Observatoire de Paris - PSL

M1 Space Science and Technology

Internship report: Detection and characterization of galaxies using Machine Learning on a massive radio-astronomical dataset

Author:
Adrien ANTHORE

Supervisor:
David Cornu

M1 Internship

June 14, 2023

Contents

Acknowledgements

I would like to express my heartfelt gratitude to my supervisor, David Cornu, as well as Philippe Salomé, Cyril Tasse, the LERMA, and the galaxy team. Their guidance, support, and contributions made this project possible. I've learned a great deal from them, and I am truly grateful for their impact on my future work.

Abstract

The objective of the MINERVA team (Machine Learning for Radioastronomy at the Observatoire de Paris) is to enhance their previous results of sources detection and characterization on the LoTSS survey. This internship report focuses on constructing high-confidence catalogs for the LoTSS fields in order to construct later a high-quality training dataset to better train the team's deep-learning method to take into account specific properties and artifacts of the LOFAR radio-telescope. The catalogs derived from the entire LoTSS field reached satisfactory performances with an average recall of $59.9 \pm 15.5\%$ and an average precision of $61.9 \pm 7.5\%$.

1 Introduction

Radio-astronomy is experiencing a rebirth in its low-frequency domain, particularly with the development of giant interferometers such as LOFAR, ALMA, NenuFAR, and the upcoming Square Kilometer Array (SKA). These instruments produce large and highly dimensional datasets, presenting challenges for traditional methods of source detection and characterization. In parallel, Machine Learning methods have undergone algorithmic developments that bring them to a high level of maturity for these tasks.

The MINERVA project (Machine Learning for Radioastronomy at the Observatoire de Paris) is at the forefront of applying Machine Learning to radio-astronomical datasets. The project led a team that won the second edition of the SKA Science Data Challenges held in 2021(Hartley, et al., 2023), and that was focused on source detection and characterization in a large 3D synthetic cube of HI emission. For this purpose, the team developed a specialized Deep Learning detection method. Their approach not only demonstrated state-of-the-art performance for data challenge 2 but also showed great performance in data challenge 1 (Bonaldi, et al., 2020), which focused on source detection and characterization in synthetic continuum 2D images, as seen in figure 1. Now the team looks forward to using this methodology for real observed datasets in order to construct new source catalogs and explore new ways of combining information from multiple surveys. In particular, the team is interested in the survey LoTSS (the LOFAR Two-metre Sky Survey). The LoTSS DR2 survey consists of a high-resolution low-frequency survey accompanied by a catalog of approximately 4 million radio sources that has been derived by using classical detection tools (Shimwell, et al., 2022). This internship builds upon the work done in a previous M2 internship where a network trained on the SDC1 dataset was applied to the LoTSS survey, yielding satisfactory results.

The goal of the internship was to construct a complementary high-confidence training sample, specifically for the LoTSS by refining and combining source catalogs obtained with other detection methods. My internship builds on the results of the previous internship by focusing on the creation of a training set that will refine the network's training specifically for LoTSS. For that purpose, I had to train in machine learning. Then, I extensively explored the LoTSS DR2 to understand its features and potential challenges. This informed my approach to constructing a highly reliable and comprehensive training set, which is crucial for effective machine-learning training. The method used to build the training set is a classical method, different from the one used for LoTSS DR2 in order to challenge their performance.
In this report, in the first section, I will describe the principles and application of machine learning, highlighting its relevance to our research. Subsequently, I will present the procedure for source detection and characterization. In the second section, I present the results of our statistical analysis of the LoTSS data and the quality of the training set. Finally, in the last section, I will discuss the result and provide further explanations or hypotheses of our observation.

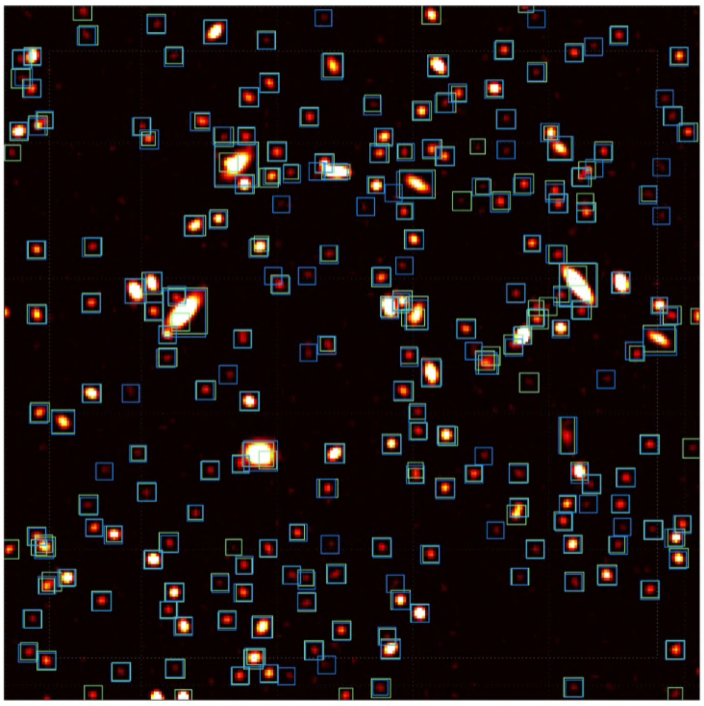


Figure 1: Source detention in a SDC1 subfield with MINERVA team's method. The predicted boxes are in blue, the targets in green. - Credit: David Cornu.

2 Method

In order to build a training dataset for a machine-learning algorithm made for object detection and characterization, it was necessary for me to understand how machine learning methods work, especially supervised methods that use dense neural networks, by following David Cornu's class on machine learning[1]. This understanding allowed me to identify and assimilate the strengths and weaknesses of Convolutional Neural Networks, as employed by the MINERVA team for object detection.. In studying the team's method, I had to specify the workflow to adapt a detection method from small subfields to its implementation in a large-scale survey, which involves overlapping observation regions. This detailed knowledge of the method and the team's tools is a necessary condition for the construction of a dataset that respects the prerequisites of purity and diversity to hope to serve as a training base.

2.1 Introduction to Machine Learning

The first question we may ask to start this method section is: what is machine learning ? Defining ML is not easy because it involves defining what "learning" means, which is not straightforward either. One can define ML as "have a computer extract statistical information from a dataset and adapt to it through an iterative process" (Cornu, 2020). While no single and exclusive definition exists, this view encompasses a comprehensive understanding of ML.

There are several methods of ML, and their use depends on the application. Typically, those methods are categorized into two primary families: supervised and unsupervised.

- The supervised approach use a training dataset that contains the expected output for each example. The method attempts to reproduce the target. It learns by comparing its current

[1]https://github.com/Deyht/ML_OSAE_M2

output with the target and correcting itself based on this comparison. After training, such algorithms are able to generalize to objects that are similar to those that were used for training. This is the most common type of algorithms because they are often simpler than the ones in other categories, and because it is easier to assess their prediction performance. They usually have a broad range of applications. (Cornu, 2020)

- The unsupervised approach use a training dataset with no information on the expected output. The algorithm will then attempt to create categories in the dataset by itself based on some predefined proximity estimator. Most of these methods are clustering ones, that can either be used for classification or dimensionality reduction for instance. Despite their reduced application range compared to supervised learning, they are commonly used as well. (Cornu, 2020)

This internship is focused on supervised learning, and specifically on convolutional neural networks (CNN). The framework used is CIANNA[2] (Convolutional Interactive Artificial Neural Networks by/for Astrophysicists) which is a general-purpose deep learning framework developed specifically for astronomical datasets and applications by David Cornu.

2.2 Neural Network and deep learning

Now that I have defined the framework of my internship, I will define the basic tools of ML.

2.2.1 Simple neuron model and algorithm

In order to provide a comprehensive understanding of deep learning and CNN, it is essential to introduce the concepts of artificial neurons.

Artificial neural networks are the most common ML familly of method in supervised learning. It is inspired by the mathematical model of a biological neuron from McCulloch and Pitts (McCulloch, W. S. and Pitts, W., 1943). The biological neuron is an elementary brick of the brain. It can be connected to multiple others in order to receive or transmit signals. One biological neuron receives and sums electrochemical input signals, and if this total signal is sufficient, it sends a new signal to transfer information toward other neurons.

The artificial neuron mimic this process. It consists of an input vector X_i with the same dimension m than the data, the dimensions are called features. Each element within the input vector represents a specific feature of the object being processed. The artificial neuron operates by considering each feature individually. Each feature, is assigned a weight ω_i. The features are then combined with their corresponding weights, allowing the neuron to evaluate the importance of each feature in the overall computation. The product of the weights with their corresponding neuron are then combined in a sum function h :

$$h = \sum_{i=1}^{m} X_i \omega_i \tag{1}$$

In practice, we add an extra feature in the input vector X_i called the bias node with a fixed value but with an associate weight that behaves like the other ones. This extra input allows to define a non-zero orgin to the linear separation of the neuron. Equation (1) can be rewrite as follow :

$$h = \sum_{i=0}^{m} X_i \omega_i = b\omega_0 + \sum_{i=1}^{m} X_i \omega_i \tag{2}$$

[2]https://github.com/Deyht/CIANNA

where b is the bias input and ω_0 its associated weight.
Then, the result of the weighted sum pass through an activation function $g(h)$ that defines the state at the output of the neuron. For instance, we emulate one of the simplified properties of a biological neuron, which is its ability to be in an "active" state (state 1) or "inactive" state (state 0) based on the total signal received. This can be done with a step function :

$$a = g(h) = \begin{cases} 1 & \text{if } h > \theta; \\ 0 & \text{if } h \leq \theta. \end{cases} \tag{3}$$

where θ is a threshold value. Depending on the value of h, the network can remain in a "0" state or can be activated to a "1" state. The action of computing the activation of a neuron, or more generally a network, for a given input vector is called a forward step.

