minimize an error that is specified a priori during the training phase.
  A simple example will be shown below.



(a) Example of a decision tree for chess playing



(b) Example of a simple feed forward NN

Figure 3.1: Two examples of AI.

As it was said before, the goal of the project was to apply a machine learning model in order to detect a jamming attacker; to understand the reasons why some models have been preferred to other ones, the most common ML models will be briefly explained.

3.2 Perceptron

The perceptron algorithm is a simple supervised learning problem that tries to solve a binary classification problem of linearly separable data.
The input of the model is a set $X=(x_1,\ldots,x_n)$ of data and $Y=(y_1,\ldots,y_n)$ of the corresponding labels. The goal of the algorithm is to find a vector $W=(w_1,\ldots,w_n)$ of weights representing a separating hyperplane such that the classification error is avoided. Whenever a classification error is found, the vector W is updated. [5]
Recall that data is linearly separable if, given the training set composed by X and Y and the halfspace defined by (w, b) we have:

$$\forall i : (< w, x_i > + b) y_i \geq 0$$

i.e.: it perfectly separates all the data in the training set.
Here below a pseudocode of the iterative algorithm:

Algorithm 3.1 Perceptron algorithm

Require: $X = (x_1, ..., x_n)$, $Y = (y_1, ..., y_n)$
  $w_1 \leftarrow (0, ..., 0)$
  for $t \leftarrow 1$ to ...
    if $\exists i$ s.t. $< w_i, x_i > y_i \leq 0$
      $w_{t+1} \leftarrow w_t y_i$
    end if
  else
    return w_t
  end for

From the pseudocode above it is easy to see that this algorithm could be easy implemented in any high level programming language nowadays and it is also demonstrated that if data is linearly separable the algorithm will stop in a finite number of steps.[6]
This model, however, shows its limitations:

- First of all, the convergence is not guaranteed when it is not dealing with a linearly separable training set, which is possible in more complex data structures.
- This algorithm can output different solutions depending on the starting values of the vector W.
- The output of this classifier can only be a binary classifier, and so cannot be used in multiple class classification problem.

3.3 Support Vector Machine

After recalling the definition of linearly separable data (see section above) before talking about SVM, it is useful to definine the concept of margin, which will be useful later.
Given a separating hyperplane defined by L:

$$L = \{v :< v, w > +b = 0\}$$

and a sample x, the distance between x and L is:

$$min\{||x - v|| : v \in L\}$$

The margin then, is defined as the minimum distance between a sample and L. [7]
The closest samples are called **support vectors**.
Here below an example of how SVM works:

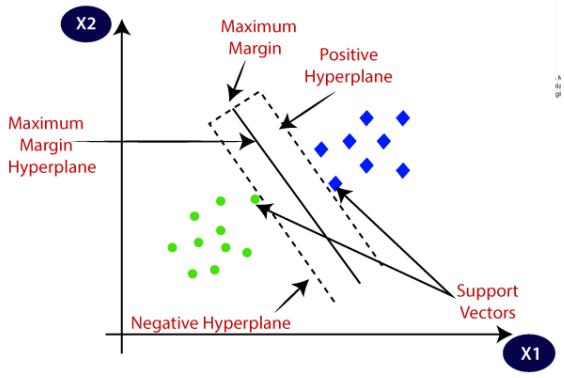


Figure 3.2: Representation of how SVM works in 2D dimensional space.

SVMs are also divided in two types:

- **Hard SVM**: a linear model that work with linearly separable data.
- **Soft SVM**: a linear model that work with non-linearly separable data.

