Observatoire de Paris - PSL

M2 Space Science and Technology: Internation Research Track

LIU report: Deep Learning for galaxy detection on radio-astronomical surveys

Author:
Adrien ANTHORE

Supervisor:
David Cornu

M2 Lab Insertion Unit

LERMA UMR 8112, Paris

July 20, 2024

Abstract

This LIU project takes place in the context of an international effort to develop new data processing and analysis tools for the upcoming Square Kilometer Array (SKA). Members of the MINERVA project from the Paris Observatory developed a deep-learning approach for source detection and characterization that demonstrated state-of-the-art performance on simulated radio-continuum images from the SKAO first data challenge (SDC1). In this report, I show that the direct application of a network trained on the simulated data from the SDC1 holds satisfactory results on observational radio surveys, namely the LoTSS and the RACS. Intending to improve these results by identifying more specific instrumental artifacts, I have built a complementary high-confidence training sample based on a classical detection method. This approach produced source catalogs that are competitive with the reference catalogs obtained from other classical methods. This work is a strong step toward high-performance deep-learning source detectors specifically constrained on observational radio-astronomical surveys.

Keywords— Radio continuum: galaxies - Instrumentation: interferometers - Galaxy: general - Methods: data analysis - Deep Learning - Neural Networks - Source detection

Contents

1 Introduction

This LIU project takes place in the context of an international effort to develop new data processing and analysis tools for the upcoming Square Kilometer Array (SKA), the first light of which is scheduled for 2028. This instrument will reach an unprecedented sensitivity, allowing it to set new constraints on the early stages of the universe and better constrain the evolution of astronomical objects over cosmological times. However, the analysis of the data produced by the SKA will be very challenging. The current forecasts predict up to 700 PB of archived data per year and a raw data output of around 1 TB per second, which must be processed in real-time to produce science data products. With such a data rate and with data products that are large and highly dimensional, current widely adopted analysis methods exhibit significant limitations. For this reason, new innovative analysis methods must be developed alongside the SKA deployment. For this, the astronomical community makes use of current instruments that are considered "pathfinders" or "precursors" for the SKA (e.g., ASKAP, MeerKAT, LOFAR, etc.).

In this context, the SKAO (SKA Observatory) started a series of data challenges (SDC), intending to provide simulated data products that should be representative of the SKA. These challenges aim to compare analysis tools on controlled datasets and encourage the development of new data analysis methods. While these SDCs seek to represent a variety of analysis tasks, the first two editions of the challenges (SDC1 Bonaldi et al. (2020) and SDC2 Hartley et al. (2023)) focused on source detection and characterization. The first edition was about detecting sources in simulated 2D continuum images, and the second edition in a simulated 3D HI emission cube.

To participate in the SDC2, members of the MINERVA project from the Paris Observatory developed a deep-learning approach for source detection and characterization that would be applied to both 2D and 3D datasets. This new innovative method demonstrated state-of-the-art performance on both types of simulated data. The team noticeably reached first place in the SDC2 and achieved much better results than all the solutions submitted during the SDC1. The method must now be generalized on observational radio datasets, which can be done using various surveys from precursor instruments.

This LIU project lies in this global effort of applying the method to existing radio datasets. I noticeably focused on trying to apply the detector trained on the first SDC on observational continuum surveys, namely the LoFAR Two-metre Sky Survey (LoTSS DR2 Shimwell, et al. (2022)) and the Rapid ASKAP Continuum Survey (RACS DR1 McConnell et al. (2020)). The two surveys are already associated with a source catalog derived using the same classical detection method. These catalogs will be considered as references to which our detection results will be systematically compared. Because MINERVA's source detector is based on supervised machine learning, it must be trained for the survey it will be applied to. This can be done in two different ways: i) by using simulated examples, or ii) by using actual observations with labels from another method or confirmed observationally using other data.

This project is separated into two steps. The first step is to evaluate MINERVA's method capabilities on the selected surveys by performing source detection using the network trained on the simulated data from the SDC1. By doing this, we will evaluate if the SDC1 is a good enough approximation of observational data for both the LoTSS and the RACS. The second step consists of using observational examples to perform complementary network training to better account for the survey's specificities. This approach requires the construction of a comprehensive and high-quality source catalog based on a different method than the one used to define the reference source catalog.