The training process of such a neuron consists in finding an appropriate combination of weight values that minimizes a specific error function. In supervised ML, this process uses a training dataset composed of examples with a known solution named target t. The error is computed from the comparison between the activation and the target. It is used to correct the weights, this step is called an update. Repeating the forward and update steps for all input vectors of the training dataset, the weights gradually converge toward suitable values. We can call this phase a learning phase. An "epoch" refers to this phase conducted on the entire dataset, and multiple epochs are required for the network to converge toward a optimum and stable solution. In practice, the adjustment of weights is determined by the derivative of the selected error function and is directly related to the relative input associated with each weight. In the case of the binary neuron with the activation described by equation (3) with a square error: $E = 0.5 \times (a-t)^2$, the correction of the weights is implemented as follows:

$$\omega_i \leftarrow \omega_i - \eta(a-t)X_i \tag{4}$$

where η is a learning rate defined according to the problem to be solved.
For the very first iteration, before the first update, the weights are initialized with small random values. It is important that those values are not the same for every weights, are non null, and not selected by the user directly. For instance, a normal distribution center around 0 with a rather small variance s usually considered as a good initialization for the weights. Note that the weights are independent from the input vector, which implies that regardless of the input, the same features will consistently be assigned the same weight.

2.2.2 Perceptron and multilayer perceptron

In the previous section, we saw that a single neuron performs only linear separations. In order to treat more complex problems, more neurons are necessary. One straightforward approach for combining neurons is adding them within the same layer. Each neuron is connected the same way as the single neuron to the input layer and has the same behavior. However, each neuron operates independently. In this case, the weighted sum and the activation equations for each neuron j becomes:

$$h_j = b\omega_{0j} + \sum_{i=1}^{m} X_i \omega_{ij} \tag{5}$$

and

$$a_j = g(h_j) = \begin{cases} 1 & \text{if } h_j > \theta; \\ 0 & \text{if } h_j \leq \theta. \end{cases} \tag{6}$$

and thus the update of each weight that connects input i to neuron j is :

$$\omega_{ij} \leftarrow \omega_{ij} - \eta(a_j - t_j)X_i \tag{7}$$

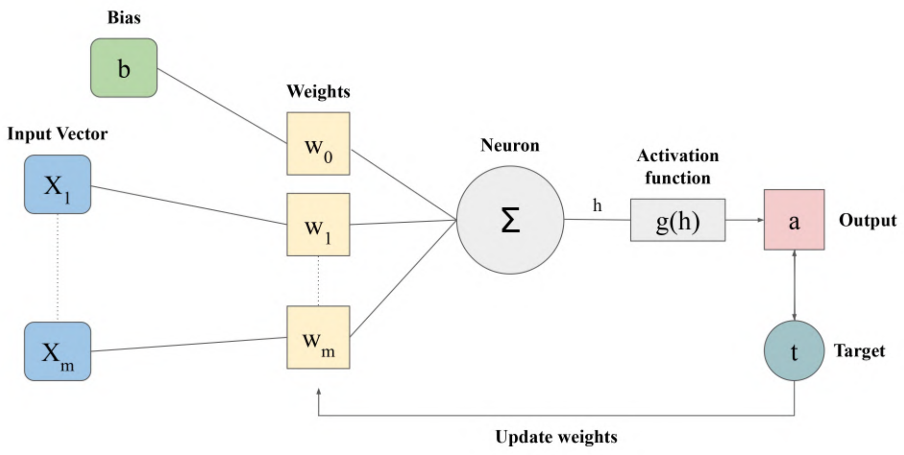


Figure 2: Schematic view of a binary neuron. $X_{[1,\ldots,m]}$ are the input features for one object in the training dataset and b is the bias, $w_{[1,\ldots,m]}$ are the weights associated with each feature and w_0 is the weight of the bias. Σ represents the sum function and h its results. The result of the sum function then goes into the activation function $g(h)$ and a is the final neuron activation state that is compared to the target t of the current object.

Networks build with this formalism are called the perceptron algorithm.
Besides adding neurons within the same layer, a new layer with a new set of weights can be added behind the previous one. It takes as input the result of the activation of the neurons from the previous layer. The neurons within the additional layer behave similarly to those in the initial layer. By repeating this procedure, multiple layers are added, constructing a "deep" network. In this case, the first layer is called the input layer, the last layer is called the output layer, and the other layers between them are called "hidden" layers. Although the different layers have the same behavior, they are not built in the same way. The input and output layers are mostly constrained by the problem to solve. In contrast, the hidden layers represent the computational strength of the network and must be adapted to the task's difficulty. This type of network architecture is called a Multi-Layer Perceptron (MLP).

In addition to adding neurons or layers, one can use continuous and differentiable activation functions. For instance, the sigmoid function :

$$g(h) = \frac{1}{1+\exp(-\beta h)} \tag{8}$$

where β is a positive hyperparameter that defines the steepness of the curve. This function is S-shaped with results in $]0;1[$,illustrated in the figure 8, and has a simple derivative form. This activation function eases the regression within the Perceptron network, enabling the representation of each continuous variable using a single neuron.

The multilayer architecture of this network enables non-linear combination of the sigmoid functions, with each layer increasing the complexity of the achievable generalization. Their combination allows the network to represent any function, making it a "Universal Function Approximator" (Cybenko, G.).

The addition of new layers in the network doesn't change much the forward processes, but the update of the weights described by equation (7) is no longer appropriate since the targets are only available for the output layer. The method used to update the weights of an MLP is the "Backpropagation" algorithm (Rumelhart, et al., 1986). It allows computing an error gradient descent starting from the output layer and propagates through the network. The gradient depends on the error function, which is often the simple sum-of-squares error :

$$E(a,t) = \frac{1}{2}\sum_{k=1}^{N}(a_k - t_k)^2 \tag{9}$$

where k runs through the number N of output neurons, a_k is the activation of the k-th output neuron and t_k the corresponding target. The weight update for a given layer l is computed as follows :

$$\omega_{ij} \leftarrow \omega_{ij} - \eta \frac{\partial E}{\partial \omega_{ij}} \tag{10}$$

where η is the learning rate and the gradient $\frac{\partial E}{\partial \omega_{ij}}$ can be expanded as :

$$\frac{\partial E}{\partial \omega_{ij}} = \delta_l(j)\frac{\partial h_j}{\partial \omega_{ij}} \quad \text{with} \quad \delta_l(j) \equiv \frac{\partial E}{\partial h_j} = \frac{\partial E}{\partial a_j}\frac{\partial a_j}{\partial h_j} = \frac{\partial a_j}{\partial h_j}\sum_j \omega_{kj}\delta_j(k) \tag{11}$$

In these equations, the i refers to the layer's input dimensions, and the j refers to the layer's neurons. These equations are the same for each layer. δ_l is a local error term that can be defined for each layer so that, for a hidden layer l, the gradient in equation (11) can be substituted by the product of the next layer weight matrix and the next layer local error δ_{l+1} with k that runs through the number of neurons of the next layer. These terms can be simplified by considering the previous error function and a sigmoid activation for all neurons:

$$\frac{\partial h_j}{\partial \omega_{ij}} = a_i \tag{12}$$

the activation of the current layer,

$$\frac{\partial E}{\partial a_j} = (a_j - tj) \tag{13}$$

the derivative of the error to replace δ_l at the output layer,

$$\frac{\partial a_j}{\partial h_j} = \beta a_j(1 - a_j) \tag{14}$$

the derivative of the sigmoid activation.

2.3 Neural Networks for images

Now that the basic ML tools are established, I will explain what their limitations are in object detection and characterization and how the MINERVA team proceeds.

2.3.1 Convolutional Neural Network (CNN)

As part of the internship, neural networks are used to detect objects in images. However, when it comes to handling images, classical MLPs are inefficient. The typical tasks to perform on images are usually characterized by a high degree of invariance. The content of an image usually preserves its nature when applying various transformations like translation, rotation, color shift, etc. A classical MLP could take an image as input by considering all pixels as independent input features. Still, it would be strongly inefficient even for the simple task of searching for the

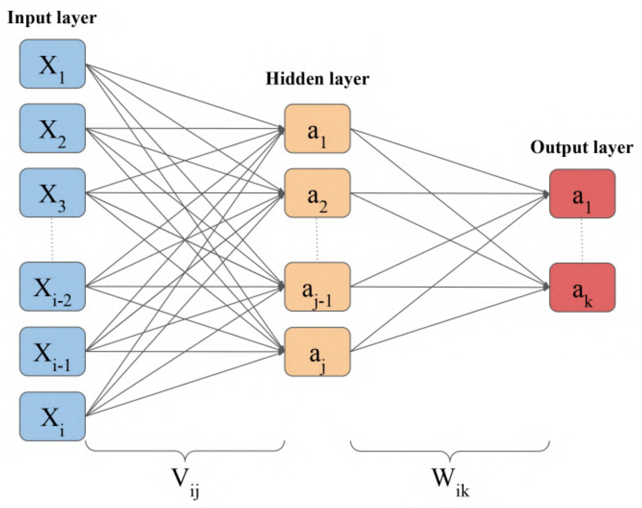


Figure 3: Schematic view of a simple "deep" neural network with only one hidden layer. The blue cells are input dimensions. The beige cells are the hidden layer dimensions. And the red cells are the output dimensions. $X_{[1,...,i]}$ are the dimension for the input vector, $a_{[1,...,j]}$ are the activation for the hidden neurons, $a_{[1,...,k]}$ are the activation of the output neurons, while V_{ij} and W_{jk} represent the weights matrices between the input and hidden layers, and between the hidden layer and output layers, respectively.

presence of a repeating pattern at different locations. The solution found to handle images is to build another type of network, a convolutional neural network.

This type of network is based on the convolution operation that consists of the application of a filter to an image through a decomposition in sub-regions, illustrated in the figure 4. A filter is a set of numerical values with a predefined spatial size. The values of the filter are multiplied element-wise to a subset of pixels, the results are then summed to obtain a single value. The filter is applied on the images at regular intervals on the input with a shift in pixels between each application called stride S. For a 2D image, the convolution uses a 2D filter that slides over the image along one axis, along the lines for instance, with a shift of S pixels between each applications. When the end of the line is reached, the operation is repeated for another line according to the stride, so that the filter is applied regularly in both dimensions. A side effect of the operation is to reduce the spatial size between the input and the output. In the case where it is preferable to preserve the spatial size, it is possible to add a zero-padding P related to the size of the filter around the input image. It results in the following relation between input and output dimensions :

$$w_{out} = \frac{w_{in}-f_s+2P}{S} + 1 \quad h_{out} = \frac{h_{in}-f_s+2P}{S} + 1 \tag{15}$$

where w and h are the input and output widths and heights respectively, f_s is the filter size, P is the padding, and S is the stride. This can be generalized for multi-channel input images. In that case, the filters are usually considered to have a supplementary dimension that spans over all the input channels at once at each location. For example, with a classical RGB image, the filter has a supplementary dimension of size three.