3.3.1 Hard SVM

Hard SVMs seek for the separating hyperplane with the highest margin, under the assumption that the data is linearly separable. The mathematic formulation of Hard SVMs is expressed as follows:

$$\underset{(w,b):||w||=1}{argmax} \; \underset{i}{min} | < w, x_i > + b \; |$$

3.3.2 Soft SVM

As mentioned earlier, hard SVM has the main problem that it assumes linearly separable data, which is impossible in most problems found in case studies.
Soft SVM therefore relaxes the previously set constraints taking into account their violation at the same time.
This implies in relaxing the constraints as follows:

- First of all, a set of slack variables are introduced: $\xi = (\xi_1, ..., \xi_n) : \xi_i \geq 0 \ \ \forall i$.
- For each $i = 1$ to n the constraint becomes : $(< w, x_i > + b)y_i \geq 1 - \xi_i \ \ \forall i$.
- The model then tries at the same time to minimize the norm of w (to maximise the margin) and the average of ξ_i (to minimize the violations of the constraints).
- The objective function of the optimization problem then becomes:

$$\min_{(w,b,\xi)} (\lambda ||w||^2 + \frac{1}{n} \sum \xi_i)$$

  Subject to the constraint defined before.
  It is clear then that a large λ makes the algorithm focus on the margin, while a small value of the variable makes the model to minimize the constraint violations.

This reformulation of the SVM model thus represents a way to solve a binary classification problem while having more complex data structures, however, which are not always linearly separable.[8]

3.4 Neural Networks

As mentioned earlier, neural networks are models whose structure is inspired by the functioning of an animal brain.
As a first general overview, a neural network can be regarded as a non linear mathematical function which transforms a set of input variables into a set of output variables. [9]
The network can me modeled as an acyclic graph $G = (V, E)$, divided into layers :

- The vertexes of the graph are the **neurons**, which take in input the sum of the outputs of the connected neurons from previous layer weighted by the edge weights and applies to this result a simple scalar function called activation function.

- The edges of the graph connect a neuron to other neurons of the next layer; to each edge is associated a weight. This implies also that the edges are directed and the flow is performed only in one direction.

The computation of the output is done by processing the input at each layer and forwarding it to the next layer, until the output layer is reached.
The activation functions of the neurons are defined a priori and cannot be changed during the training phase, making so the weights the trainable parameters.
Given a training set composed by $X = (x_1, ..., x_n)$ and $Y = (y_1, ..., y_n)$, where X is the training dataset and Y is the set of labels associated to the data of X and given a loss function l, the goal of the NN is to find the optimal values of the weights in order to minimize the loss L in the training set, that is:

$$L = \frac{1}{n} \sum l(x_i, y_i)$$

To do that, the most common way nowadays is the so-called **backward propagation** algorithm: at each epoch, the loss of the output is computed, and after that, it is propagated backwards to the input. Using the gradients calculated during error back-propagation, the network weights are updated. The update rule for the weights is defined as follows:

$$w_{ij}^{(t)[s+1]} = w_{ij}^{(t)[s]} - \eta \frac{\partial L}{\partial w_{ij}^{(t)[s]}}$$

Where $w_{ij}^{(t)[s+1]}$ stands for the weight at layer t computed at iteration s+1.
The gradient so is computed for each weight of the network, but after having computed the output at the last layer.
This procedure is repeated for many iterations, after a stopping criteria is reached, that could be:

- The maximum number of iterations.
- The reaching of a value in the training loss.
- The increasing of the validation loss for n epochs. (Recall that the validation set is a set with the same distribution of the training set but on which the NN is not trained; it is often used to set hyperparameters the ML models).

The simplest example of NN that can be shown is the so called *feedforward*: in this case each neuron is connected to every neuron of the next layer and the output is propagated from

the input layer throught the hidden layers and then to the output layer; an example was shown above in figure 3.1.b.
Here below will be shown a graphical representation of how a neuron works:

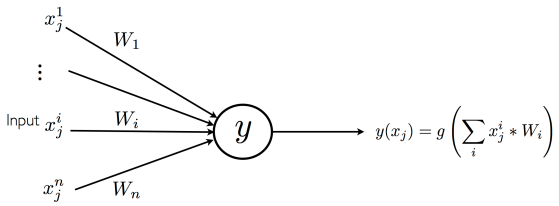


Figure 3.3: Representation of how a neuron works.