This report comprises two main parts that follow the two objectives of the project. In section 2, I will introduce the required theoretical context of artificial neural networks, describe MINERVA's deep learning method for source detection and characterization, and present results of direct application of the SDC1 network to LoTSS and RACS images. In section 3, I will present the classical detection method AstroDendro and its application to LoTSS and RACS to construct high-quality source catalogs that could be used as complementary training samples. Eventually, I will put both approaches into perspective regarding the overall progress of the project in section 4.

2 Machine Learning based detection

In this section, I will describe our Machine Learning framework in a general context, then the prediction pipeline used to make detection on a survey. I will highlight our objectives as well as the results of the detection. Eventually, I'll discuss them and provide clarification on our observations.

2.1 Context and objectives

The objective of the detection is to find sources in radio images and to extract key parameters such as the size of the source (major and minor axis), the integrated flux, and the position angle (PA) from the sources. This type of detection has been performed several times for SKA precursors with classical methods. However, the increase in the volume and dimensionality of radio-astro data is a challenge for the classical method, while it is an advantage for statistical methods that require large databases. ML methods therefore make it possible to see an improvement in the quality of detections while being more numerically efficient on large masses of data than classical methods. In this context, the MINERVA team has developed its detection and characterization method using deep neural networks. The framework used by the team is CIANNA[1] (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. It demonstrates state-of-the-art performance on simulated 2D continuum images. We will now test the method on the two observational surveys LoTSS and RACS.

2.2 ML methodology

The definition of Machine Learning is not straightforward, one can define ML as "having a computer extract statistical information from a dataset and adapt its response through an iterative process" (Cornu, 2020). While no single and exclusive definition exists, this view encompasses a comprehensive understanding of ML. The method chosen for our research employs a neural network operating under a supervised learning paradigm. This approach uses a training dataset containing each example's expected output. The method attempts to reproduce the expected target for each input. It learns by comparing its current output with the target and correcting itself based on this comparison. After training, such algorithms can generalize to objects similar to those used for training (Cornu, 2020). I will now explore the fundamental aspects of neural networks and deep learning in the context of our approach.

2.2.1 Neural Network and deep learning

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

Artificial neural networks are the most common ML family of methods 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 is connected to multiple others 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 to other neurons.

The artificial neuron mimics this process. It consists of an input vector X_i with the same dimension m as 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. 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 us to define a non-zero origin to the linear separation of the neuron. The product of the weights with their corresponding neuron are then combined in a sum function h:

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

where b is the bias input and ω_0 its associated weight. Then, the result of the weighted sum passes through an activation function $g(h)$ that defines the state at the output of the neuron. We can emulate one of the simplified properties of a biological neuron, which is its ability to be in an "active" state (state

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

1) or "inactive" state (state 0) based on the total signal received from a simple step function. However, it is more common to use continuous and differentiable activation functions. For instance, the Rectified Linear Unit activation or ReLU (Nair and Hinton, 2010):

$$a_j = g(h_j) = \begin{cases} h_j & \text{if } h_j \geq 0 \\ 0 & \text{if } h_j < 0 \end{cases} \tag{2}$$

This function is simply linear for any input value above zero and equal to zero if the input is negative. It 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. The ReLU function is illustrated in figure 1.

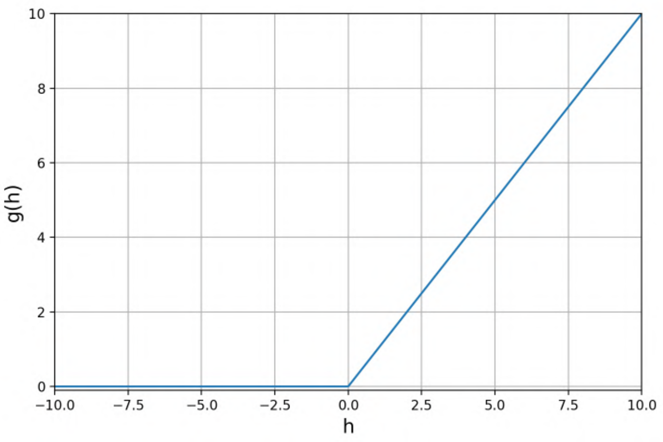


Figure 1: Representation of the ReLU activation function. The graph derives from the equation 2.

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 of finding an appropriate combination of unique 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 an 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 (2) with a square error: $E = 0.5 \times (a-t)^2$, the correction of the weights is implemented as follows:

$$\omega_i \leftarrow \omega_i - \eta(a-t)X_i \tag{3}$$

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 weight, are non-null, and are not selected by the user directly. For instance, a normal distribution centered around 0 with a rather small variance is usually considered a good initialization for the weights. Note that for each feature the same weight will consistently be assigned. This process is illustrated in figure 2.