From this formalism, it is possible to build a convolutional layer that is composed of several filters with the same size, stride, and padding, so they are used uniformly over the input image. This layer is analogous to the perceptron algorithm, where the flattened sub-regions of the input image can be considered as the input vector X_i, and the flattened filters can be considered as the weights W_i. But this time, the same weights are applied to several subparts of input space in order to scan for a given pattern at all possible locations. Just like with the perception, to

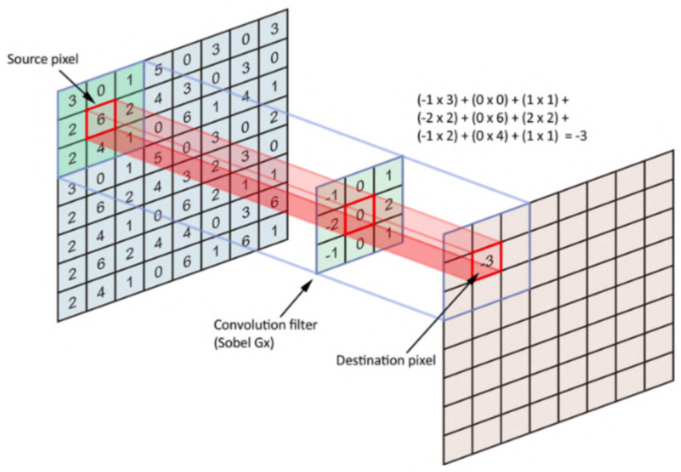

Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
(Sobel Gx)
Destination pixel

Figure 4: Schematic representation of a convolution operation. - Source : https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/

be considered as neuron response, the weighted sum between the filter and a given region must go through an activation function to add some non-linearity. For each filter, the result of this activation at all the possible locations in the input image forms a spatially arranged activation map. Consequently, the output of a convolutional layer is a set of 2D activation maps with the number of activation maps equal to the number of filters used in the layer, as illustrated in figure ??.

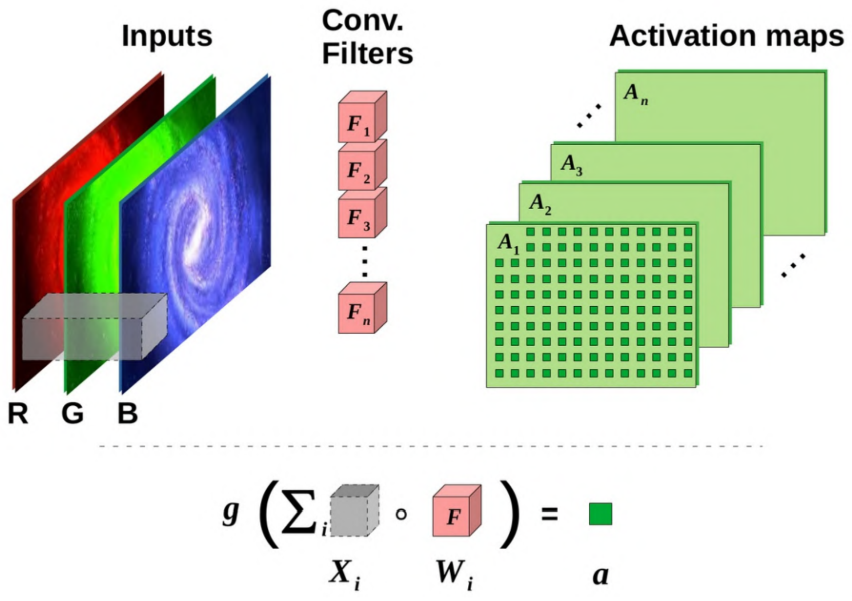


Figure 5: Schematic representation of a convolutional layer. The input consists of a three-channel image (RGB) with n filters. The output consists of n activation maps. - Credit: David Cornu

A single convolutional layer can identify many patterns if provided with many filters. However, it remains limited to patterns of the size of the filter and is often unable to identify complex non-linear patterns efficiently (Lecun and Bengio, 1995). Therefore, convolutional layers can be repeated, each layer taking the activation maps of the previous layer as its own multi-channel input image. Increasing the number of convolutional layers allows the network to identify more complex patterns efficiently. Moreover, those layers are often combined with what is called pooling layers, which reduces the spatial dimensionality of the output images or activation maps. Max-Pooling is the most commonly used pooling operation. With a pooling size of P_0, it divides each input image channel into non-overlapping sub-regions of size $P_0 \times P_0$ pixels. The resulting output consists of the maximum value from each of these sub-regions and preserves the global organization of the activation maps. This operation aims to reduce dimensionality by selecting the dominant pixels in the input image. By carefully alternating convolution and pooling layers, it is usually possible to significantly accelerate the learning process without significantly sacrificing the predictive capability of a CNN network. Most of the time, a convolutional network end with a few more classical MLP layers by flattening all the pixels of the last convolutional layer feature maps as independent features of the first dense layer. This structure can then be used to identify a few classes, perform regression, etc, like the classical MLP but by efficiently handling input images.

Although any activation function can be used for such deep convolutional neural networks, some of them are more efficient and avoid issues that may stop the learning of the network, such as the gradient vanishing problem. The most used activation function in such a network is the Rectified Linear Unit activation or ReLU (Nair and Hinton, 2010). This function is simply linear for any input value above zero and equal to zero if the input is negative :

$$a_j = g(h_j) = \begin{cases} h_j & \text{if } h_j \geq 0 \\ 0 & \text{if } h_j < 0 \end{cases} \tag{16}$$

This function has several advantages: it preserves a simple form of non-linearity with two states, it is scale-invariant in its linear regime, it is easy to compute, and it has a constant derivative of 1 in its linear regime, ensuring the full propagation of the error gradient, illustrated in figure 16.

While it is possible to use only pre-defined filters, the true objective of such deep network architecture is to learn these filters automatically. Before starting the training process, the filters are initialized to small random values, just like the weights in the MLP. These weights are then updated during the training process based on the output error and using backpropagation of the error gradient through the whole network. Therefore, error propagation in the convolutional structure plays a crucial role in the learning process.
For max-pooling layers, the local error term has to be propagated to the layer input by associating the error to the input location that was the maximum value of each sub-region. All other elements involved have their error values set to zero.
For convolutional layers, the error is propagated using a transposed convolution operation (Dumoulin and Visin, 2018). Without entering into the details, this operation spread the error value at a given output location to all the input pixels that contributed to corresponding neuron activation. This operation is the equivalent of the equation (11) for each sub-region of the considered operation input image. After the error propagation, each convolutional layer can update its own filters by summing the correction coming from all the regions they where applied to.

2.3.2 You Only Look Once (YOLO)

The deep learning method developed by the MINERVA team is inspired by the YOLO (You Only Look Once) architecture. This method belongs to a family of regression-based object detectors. The network used was built by David Cornu with a combination of features and capabilities that are a mixture of the YOLO v1 (Redmon et al., 2016), v2 (Redmon and Farhadi, 2016) and v3 (Redmon and Farhadi, 2018) architecture, and that was implemented inside the CIANNA framework. Regression-based methods are known for speed as they are widely used for real-time object detection, but they also offer a more straightforward formalism to object detection in comparison with region-based or segmentation methods.

A classical YOLO network consists of a fully convolutional network in which a YOLO layer is added at the end. The role of the YOLO layer is to encode the network output as a grid that maps the input image of the network, where each grid element contains the properties of several possible objects centered on this specific grid element. Each grid cell predicts several possible bounding boxes from a set of priors and a detection score called "objectness". The information about each possible box is stored consecutively in a vector corresponding to the grid element. The detection of an object is done on each grid element, where the network will try to overlay the boxes with the targets as much as possible. At each training step, the YOLO layer will associate the actually predicted boxes with a list of target boxes. For each target, the closest prediction is found by searching for the highest Intersection over Union (IoU) between the target and any prediction. The current best prediction receives an error that pushes the prediction in the direction of the target, both for the box center and for the box size. Regarding the prediction score, the associated predictions receive an error that increases their score, while all predictions that are not associated with any target are pushed to lower their score.

At inference time, all the possible boxes in each grid element are always predicted but they can be filtered based on their objectness score. To remove possible multiple detections in the remaining high score predictions, it is necessary to apply what is called a Non-Maximum Suppression (NMS), in order to keep only the best-predicted box for each object in the image. The box with the highest probability is selected and placed in a closed list, and then all the boxes that overlap with it (based on an IoU threshold) are removed. This process is repeated until there is no more box in the open list of predicted boxes.

In the context of the internship, the detection of the radio sources predicts both the central position and box size for each object but also a list of complementary parameters (flux, position angle) that are predicted by the network as a regression thanks to the specific implementation inside the CIANNA framework.

2.4 Workflow

Now that we have described the source detection and characterization network used for the internship, as well as its outputs, we can define a procedure to apply it to the complete LoTSS survey. The workflow consists of 2 main loops. The first loop is an iteration of every mosaic fits file to be treated. The second loop iterates over "patches," which consist of sub-spaces having a specific number of pixels and a slight overlap between them.

Due to the multiple divisions within a larger sky image, it becomes necessary to implement 3 non-maximum suppression :

1. The first NMS, or intra-patch NMS, suppress overlapping within a patch (for YOLO detection).
2. The second NMS, or inter-patch NMS, suppress overlapping within the overlaps between patches.
3. The third NMS, or inter-fits NMS, suppress overlapping within the overlaps between mosaics.

The NMS proceed the same way. It involves iterating over a list of bounding boxes, selecting and saving the box with the highest objectiveness, and removing it from the list. Then any bounding boxes that have an IoU greater than the threshold are removed from the list. The procedure repeats until the list is empty.

In the detection, the bright sources produce a lot of artifacts and, therefore, a considerable amount of false detection. Our approach involves implementing rejection radius criteria centered around bright sources as described in the result section. The rejection radius increases as the source's flux becomes higher. Sources located within this radius, with a flux below a determined threshold, are suppressed.