3.4.1 Convolutional Neural Networks

A convolutional neural network is one of the most significant networks in the deep learning field. Since CNN made impressive achievements in many areas, including but not limited to computer vision and natural language processing, it attracted much attention from both industry and academia in the past few years.[10]
The CNN is a kind of feedforward neural network that is able to extract features from data with convolution structures. Different from the traditional feature extraction methods, CNN does not need to extract features manually.
The main problem that occurs in feedforward NNs is the fact that the model does not take into account that a neuron can be more related to others and less to other ones; in fact, in this case every neuron is connected to each neuron of the next layer, without taking into account any kind of correlation.
CNNs then use some structures to achieve feature extraction:

- **Local connections**: each neuron is no longer connected to all neurons of the previous layer, but only to a small number of neurons, which is effective in reducing parameters and speed up convergence.

- **Weight sharing**: a group of connections can share the same weights, which reduces parameters further.

- **Downsampling dimension reduction**: a pooling layer harnesses the principle of image local correlation to downsample an image, which can reduce the amount of data while retaining useful information. It can also reduce the number of parameters by removing

trivial features. These three appealing characteristics make CNN one of the most representative algorithms in the deep learning (see definition below) field.
The dimensional reduction is also reached using **pooling layers**: at first used to reduce computational complexity, but turned out to be crucial to improve performance in many applications since they increase the receptive field of the inner layers. One of the most common pooling layer is the max-pooling, that it takes in input a window of samples and outputs the highest value of it.

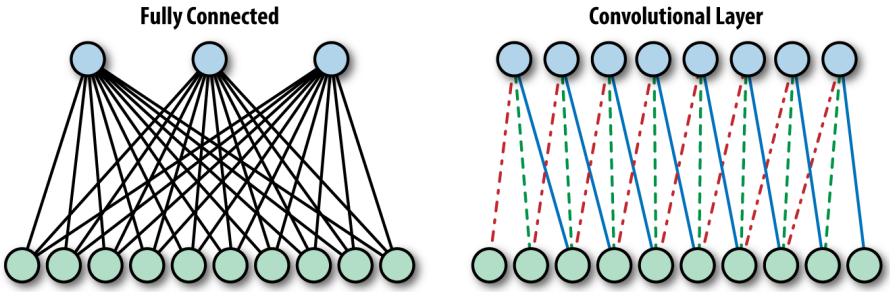


Figure 3.4: Graphical representation of the difference in terms of connections between feedforward NNs and CNNs.

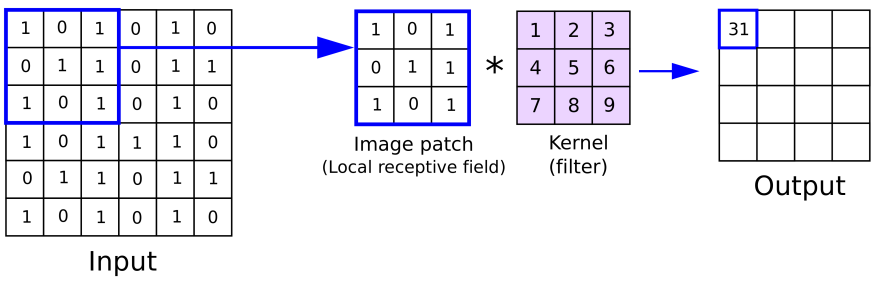


Figure 3.5: Graphical representation of how a convolutional filter can be viewed to process a 2D matrix.

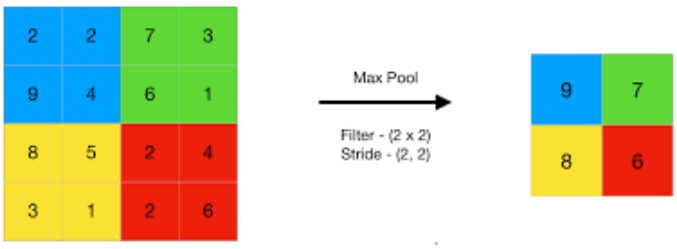


Figure 3.6: Graphical representation of a max pooling layer applied to a 2D matrix.

From figure 3.5 it is easy to see how convolutional filters can be represented as windows that slide along the input data.
To design a better CNN, it will be very important to set properly the **size** of the windows and the **stride**.
The term **deep learning** was previously mentioned: it is a branch of machine learning that defines all models of NNs that have many layers.