To treat more complex problems, more neurons are necessary. One straightforward approach for combining neurons is to use them independently but connected to the same input. This structure is called a "layer". Each independent neuron makes a linear separation, and using several of these separations makes it possible to solve a more complex problem. It allows us to predict several quantities at the same time from the same data. Each neuron is connected the same way as the single neuron to the input layer and

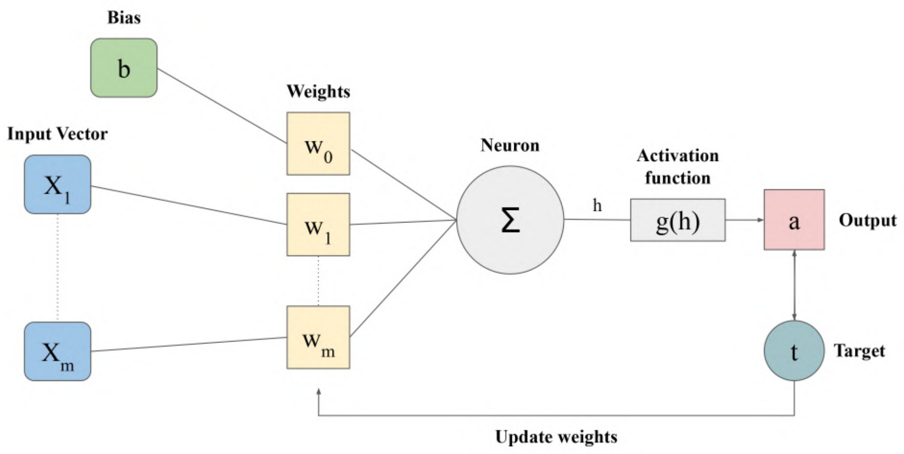


Figure 2: Schematic view of a binary neuron. $X_{[1,\ldots,m]}$ are the input features for one object in the training dataset and b is the bias, $w_{[1,\ldots,m]}$ are the weights associated with each feature and w_0 is the weight of the bias. Σ represents the sum function and h its results.

has the same behavior. However, each neuron operates independently. In this case, the weighted sum and the activation equations for each neuron j become:

$$h_j = b\omega_{0j} + \sum_{i=1}^{m} X_i \omega_{ij} \tag{4}$$

and

$$a_j = g(h_j) = \begin{cases} h_j & \text{if } h_j \geq 0; \\ 0 & \text{if } h_j < 0. \end{cases} \tag{5}$$

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

$$\omega_{ij} \leftarrow \omega_{ij} - \eta(a_j - t_j)X_i \tag{6}$$

Networks built with this formalism are called the perceptron.
Each neuron applies a non-linear activation function, but since the inputs are added in sum weighted by the weights, the information that the neuron receives is only a linear combination of the input. To add non-linearity, we can take the output of several neurons after their non-linear activation function consider this as our new feature space, and add a layer of neurons. 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 expressivity of the network and must be adapted to the task's difficulty. This type of network architecture is called a Multi-Layer Perceptron (MLP).

The multilayer architecture of this network enables a non-linear combination of the activation 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" as demonstrated by **?**.

The addition of new layers in the network doesn't change much the forward processes, but the update of the weights described by equation (6) 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:

$$E(a,t) = \frac{1}{2}\sum_{k=1}^{N}(a_k - t_k)^2 \tag{7}$$

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 :

$$\omega_{ij} \leftarrow \omega_{ij} - \eta\frac{\partial E}{\partial \omega_{ij}} \tag{8}$$

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

$$\frac{\partial E}{\partial \omega_{ij}} = \delta_l(j)\frac{\partial h_j}{\partial \omega_{ij}} \quad \text{with} \quad \delta_l(j) \equiv \frac{\partial E}{\partial h_j} = \frac{\partial E}{\partial a_j}\frac{\partial a_j}{\partial h_j} = \frac{\partial a_j}{\partial h_j}\sum_k \omega_{kj}\delta_{l+1}(k) \tag{9}$$

In these equations, the i index refers to the layer's input dimensions, and the j index 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 (9) 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 ReLU activation for all neurons:

$$\frac{\partial h_j}{\partial \omega_{ij}} = a_i \tag{10}$$

the activation of the current layer,

$$\frac{\partial E}{\partial a_j} = (a_j - t_j) \tag{11}$$

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

$$\frac{\partial a_j}{\partial h_j} = \begin{cases} 1 & \text{if } h_j \geq 0 \\ 0 & \text{if } h_j < 0 \end{cases} \tag{12}$$

the derivative of the ReLU activation.