The workflow can be described as follow :

- Enter loop 1
  - The current mosaic is decomposed into patches
  - Enter loop 2
    - Perform detection on the current patch using YOLO
    - Perform the first NMS
  - Exit loop 2
  - Perform the second NMS
- Exit loop 1
- Perform the third NMS
- Perform the box rejection

3 Results

3.1 Summary statistics of the LoTSS data

In this subsection, I present the results of exploring the source catalog obtained from Shimwell, et al. (2022). The LoTSS survey consists of high-resolution 120-168 MHz images covering 27% of the northern sky. From this data, a catalog of 4,396,228 radio sources candidates has been derived by using the Python Blob Detector and Source Finder (PyBDSF)[3].
However, our main focus lies on source properties within the catalog, namely the coordinates (RA, DEC J2000), the major and minor axis, the position angle, and the integrated flux of each source. In this catalog, 92% of the sources are considered to be unresolved.

The first catalog property I studied was the distribution of the source flux. The flux values in the catalog exhibit a wide range, spanning from 10^{-2} mJy to 10^{5} mJy. Most sources have a rather low flux, with half of them having a flux lower than 0.961 mJy. The distribution of the flux is shown in figure 6. We observe three different patterns in the distribution, a roughly bell-shaped pattern (pattern 1), a repeated pattern where the number of sources suddenly increases, followed by a decrease in the following bins that tend to follow the global diminution of the distribution (pattern 2), and a jagged pattern repeated before the other patterns where the number of sources successively increases and decrease (pattern 3). Those patterns will be discussed later in the section 4.

The second diagnostic I studied was the possible link between the size of the sources and their flux. We decided to represent a 2D histogram of the integrated flux of the sources as a function

[3]https://pybdsf.readthedocs.io/en/latest/

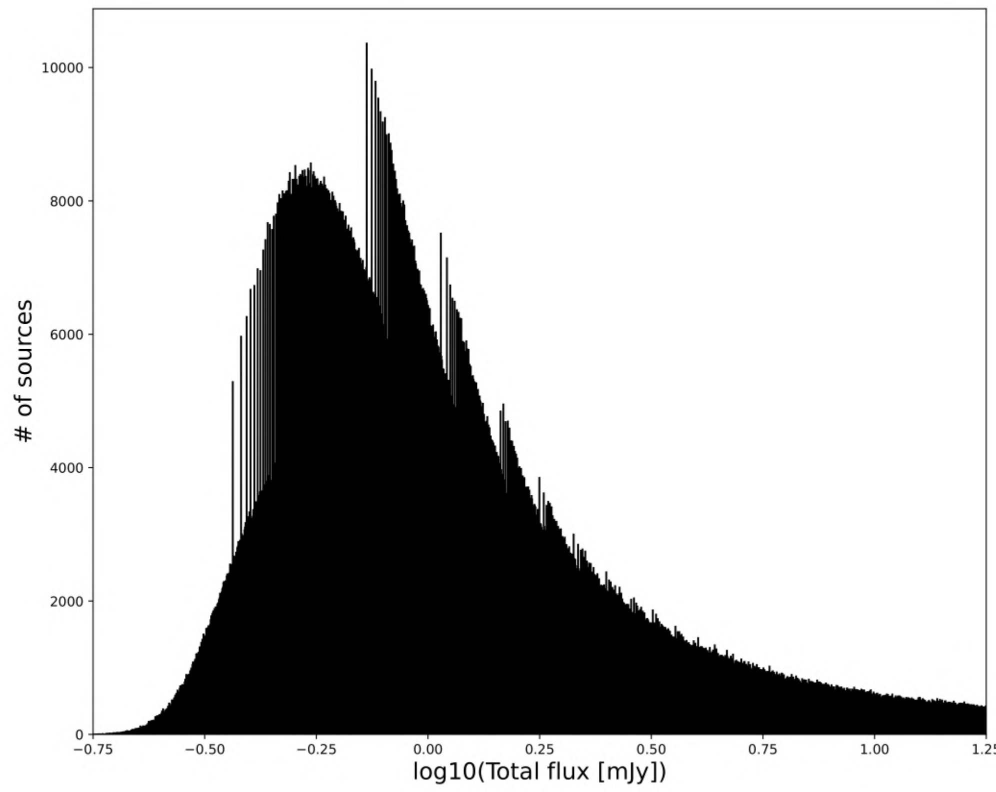


Figure 6: Histogram of $\log_{10}$ flux distribution of sources. The histogram is constructed using 5000 bins linearly spaced over the full range of flux in the catalog. The window has been constrained from -0.75 to 1.25 to focus on the predominant flux trends observed within the catalog.

of their major axis seen in figure 7. Most of the sources in the catalog are rather small with a rather low flux, we see in the figure the highest density region in the histogram is for flux between 0.4 mJy and 2 mJy, and for the major axis between 6 arcsecs and 10 arcsecs. We notice that the greater sources tend to have a higher integrated flux, while sources with a major axis inferior to 6 arcsecs seem not to have flux greater than 10 mJy.

Eventually I studied the distribution of sources through a 2D histogram of the minor axis as a function of the major axis. We looked at the axis convolved (left) and deconvolved (right) with LOFAR's beam. As expected, most of the targets have a small size and are therefore unresolved. We see on the graph, for the convolved and deconvolved axes, two different regimes that we will discuss in section 4.

3.2 Astrodendro method

Since our results will be compared to this catalog generated by PyBDSF, it is essential for us to construct the purest and most comprehensive catalog possible using an alternative method. The chosen method is the Astrodendro package in Python[4]. The algorithm works as follows: starting

[4]https://dendrograms.readthedocs.io/en/stable/index.html

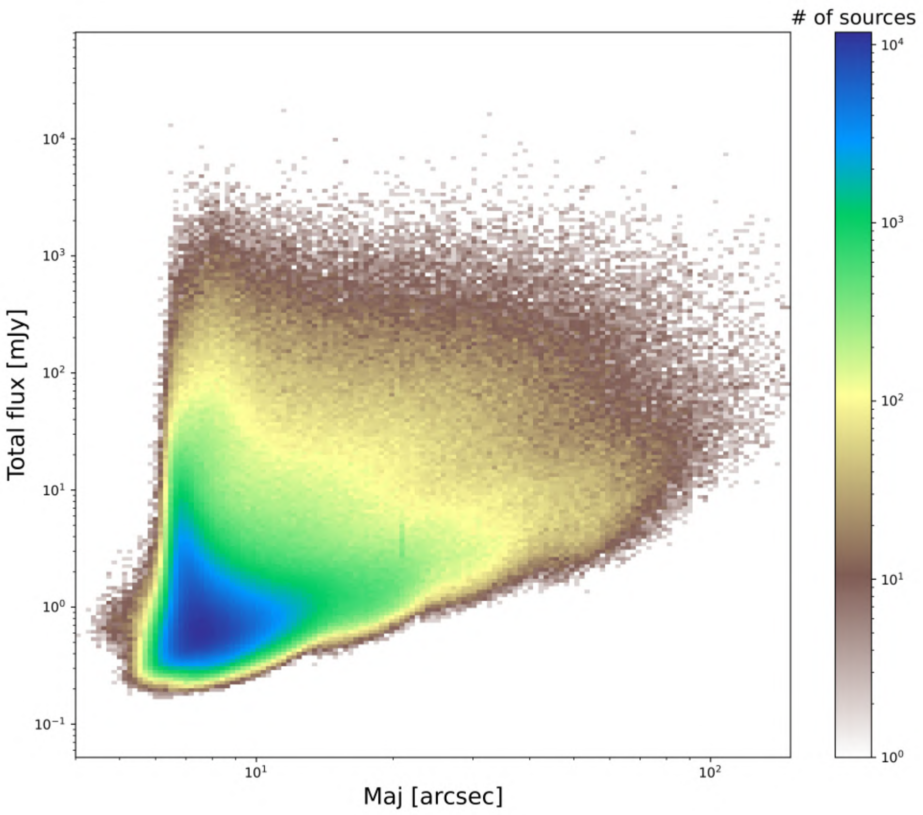


Figure 7: 2D histogram of the flux of the sources as a function of their major axis. Maj refers to the major axis, and Total flux refers to the total integrated flux of the source. The histogram was constructed using 200 bins logarithmically spread over the full range of both dimensions.

from the highest flux value, it iterates downwards with a selected step size called "*min_delta*" until reaching a selected baseline value called "*min_value*", illustrated in the figure 9. The structures are constructed based on the detected local maxima. We can also choose the minimum amount of pixels used to compute a structure through the parameter "*min_npix*" to prevent small islets of noise to be considered as sources. The leaves of the structure represent the local peaks, while the trunks correspond to clusters of adjacent leaves. From those structures, a built-in method creates a catalog of structures containing all the source candidates parameters (position, size, flux, position angle).

3.3 Parameter Selection

In the mosaics produced by LOFAR, all detection method will inevitably make false detections. This is due to the presence of numerous artifacts in the field with enough local contrast to be identified as sources. For example, in figure 18 we can see the the artifacts we most likely met while working with LoTSS data. Therefore it is common to search for optimized parameters of the method in order to minimize false detection while preserving a high detection rate. For Astrodendro we can reduce these false detections through careful selection of *min_npix*, *min_value* and *min_delta*. However, making the method too intolerant will decrease the recall of the detection, which means that the number of detected sources will be underestimated. Therefore,