4

Jamming detection with machine learning

As mentioned in Chapter 2, the informations that can be obtained from a wireless channel at level 1 of the ISO/OSI stack are two:

- The power of the signal among a bandwidth, the so called **spectrogram**.

- The sampling of the signal in two coordinates (in phase and quadrature) along a time interval. The result of this operation leads to the realization of the so-called **i-q diagram**.

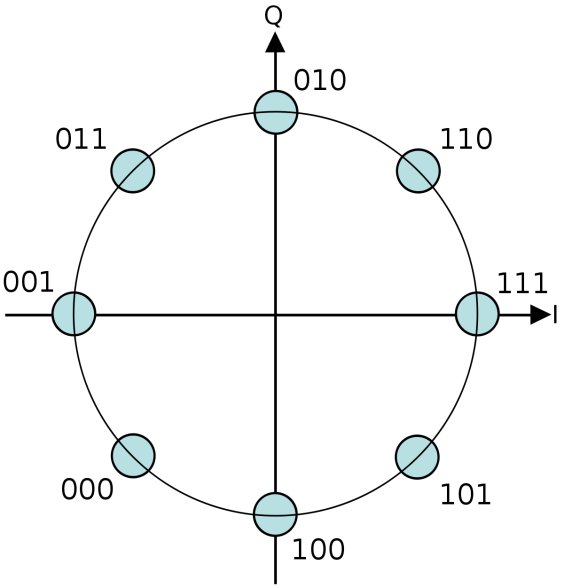


Figure 4.1: Example of an i-q diagram with 8psk modulation.

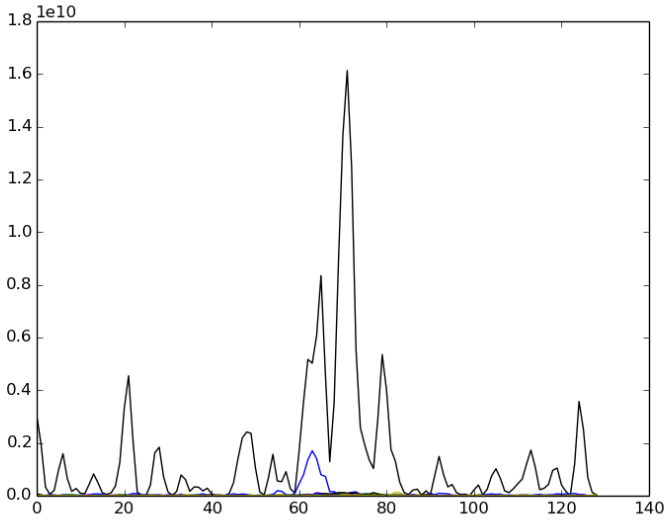


Figure 4.2: Example of a plot of a the power spectral density of a general signal among a bandwidth of 140 Hz.

The goal of the project, then, is to use this informations to create a model that can distinguish a channel in states deemed *trusted* from a channel corrupted by the action of a jammer.
This allows us to say that the goal of the project will be to solve a binary classification problem:

the output of the ML model will be the predicted class of the wireless channel, that will be: *trusted* or *jammed / anomal*.

4.1 Jamming detection on i-q diagrams

An i-q diagram is a collection of i-q samples over a period of time; the more the time resolution is high, the less there will be i-q samples in our plot.
To make it more clear, it is easy to think that for a given sampling rate collecting the double of the samples will take the double of the time.
After having determined the time resolution, it is necessary to decide the way to represent an i-q diagram in a computer; there are two ways:

- The first one is to represent our **n** samples as an array of tuples of length two (recall that each sample has two coordinates: in phase and quadrature), then use this data structure to classify the input data.
  This task can possibly be done by all of the ML models explained in chapter 3.
- The second one is to plot the samples in a 2D plane and use the plot to classify the channel. This kind of problem can be easily linked to an image classification problem, which is well implemented in the state of the art by CNNs.

After having clear in mind of how the data can be represented, the next choice to be done is the ML model that will be able to solve the binary classification problem explained above.