2.2.2 Neural Networks for images

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 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. This operation is by nature equivariant by translation. The filter is applied to sub-regions of the images at regular intervals with a shift in pixels between each application called stride S. For a 2D image, the convolution uses a 2D filter that is applied regularly in both dimensions following the stride S as shown in figure 5. A side effect of the operation is to reduce the spatial size between the input and the output, this phenomenon is due to the image edges. 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 :

$$w_{out} = \frac{w_{in} - f_s + 2P}{S} + 1 \quad h_{out} = \frac{h_{in} - f_s + 2P}{S} + 1 \tag{13}$$

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

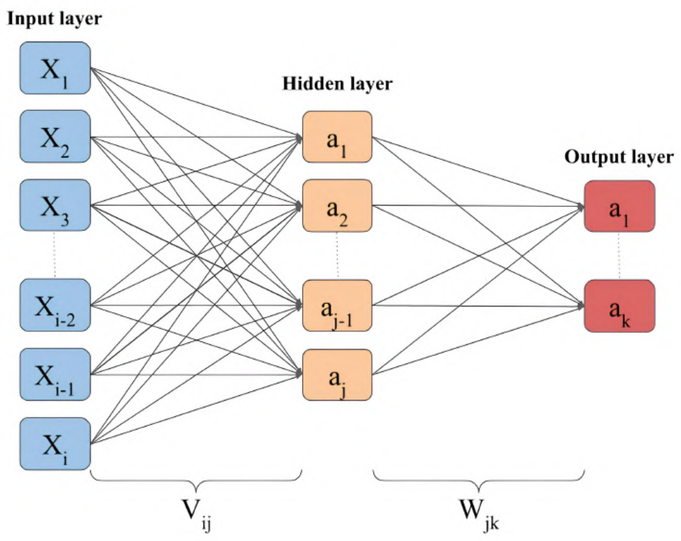


Figure 3: Schematic view of a simple "deep" neural network with only one hidden layer. The blue cells are input dimensions. The beige cells are the hidden layer dimensions. And the red cells are the output dimensions. $X_{[1,\ldots,i]}$ are the dimension for the input vector, $a_{[1,\ldots,j]}$ are the activation for the hidden neurons, $a_{[1,\ldots,k]}$ 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.

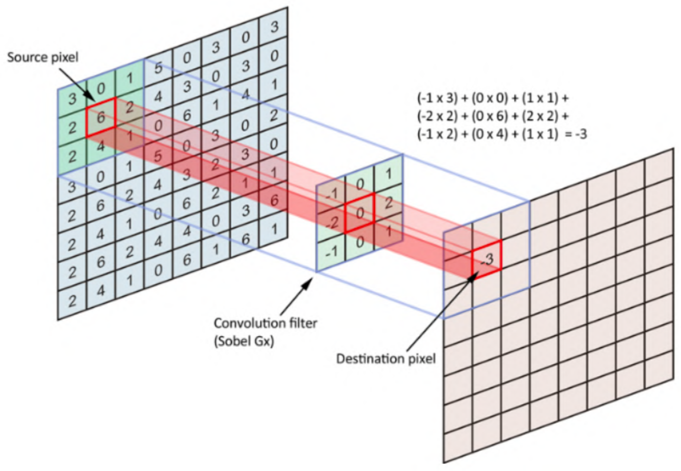

Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
(Sobel Gx)
Destination pixel

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

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 to scan for a given pattern at all possible locations. Just like with the perception, to 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 5.