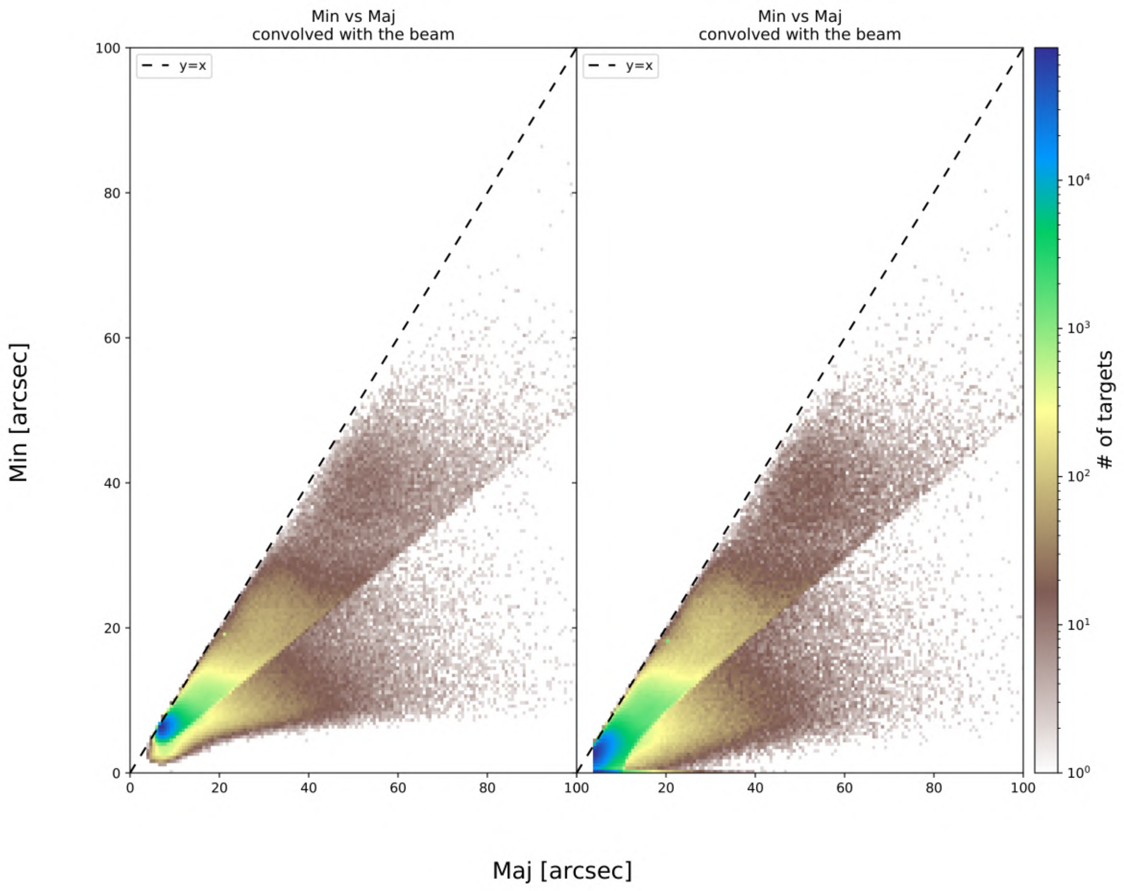


Figure 8: 2D histogram of the major axis versus the minor axis of the sources for both convolved and deconvolved axis. Min refers to the minor axis, and Maj refers to the major axis. The histograms were constructed using 1000 bins linearly spread over the full range of both dimensions. The slope x=y is over-plotted on the graph, it represents an asymptote in the distribution.

we have made the decision to establish criteria to eliminate as many false detections as possible from the catalog generated by Astrodendro and keep a reasonable level of purity. The first step was to select the parameters based on the observed field. Then we had to find additional corrections to the produced catalogs.

3.3.1 *min_npix*

The parameter *min_npix* is the minimum amount of pixels used to compute a structure, which prevents the method from computing structure for too small features that are too small. We chose it accordingly to the pixel size and the beam of the instrument:

$$min\_npix = 2 \times \frac{\text{beam\_size}}{\text{pix\_size}} \tag{17}$$

where beam_size is the size of the beam, and pix_size is the size of one pixel in the sky. With the LoTSS data, we have beam_size = 6 arcsecs, pix_size = 1.5 arcsecs, so we obtain *min_npix* = 8 pixels.

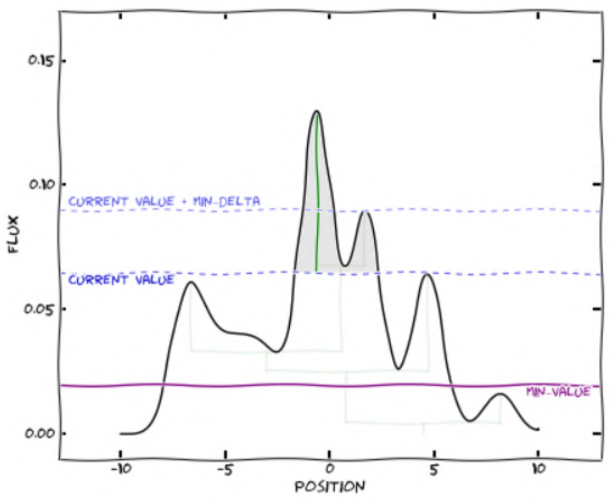


Figure 9: Illustration of the Astrodendro algorithm. - Source : https://dendrograms.readthedocs.io/en/stable/algorithm.html

3.3.2 *min_delta*

For the the *min_delta*, the step of each iteration, we chose the value of this parameters based on the pixel value's distribution for a specific field. This approach allowed us to fine-tune the algorithm's behavior based on the field studied. We create the pixel flux value histogram, from this distribution, we perform a Gaussian fitting to obtain the mean and standard deviation. The value of 3 times the standard deviation determines the *min_delta*.

3.3.3 *min_value*

For the *min_value*, the minimum value of the pixel taken into account, its value is also chose based on the pixel value's distribution for a specific field. The method used to determine its value relies on the previously defined histogram as well as the mean fitted by the Gaussian. From the histogram, we compute the relative difference on both sides of the Gaussian fitted mean compared to the right side of the histogram. Among the obtained relative differences, we select the first positive pixel value for which the relative difference exceeds 5% as the *min_value*. The method is illustrated in the figure 10 with the mosaic P7Hetdex11 used as example. The relative difference between both part is computed such that:

$$\eta = \frac{\text{left} - \text{right}}{\text{left}} \tag{18}$$

The left part of the histogram is typically pixels that contains noise where the right part may contains signal at some point. We are looking to the pixel value where the right part of the histogram become great enough in front of the flipped left side.

3.3.4 Purity enhancement

With the previous parameters, Astrodendro is capable of producing a source candidate catalog. However, the produced catalog contains a considerable amount of false detections due to the variety artifacts present in the LoTSS data we see in figure 18. We see in the right image that the artifacts are linked with the presence of bright sources in the field. They have an elongated shape

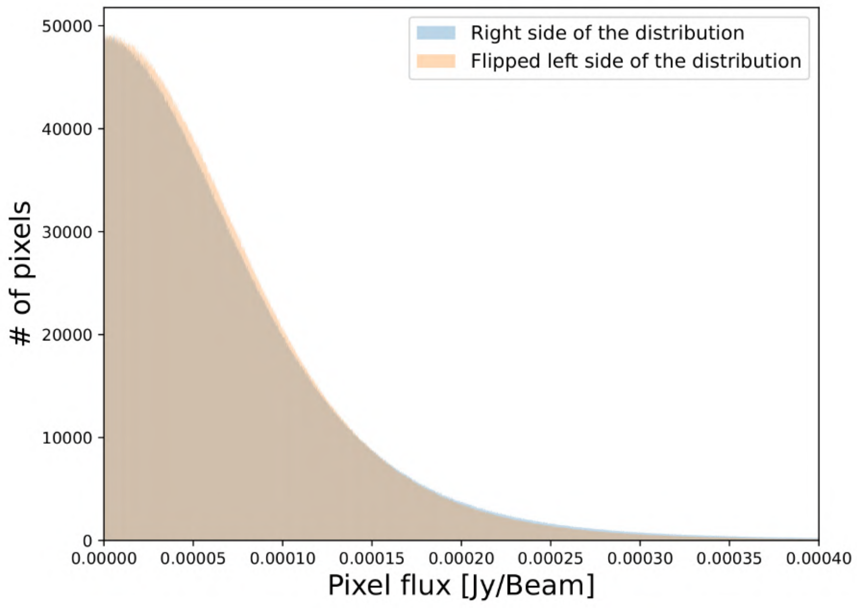


Figure 10: Histogram of the pixel flux value flipped and over plot on the right part. The mosaic used for this exemple is P7Hetdex11. The left part is in light orange. The right part is in light blue. The brown area is where the two side are superimposed

and are expended in the radial direction of the bright sources. By analyzing catalogs produced without corrections, we were able to construct a criteria that significantly increase the catalog purity.

The first refinement I proposed is to remove all the sources in the catalog that have no spatial extension (null major axis). Such detections are irrelevant and may sometimes occur.
The second refinement I proposed is to remove the sources that have a too high eccentricity. As we can see in figure 18, the artifacts present in LoTSS data take on the shape of thin lines. Most of the time, Astrodendro characterizes them as very elongated ellipses. I have chosen to set the eccentricity threshold at 0.9.
Despite the second correction, there is still a non-negligible amount of artifacts detected around bright sources, as illustrated in figure 18. These are the artifacts that Astrodendro doesn't characterize as elongated. To remove such detections, we use a rejection radius around bright sources, which is function of their flux:

$$4 \times e^{\log_{10}(flux)} + 2 \tag{19}$$

this function was empirically derived from our observations on the global flux's distribution in the catalog produced by Astrodendro as well as the typical radius within which we can find the artifacts around bright sources. A detection is removed if it is contained inside this radius unless the detection has a flux greater than 10 mJy. The rejection radius as a function of the flux of the source is represented in figure 19 in the Annex section. This radius is illustrated on an example in figure 11.

3.4 Astrodendro results

In order to evaluate the performance of the detection by Astrodendro, we will compare the catalogs obtained with the source catalog of the DR2 obtained by PyBDSF. For this, we do a cross-match by considering the position of the sources, where we define a match between two

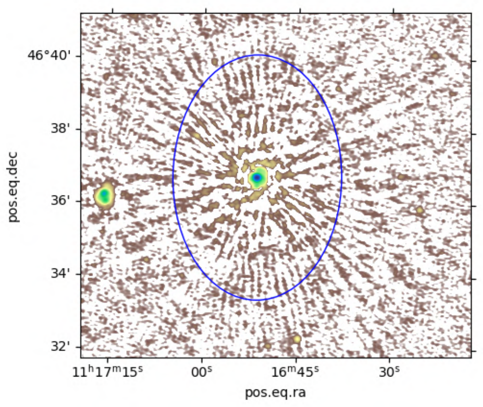


Figure 11: Rejection radius illustrated with a blue circle around a bright source.

sources if their separation is less than 12 arcseconds. Once the cross-match is done, we can define a recall that represents a detection rate and a precision that represents the purity of the prediction by taking the PyBDSF catalog as a reference. The recall is defined as:

$$\text{recall} = \frac{\text{N}_{\text{match}}}{\text{N}_{\text{PyBDSF}}} \tag{20}$$

where N_{match} is the number of cross-matched sources, and N_{PyBDSF} is the number of sources detected by PyBDSF in the field studied. The precision is defined as:

$$\text{precision} = \frac{\text{N}_{\text{match}}}{\text{N}_{\text{dendro}}} \tag{21}$$

where N_{dendro} is the number of sources detected by Astrodendro in the field.
We also look at the relative difference between the source parameters derived by Astrodendro and PyBDSF.