- At first, the choice to use the perceptron algorithm was discarded, due to the high dimensionality of the input data (recall that it would be represented as an array of **n** samples and each sample will have two coordinates) and the not guaranteed convergence during the training phase.
- The SVM model was also discarded because it is reported that SMV with high dimensional data tend to use a lot of memory and loose performance in terms of accuracy. (remember to cite the paper, TBD)

The choice therefore fell between the standard feedforward NNs and the CNNs.
Due to the advantages of the CNNs explained above, it was chosen to represent data as actual i-q plots (images) and feed a CNN with that kind of data to solve the binary classification problem.
Using images to represent i-q plots also allows to train the same model on different time resolution cases, due to the fact that the images will have the same spatial resolution.

This will allow later to use the same model and compare 3 different time resolutions, which would not be possible if there were used other kind of data representations.
Modern literature shows two ways to solve this problem of image classification for jamming detection:

- The first one is the standard deep learning image classification model: it consists into taking as an input an image, apply several convolutional and pooling layers and end the NN with a final feedworward part that will have as the last layer **n** neurons, where n is the number of classes of the dataset.
  The output of the NN will be an array of probabilities that the input belongs to a class, and the predicted class will be the one with the highest score.

- The second one is to perform an anomaly detection problem using a deep learning model called **autoencoder**: an autoencoder is a deep learning model that takes an input data and by performing several operations in the inner layers tries to reconstruct the original input.
  The model is composed by two parts: the **encoder**, which takes the original input and transform it to a lower dimensional space data, and the **decoder**, which takes as input the output of the final layer of the decoder and tries to reconstruct the original input.
  This kind of model implies that the classification is performed not on the labels of the input data, but on the reconstruction error (e.g.: MSE): is during the training phase the model is fed only with *trusted* images, it will learn how to reconstruct in a proper way only that kind of data, which implies that the so called *anomal* data will be reconstructed with an higher error. This implies also that the training phase is indipendent on the anomaly data.
  For the reasons explained above, the model choosen to perform the classification task on the cellular channel was the autoencoder.

By setting a threshold on the reconstruction error, the classification task between *trusted* and *anomal* data can be easily implemented, here below a pseudocode example:

Algorithm 4.1 Autoencoder classification task

Require: $X = INPUT\_DATA, Model = AUTOENCODER(), Threshold$
  $recon\_error\_x \leftarrow Model.compute\_reconstruction\_error(X)$
  if $recon\_error\_x > Threshold$
    return 0
  end if
  return 1

From this pseudocode it easy to see that with this approach is not possible to perform a multiple classification problem (even if the *trusted* cases belong to multiple classes), but it is possible to perform only an anomaly detection problem, which perfectly fits into the task of the project: in fact, it is better that the model knows only the normal behaviours of the wireless channel, in order to have a better response on attacks of different kind.
Here below an example of a simple CNN for image classification task and of an autoencoder:

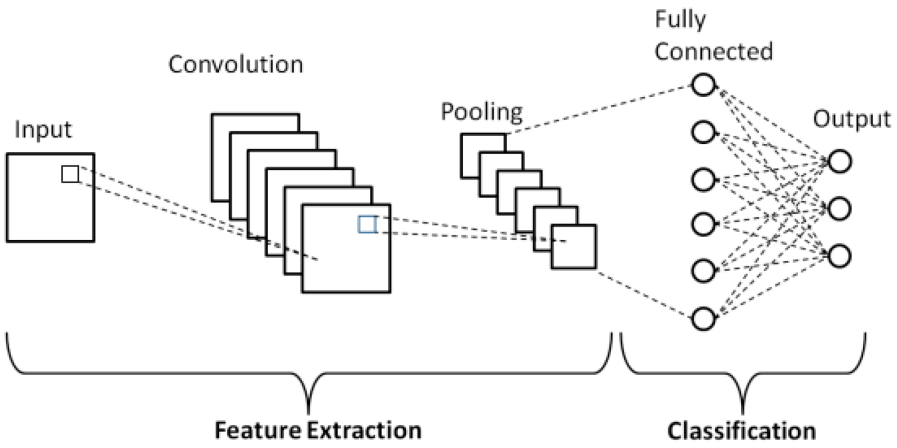


Figure 4.3: Example of a CNN for image classification

Fom this is image it is easy to see that the first layers are dedicated to the feature extraction task (pooling and convolutional layers) and then the NN in the last parts becomes feedforward and ends with **n** neurons, three in this case.