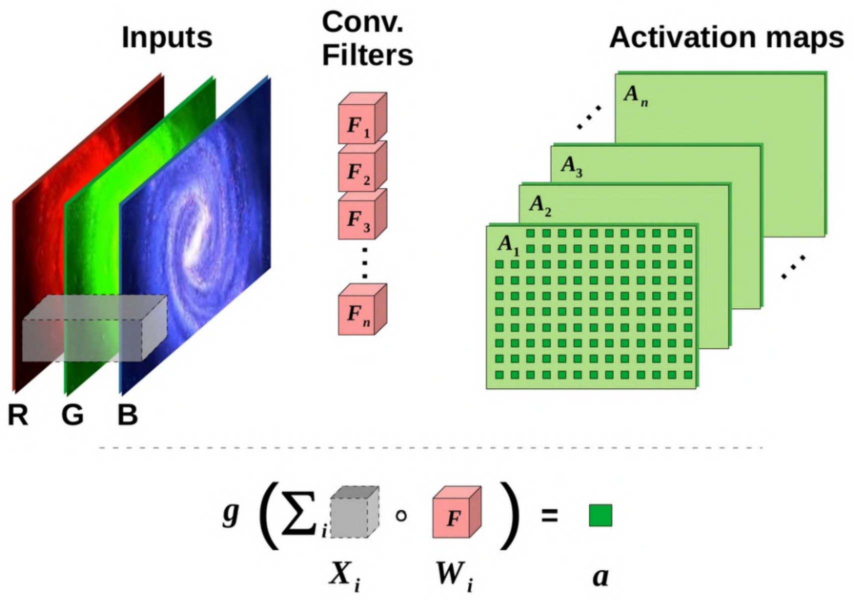


Figure 5: Schematic representation of a convolutional layer. The input consists of a three-channel image (RGB) with n filters. The output consists of n activation maps. - Credit: David Cornu

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 multi-channel input image. Increasing the number of convolutional layers allows the network to construct 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 spatial 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 ends 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.

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 (9) for each sub-region of the considered operation input image. After the error propagation, each convolutional layer can update its filters by summing the correction from all the regions they were applied to.

2.2.3 You Only Look Once (YOLO)

Object detection is a common task in computer vision, and Deep Neural Networks (DNN) are an effective approach. The deep learning method developed by the MINERVA team is inspired by the YOLO (You Only Look Once) architecture. The method used was built by David Cornu with a combination of features and capabilities that are a mixture of the YOLO v1 (Redmon et al., 2015), v2 (Redmon and Farhadi, 2016) and v3 (Redmon and Farhadi, 2018) architecture, and that was implemented inside the CIANNA framework. This method belongs to a family of regression-based object detectors. 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 than 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. The grinding of the image is shown in the first frame of figure 8. 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 format of the output vector is given in figure 6.

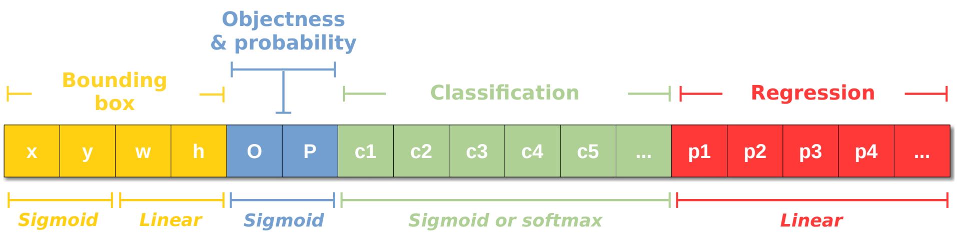


Figure 6: Output vector from a single grid output grid cell for a single detection unit. The elements are colored by type (geometry, probability, classification, and other properties), and the corresponding activation functions are indicated. For multiple detection units, multiple identically shaped vectors are repeated on the same axis. Credit: David Cornu

The center of an object is contained in a single grid cell, but the object itself can be larger or smaller than the grid cell. At first, the box is associated with a prior given as a hyperparameter in the network. A good association with a prior helps the network predict the best bounding box. To derive the bounding box (x,y,w,h), each grid cell must predict a 4-element vector (o_x, o_y, o_w, o_h) from which we derive the box's geometric properties as follows:

$$x = \sigma(o_x) + g_x, \tag{14}$$
$$y = \sigma(o_y) + g_y, \tag{15}$$
$$w = p_w e^{(o_w)}, \tag{16}$$
$$h = p_h e^{(o_h)}. \tag{17}$$

where (g_x, g_y) correspond to the relative position of the grid cell, and $(o_x,\ o_y)$ are the position inside the grid cell. The size of the object is determined from the exponential of the predicted values $(o_w,\ o_h)$ that act as a scaling on the associated prior (p_w, p_h). This process is illustrated in figure 7, where the detected object is a galaxy. The bounding box's output is represented in red, while the geometrical elements to compute them are represented in black.