Previously, we mentioned that Astrodendro could compute a catalog from two different structures: the leaves and the trunks. In the table 1, we have the comparison between the catalog formed by the two possible structures in Astrodendro, tested on the mosaic P7Hetdex11: This

	leaves	trunks
Recall	81%	72%
Precision	63%	57%
Flux average relative difference	0.48	0.54
Major axis average relative difference	-0.39	-0.37
Minor axis average relative difference	-0.44	-0.42

Table 1: Comparison of the catalog produced by Astrodendro using the leaves or the trunk structure in the mosaic P7Hetdex11.

table gives us an overview of the performance differences between the two types of structures on our example field. We chose to use the leaves to build our catalog, this choice will be discussed in the section 4.

Then, I used the mosaic list available in the LoTSS DR2. The quality of these mosaics varies significantly due to the background noise, the amount of artifacts, and the presence of holes in the mosaics. There is also a considerable amount of overlaps. In order to have an estimate of the quality of the Astrodendro method, I decided to apply it on each mosaic independently, by finding the appropriate parameters as mentioned in section 3.3. I chose to show the same metrics metrics as previously (recall, precision, and relative errors of source parameters) averaged on each independent fields, with a standard deviation taken at 1σ levels taken as uncertainty. The results are given in the following table: I will discuss in the section 4 what should be changed

Recall	$59.9 \pm 15.5\%$
Precision	$61.9 \pm 7.5\%$
Flux relative difference	0.10 ± 0.17
Major axis relative difference	-0.46 ± 0.21
Minor axis relative difference	-0.54 ± 0.14

Table 2: Quality table of the training set tested on the full LoTSS data, averaged over all individual fields. The sources cataloged by Astrodendro comes from the leaves of the dendrograms computed.

to make a unified detection.

4 Discussion

In this section, the result from the previous section will be further discussed and put in perspective with the internship goals. I will provide details about what we have observed and possible explanations.

4.1 Summary statistics of the LoTSS data

The first results I will discuss are those from the statistical analysis of the LoTSS data.

Regarding the histogram in figure 6, we can further analyze the patterns described in the previous section. Especially the patterns described in the previous section. Pattern 1, the bell shape in the distribution, was actually the expected distribution of the flux. The two other patterns are symptomatic of distinct behaviors within the dataset. We are unsure of the origin of pattern 2 (the sudden increase) we suppose that it comes from the threshold effect on the flux characterization, but we did not find a solid explanation yet. Concerning pattern 3 (the jagged pattern) we suppose it might come from false detections in the catalog. To confirm that hypothesis, we looked at the histogram of the catalog Radio Galaxy Zoo (RGZ), which is the LoTSS catalog pruned of questionable detections for which sources properties were further refined by different methods, including human inspection when necessary. I produced an equivalent histogram of figure 6 in the figure **??**. The pattern 2 and 3 totally disappeared from the histogram, leaving only a bell-shaped histogram. This improvement may come from the refinement of the source parameters, but also from the removal of sources in a specific region of the LoTSS field due to the poor quality of the mosaics in that area.

Concerning the distribution of the integrated flux of the sources as a function of their major axis in figure 7, it is a good diagnostic for selecting certain hyperparameters or architecture properties in the YOLO detection network that would be trained from this catalog. For example, the MINERVA team illustrated that the expected purity and completeness of the detection vary in a similar size-flux representation in the SDC2. This is informative of how the network structure processes the information. This diagnostic is especially useful to estimate the distribution of the source's flux for each box size prior or to set the flux limits and rescale the corresponding flux distribution if necessary.

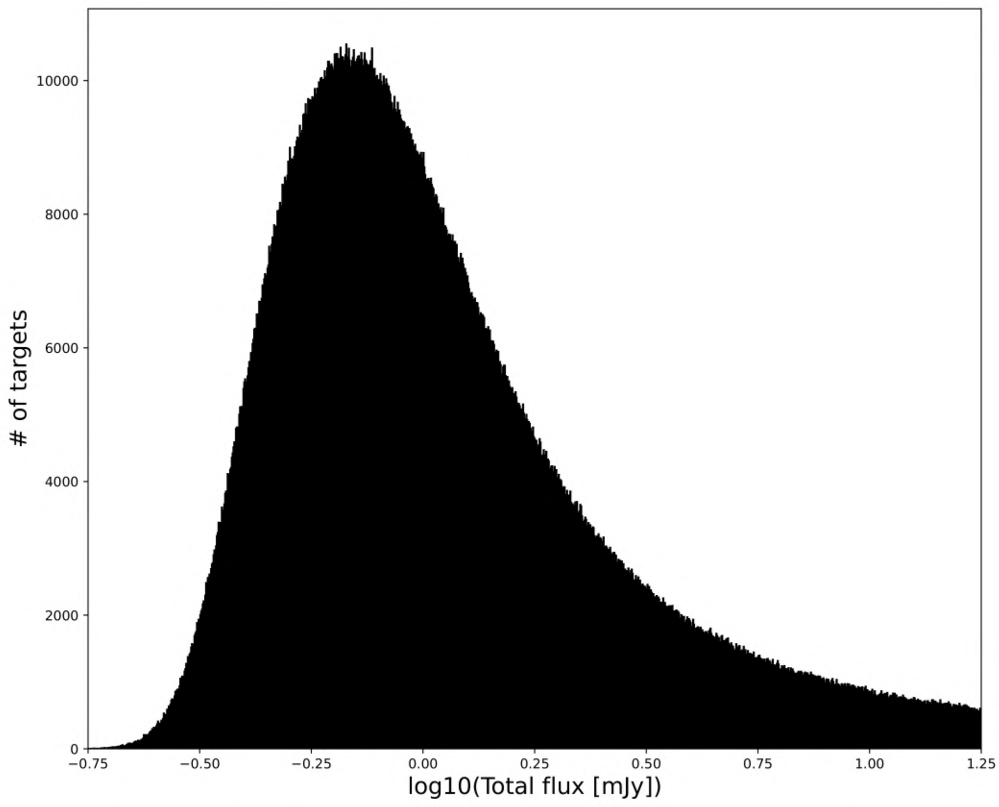


Figure 12: Histogram of $\log_{10}$ flux distribution of sources.

Eventually, the examination of the distribution of sources through a 2D histogram of the minor axis as a function of the major axis shows, as mentioned in the result section, exhibit two different regimes in the distribution. It appears that this effect comes from the way the major and minor axis have been derived. In the LoTSS catalog, there is a flag that defines the source structure in terms of the fitted Gaussian components, called "S-Code". There are 3 distinct flags:

- S refers to an isolated source fitted with a single Gaussian;
- C refers to sources fitted with a single Gaussian but within an island of emission that also contains other sources;
- M refers to sources fitted by multiple Gaussian;

The sources flagged as "M" concerns only 1,199 sources in the whole dataset, which is slim to negligible. The other S-Codes are actually the explanation for the two different regimes observed in our 2D histogram. The sources flagged as "S" concern 4,051,444 sources, which is the majority of the dataset. Most of these sources are located in the upper part of the distribution, illustrated with the figure 13. The sources flagged as "C" concern 343,585 sources. Most of these sources are located in the lower part of the distribution, illustrated in the figure **??** Still, both "S" and "C" sources have the same peak density location in the distribution. This can be explained because the extended sources get elongated because of the surrounding noise from the other radio sources that tend to fit a larger Gaussian. While isolated sources are easier to fit an

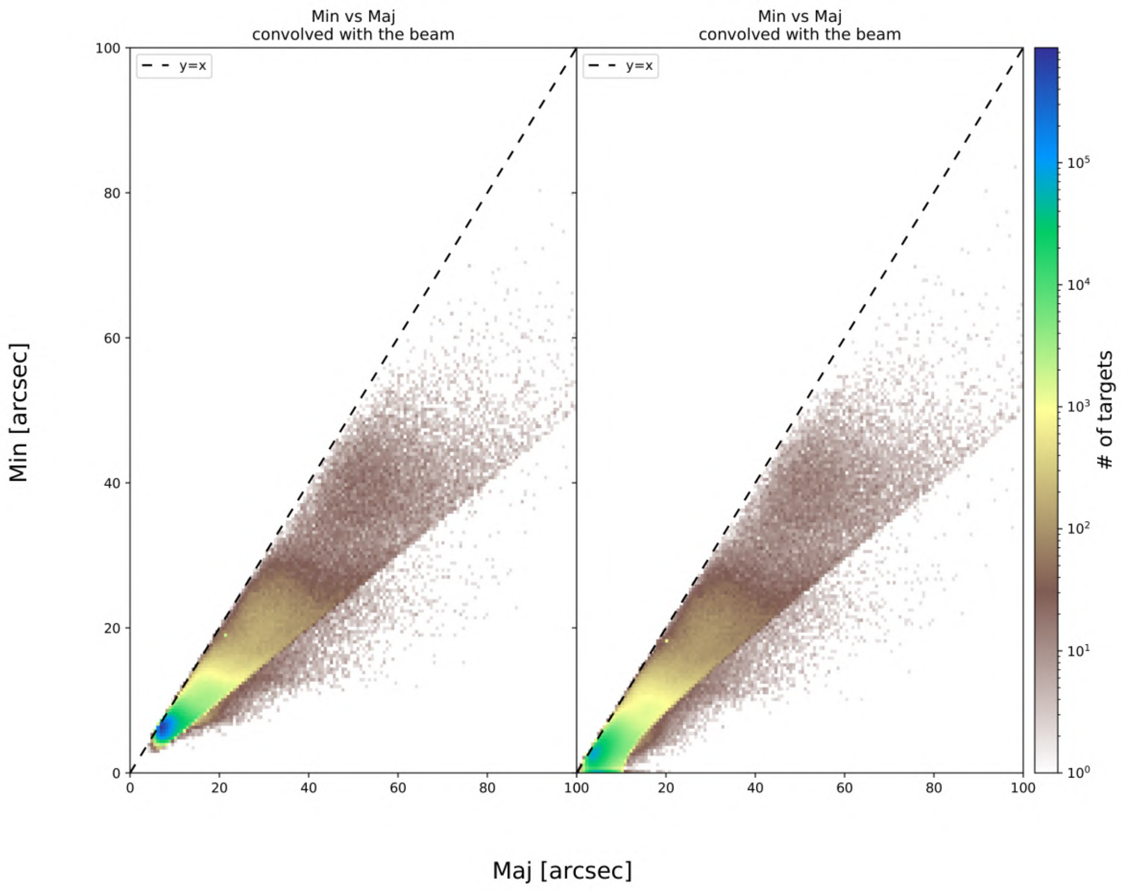


Figure 13: 2D histogram of the major axis versus the minor axis of the sources for both convolved and deconvolved axis for sources flagged with the S-Code "S".