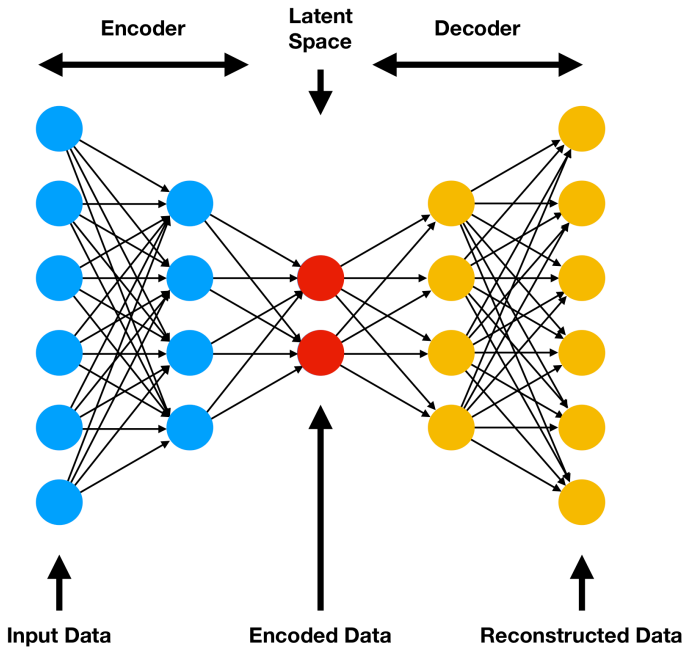


Figure 4.4: Example of an autoencoder.

Here above then is clear how the model reduces the input dimensionality to a latent space (output layer of the encoder) and then tries to reconstruct the original input.
For the reasons explained above, it was chosen to use a **Convolutional Auto Encoder** to perform anomaly detection on the images of the i-q plots, structured as follows:

- The first layers will be convolutional, in order to perform feature extraction on the image and reducing its dimension.
- Then, there will be some fully connected layers until it will be reached the latent space, in order to add parameters to the network and so perform better generalization, due to the fact that convolutional layers contain less trainable parameters than fully connected ones.
- The structure of the decoder (from the latent space to the output layer) will be symmetrical to the one of the encoder, with some upsampling operations in order to have in the output layer the same dimension of the input layer.

Due to the fact that the network is trying to reconstruct an image, the chosen loss function will be the MSE, computed as follows:

$$\frac{\sum_{(i,j)\in M}[x(i,j) - y(i,j)]^2}{N}$$

Where:

- M is the input image, which is seen as a matrix from the model perspective.
- i and j are the pixel raw and column indexes.
- $x(i,j)$ and $y(i,j)$ are respectively the value of the pixel in the input image and in the output image.
- N is the number of pixels contained in the input image.

The details and the performances of this CAE adopted will be analyzed in the next chapter.

4.2 Jamming detection on waterfall plots

As it was said before, the other physical layer aspect that describes the wireless channel is the spectral power.
Using this aspect as input data, to perform jamming detection there will be created and analyzed the so-called **Waterfall Plots**.
Waterfall plots as a first approach can be seen as a sequence of PSD arrays stacked in order to create a matrix, built in this way:

- The X axis is the frequencies axis, centered on the carrier frequency, such as PSD arrays.
- The Y axis is the time axis.
- The value contained in the matrix represents the spectral density in a binned frequency range in a time slot.

Due to the description given above, it is clear that waterfall plots represent the evolution of the power spectral density over the time of the wireless channel. Using this kind of data structure, there will be explained the aspects of the channel that will be investigated in order to see if there is the presence of a jammer or not.
Waterfall plots can be represented in two ways:

- The first one is a raw matrix containing the power values for each cell of the matrix. This data structure is fast to compute, but is not so well human-readable.