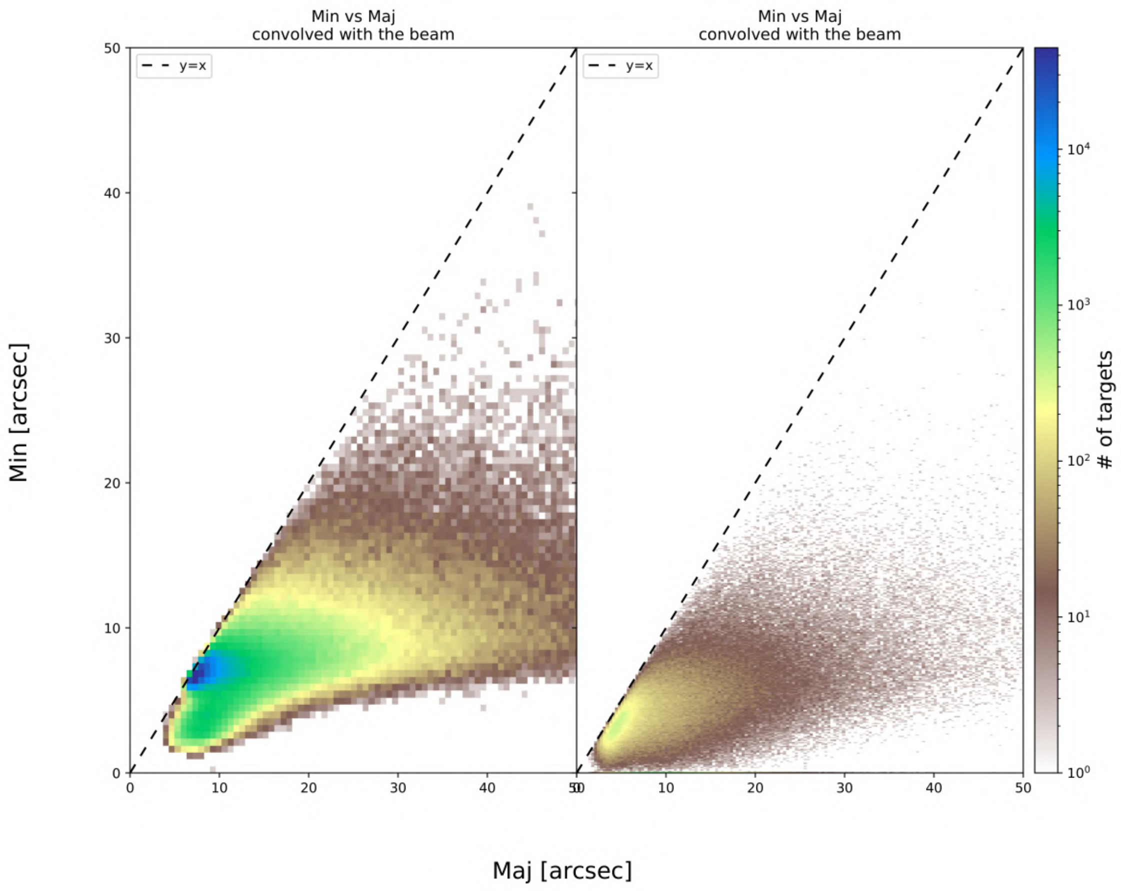


Figure 14: 2D histogram of the major axis versus the minor axis of the sources for both convolved and deconvolved axis for sources flagged with the S-Code "M".

appropriate Gaussian as they are less affected by the neighboring emission. Moreover, the high peak density of the 2D histogram concerns the unresolved sources. In the LoTSS DR2 paper, Shimwell, et al. (2022), the resolution limit is not only based on the beam. There is actually a criteria that depends on the source and its environment to determine if it is resolved. In the histogram, we find a region the corresponding region which contains approximately 90% of the sources with a simple criteria: the sources with a major axis (convolved with the beam) below 12 acrsecs, which is twice as larger than the LOFAR actual beam size.

4.2 Astrodendro results

4.2.1 Performance evaluation

Our motivation with Astrodendro was to construct a pure and comprehensive catalog. This aligns with the objective of the team, which is to improve upon the results of the network trained with SDC1 data. To enhance them, the network will undergo pre-training using the same data as the previous internship, but the complementary training will be conducted using LoTSS data. The objectives of the internship were to build a catalog of high purity and completeness and to evaluate difficult artifact regimes, and the Astrodendro catalog is a good first step in this direction.

The first aspect I want to discuss, regarding the catalog I produced, is the choice of structure used for the catalog. We chose to use the leaves because the catalog produced by this structure is closer to the PyBDSF catalog. As we can see in the table 1, the quality of the catalog derived from the leaves is overall better than the catalog derived from the trunks.

Concerning the results in table 2, having a recall of 59.9% and a precision of 61.9% is a satisfactory result. The difference between the two catalogs is illustrated in the figure 15 tested on the mosaic P7Hetdex11. We see in the figure the limitation of our method. The rejection radius around bright sources encounters some limitations for some bright sources where some artifacts remain in the catalog. This is the case when the surface covered by the detected artifact is large enough to have a high integrated flux, therefore it is not suppressed. Another limitation is on the bright extended sources where our method keeps most of the time multiple points. These detections are not taken into account in our enhancement since their integrated flux is high enough to be kept. Some sources in PyBDSF catalog have characteristics that, according to our method, would have been rejected. This explains a portion of the observed differences in our example. Overall, among all the catalogs derived by the method, 4% of them have a precision below 50%, while 20% of them have a recall below 50%, which is satisfactory.

4.2.2 Perspectives for enhancements

One of the enhancement perspectives concerns the cross-matching with other catalogs. This method could enhance the recall by re-injecting sources excluded by our previous criteria. We could divide our catalog into two parts: one with an high reliability and another one with a low reliability that requires counterparts in other catalogs. I did not implement such a method, but an aspect I explored was the possible fortunate match with catalogs. To have the estimate of the fortunate detection, we took a random number of target n chosen with respect to a normal distribution centered on a density of sources. Then, we randomly spread the n targets over an area of the sky covered by the tested catalog. Finally, we performed the cross-matches with the catalog to see the number of matches. The test have been repeated 10 times, the results are average and given with standard deviation at 1σ level as uncertainty in the table 3: where ρ_1 correspond to DESI density, ρ_2 to Allwise density, ρ_3 to LoTSS density and ρ_4 to the Astrodendro density with the optimized parameters. I found that with our optimal parameters,

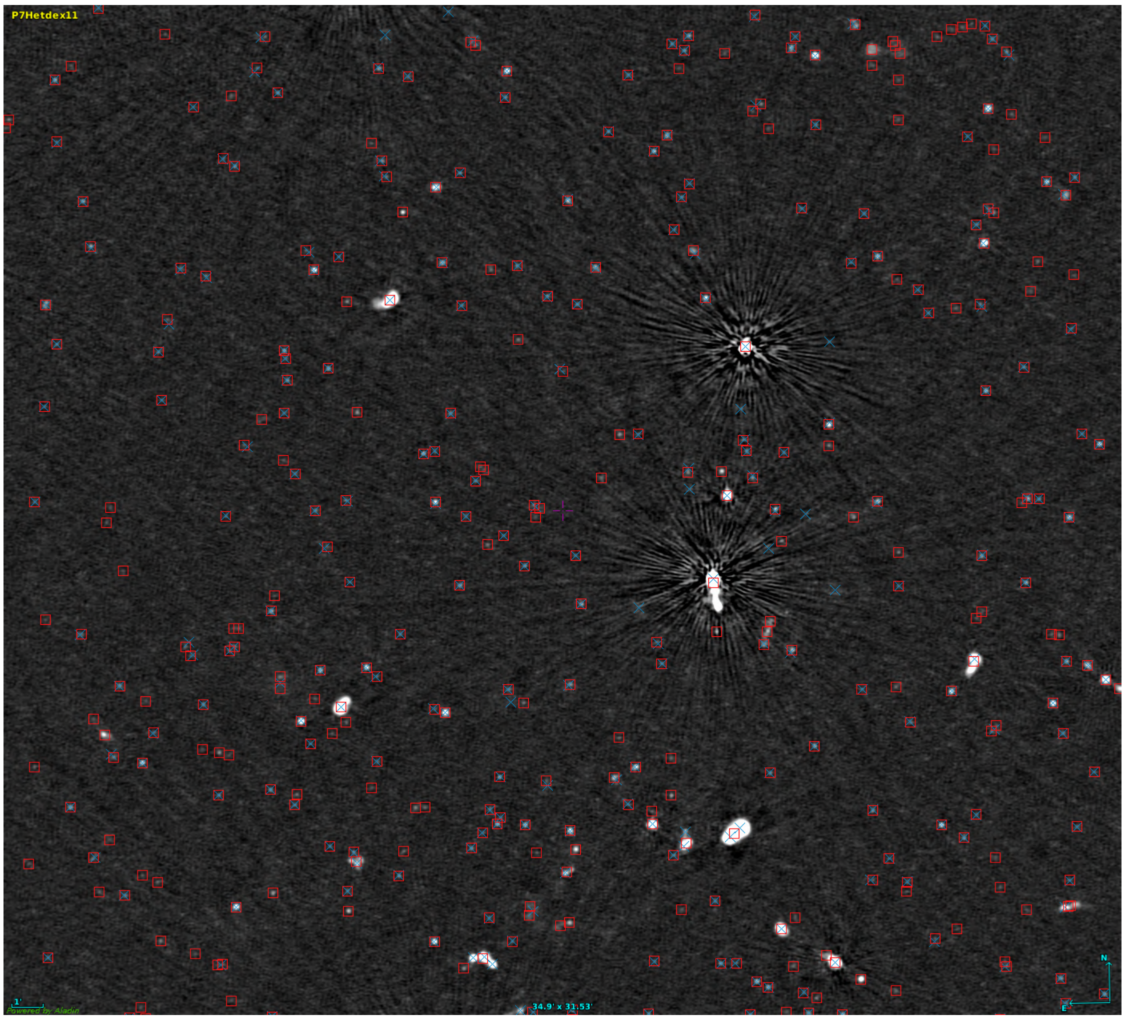


Figure 15: Comparison between the PyBDSF and Astrodendro catalogs in the mosaic P7Hetdex11. The red squares are the sources detected by PyBDSF. The blue crosses are the sources detected by Astrodendro.

source density sources/degree^2	LoTSS (Shimwell, et al., 2022)	Allwise (Cutri, et al., 2014)	DESI (Dey, et al., 2019)
$\rho_1 = 52500$	$15.93 \pm 0.99\%$	$16.42 \pm 0.20\%$	$16.60 \pm 0.22\%$
$\rho_2 = 14663$	$5.41 \pm 0.39\%$	$5.71 \pm 0.13\%$	$5.69 \pm 0.11\%$
$\rho_3 = 1169$	$0.49 \pm 0.23\%$	$0.44 \pm 0.01\%$	$0.43 \pm 0.02\%$
$\rho_4 = 800$	$0.40 \pm 0.08\%$	$0.31 \pm 0.03\%$	$0.31 \pm 0.02\%$

Table 3: Fortunate cross matches with different catalogs for different densities of source.

there are not enough fortunate matches to skew our results, which makes us even more confident with the quality of the catalog obtained.

Concerning the result of the Astrodendro catalogs over the full LoTSS field. The enhancement here would be to implement an unification of the catalogs based on the third NMS discussed in the workflow section 2.4. Because mosaics complete each other with their overlapping neighbour. They can fill holes from one mosaic to another or have a better signal-to-noise ratio in specific regions. Unifying the whole Astrodendro catalog would significantly improve our detection and also the compression with PyBDSF will be easier.

5 Conclusions

To conclude, by studying the properties of the data in the LoTSS DR2 (Shimwell, et al., 2022), I have been able to construct high-confidence catalogs in the LoTSS field. They were built using the package Astrodendro, a classical approach FOR generating dendrograms from observed astronomical data in Python. From these dendrograms, we can produce a catalog of sources candidates with parameters optimized on the specific observed field, and with selection criteria studied in the summary statistic of the LoTSS data. These conditions allowed us to produce high-quality source catalogs on the LoTSS survey. We reached an average recall of $59.9 \pm 15.5\%$ and a precision of $61.9 \pm 7.5\%$. Thanks to the product catalogs, the team is getting closer to a reliable training set specifically for the LoTSS data. This training set will help the network to better identify specificities of the LoTSS and we expect better results for both detection and characterization.

However, the catalogs are not self-sufficient at the moment and require to be unified and find methods to enhance the precision and recall of the data. Once the highest level of quality is reached, the workflow described in section 2.4 will be implemented to proceed with the detection and characterization of the sources.

6 Annex

6.1 Code

All the code I have written during the internship can be found on a github repository[5]. You can find further details on the file README.md as well as in the files themself where any function is commented.

6.2 Figures

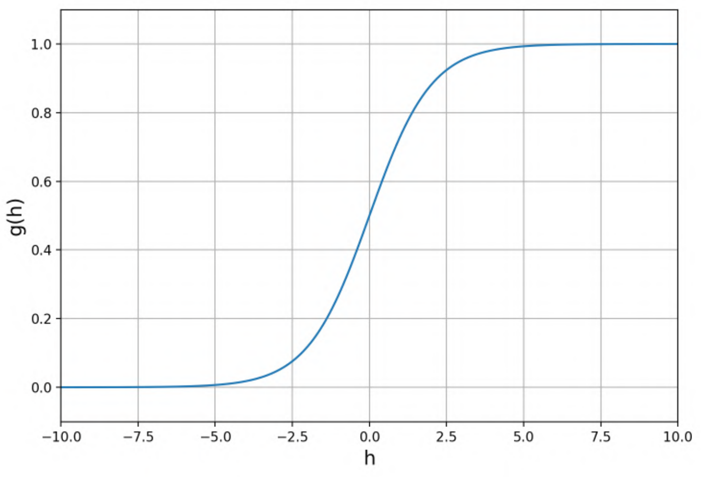


Figure 16: Representation of the sigmoid activation function. The graph derives from the equation 8, where $\beta = 1$.

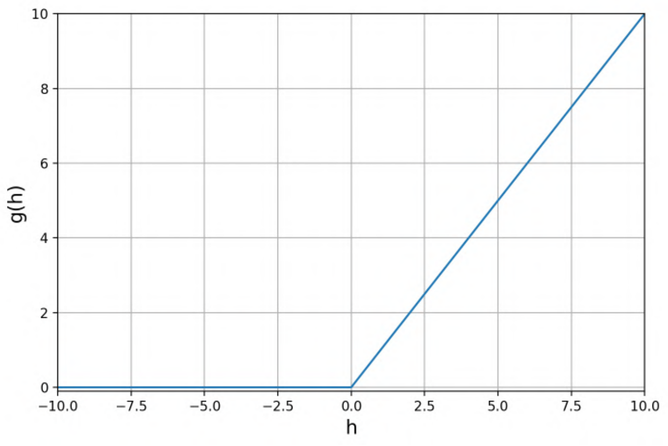
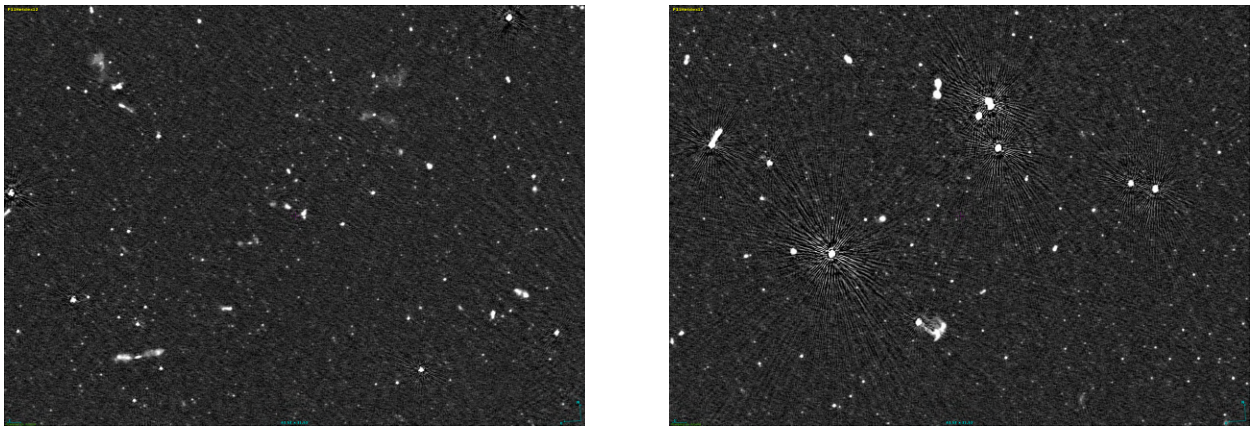


Figure 17: Representation of the ReLU activation function. The graph derives from the equation 16.

[5]https://github.com/ADnothing/M1_Internship


(a) 43.53’× 31.53’ section of the LoTSS field with few artifacts.

(b) 43.53’× 31.53’ section of the LoTSS field many artifacts.

Figure 18: Examples of LoTSS fields displayed in Aladin software. It shows typical features we can encounter in the LoTSS survey. Most of the artifacts encountered in the data are the streaks around bright sources in the radial direction we see in the left image. Even if not all the sources have those artifacts, we can still see them in a distance such as in the right image.

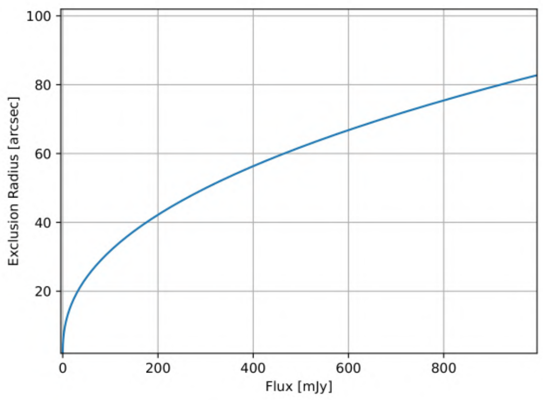


Figure 19: Rejection radius used to clean the artifacts around bright sources in the LoTSS field. This plot is obtained from the equation (19)

References

Bonaldi, et al. Square Kilometre Array Science Data Challenge 1: analysis and results. *Monthly Notices of the Royal Astronomical Society*, 500(3):3821–3837, 10 2020. ISSN 0035-8711. doi: 10.1093/mnras/staa3023. URL https://doi.org/10.1093/mnras/staa3023.

D. Cornu. *Modeling the 3D Milky Way using Machine Learning with Gaia and infrared surveys*. Theses, Université Bourgogne Franche-Comté, Sept. 2020. URL https://theses.hal.science/tel-03155785.

Cutri, et al. Allwise data release, 2014.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*. doi: 10.1007/BF02551274. URL https://doi.org/10.1007/BF02551274.

Dey, et al. Overview of the desi legacy imaging surveys. *The Astronomical Journal*, 157(5):

168, apr 2019. doi: 10.3847/1538-3881/ab089d. URL https://dx.doi.org/10.3847/1538-3881/ab089d.

V. Dumoulin and F. Visin. A guide to convolution arithmetic for deep learning, 2018.

Hartley, et al. SKA science data challenge 2: analysis and results. *Monthly Notices of the Royal Astronomical Society*, 05 2023. ISSN 0035-8711. doi: 10.1093/mnras/stad1375. URL https://doi.org/10.1093/mnras/stad1375. stad1375.

Y. Lecun and Y. Bengio. Convolutional networks for images, speech, and time-series. 01 1995.

McCulloch, W. S. and Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943. doi: 10.1007/BF02478259. URL https://doi.org/10.1007/BF02478259.

V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA, 2010. Omnipress. ISBN 9781605589077.

J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger, 2016.

J. Redmon and A. Farhadi. Yolov3: An incremental improvement, 2018.

J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection, 2016.

Rumelhart, et al. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986. doi: 10.1038/323533a0. URL https://doi.org/10.1038/323533a0.

Shimwell, et al. The lofar two-metre sky survey - v. second data release. *A&A*, 659: A1, 2022. doi: 10.1051/0004-6361/202142484. URL https://doi.org/10.1051/0004-6361/202142484.