- The second one is to convert the values of the matrix into a colorscale, which implies to convert the matrix into an image, usually RGB. This kind of data structure implies a better human-readability, but it involves the disadvantage of having an higher computation time, due to the fact that each information contained in the matrix has to be converted into a pixel of the output image.

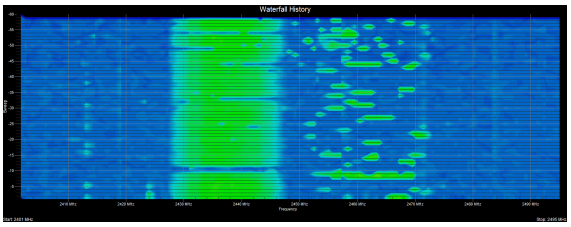


Figure 4.5: Example of a waterfall plot with the colorspace representation.

During the data creation phase, it was noticed that creating waterfall plots with the colorspace was more time consuming compared to creating raw matrixes with the raw power values inside each cell of the matrix.
In fact, creating a dataset with four thousands images took about 2 days, while creating a dataset with the same number of matrixes took about 15 minutes.
For this reason and due to the fact that both representations are 2D structures (and so the same ML structures can be used to analyze them), it was chosen to use a dataset composed by raw matrixes.
Due to the nature of the input data explained above, it was chosen to use also in this case a CAE, which is structured in a similar way as explained above in the i-q plots case.
While the model to analyze data is the same, the way how results will be seen is different. In the first case, the goal of the model was to distinguish between a trusted i-q plot and a jammed one; here instead the goal is to find out portions of the spectrum that result anomal according to the normal behaviour.
For example, in this case is not so meaningful to identify a globally jammed waterfall plot, because a smart jammer could act on a tiny portion of the spectrum, which in this case could lead a reconstruction error that could be below a given threshold. It will be more useful then to highlight the portion of the waterfall plot where the reconstruction error is high, highlighting then the corrupted parts of the spectrum.
It is also useful to explain a reasoning that was done before the data analysis on these plots: due to the fact that looking at the power in the mainband is not so meaningful to detect the

presence of a jammer, it is expected that the higher reconstruction errors will be present in the sidebands of the channel, where usually our device tends to leak less power.
To simplify this concept, it is useful to imagine ourselves as the *watchdog* that is looking at the power of the channel in order to see if there is or not the presence of a jammer: if normally the system tends to transmit at 1 mW in the mainband and the attacker transmits random noise at 1.1 mW, no noticeable anomaly will be detected.
If instead the jammer also transmits in the sidebands of the wireless channel, where the device in the network tends to leak a very low amount of power, it will be easy to see the presence of a jammer that transmits at the same power detected in the mainband. In the case the jammer became smart and started injecting noise only in the mainband of the wireless channel, it could be highlighted that no filter in the real world is ideal, and so there will be an increasing amount of power detected in the sidebands of the channel, which according to our system can be meaningful or not.

5 Experiments

6

Conclusion and future works

References

[1] https://www.marketsandmarkets.com/Market-Reports/iot-manufacturing-market-129197408.html.

[2] https://www.statista.com/statistics/666864/iot-spending-by-vertical-worldwide/.

[3] https://www.cybersecurity360.it/nuove-minacce/ransomware/toyota-sospende-lattivita-per-un-attacco-informatico-perche-e-evento-notevole/.

[4] https://arxiv.org/abs/1511.04352.

[5] https://www.ling.upenn.edu/courses/cogs501/Gallant1990.pdf.

[6] https://web.mit.edu/course/other/i2course/www/vision_and_learning/perceptron_notes.pdf.

[7] https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=708428.

[8] https://www.sciencedirect.com/science/article/pii/S0950705121009576.

[9] https://pubs.aip.org/aip/rsi/article/65/6/1803/682910/Neural-networks-and-their-applicationsNeural.

[10] https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9451544.

Acknowledgments

A Zlatan Ibrahimović
Ad Enzo Conte