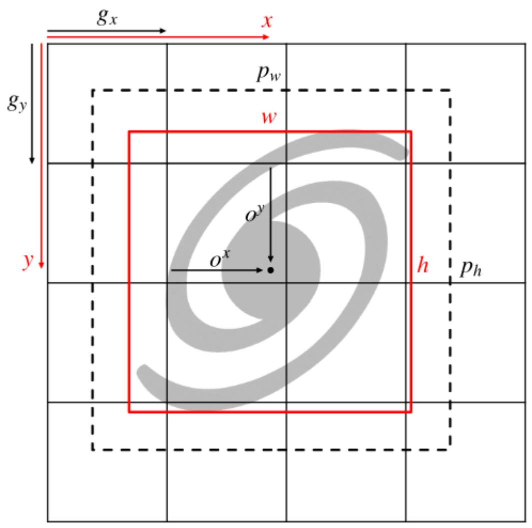


Figure 7: Illustration of the YOLO bounding box representation based on the output grid construction. The dashed black box corresponds to the theoretical prior, while the red box represents the network's predicted size. Credit: David Cornu

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 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 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), 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 no more boxes are in the open list of predicted boxes. This process is illustrated in figure 8 where all the predicted boxes are displayed on the middle frame, and the right frame is the result of the NMS where the remaining boxes are the boxes corresponding to the predicted objects in the image.

In the context of the LIU, the detection of the radio sources predicts both the central position and box size (Bounding box in figure 6) for each object but also a list of complementary parameters (flux, PA) that are predicted by the network as a regression (Regression in figure 6) thanks to the specific implementation inside the CIANNA framework. As we are not classifying the detected sources, we don't have a classification segment in the output vector. The dimension of our output vector is 11 with 4 dimensions for the bounding box, 2 dimensions for the probability and objectness, and 5 dimensions for the regression parameters (integrated flux, Bmin, Bmaj, cos(PA), sin(PA)).

2.3 Detection process and results

As for the previous detection made on the LoTSS by the team, the network has been trained on the simulated data from the SDC1 for 1600 epochs. The number of epochs was chosen by evaluating the network trained at different numbers of epochs and taking the one that had the best performance on the SDC1.

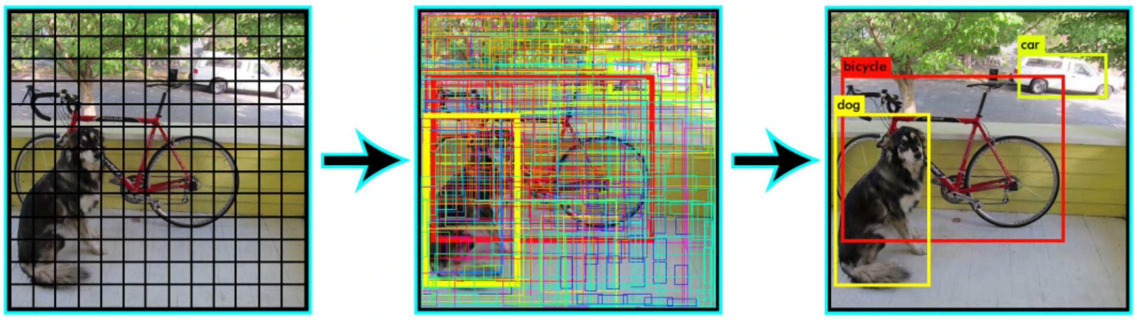


Figure 8: Representation of an image with the output grid and the process of the detection. The left frame is the input image on which the YOLO gris has been super-imposed. The middle frame is the input image on which all the prediction boxes are represented, the thickness of the box lines represents the score (thicker is higher). The right frame represents the best prediction boxes selected by the NMS. Source: https://pjreddie.com/darknet/yolo/

2.3.1 Network description

The network used during my LIU is a YOLO network. It is a dense CNN of 17 convolutional layers with a YOLO layer attached at the end. A schematic representation of the network is given in figure 9. Its structure is the result of previous testing on the SDC1 data, and the fine tunings were obtained through optimization of the SDC1 score. The output of each convolutional layer is activated with the ReLU function (Eq. 2). The detector predicts 9 independent boxes in the same grid cell, each characterized by size priors (p_w, p_h), which are parameters of the network. In our configuration, we chose square priors $(p_w = p_h)$, resulting in 6 boxes measuring 6 by 6 pixels, 2 boxes measuring 12 by 12 pixels, and 1 box measuring 24 by 24 pixels.

2.3.2 Workflow

The radio surveys are composed of large images, much larger than the typical network input, and these large images overlap with each other. Hence we need an inference pipeline that makes it possible to obtain a result on an entire survey. The objective is to make the detection on each image without making redundant detections.
The workflow begins by looping over the mosaics, each of which is processed with a hyperbolic tangent and decomposed into tiles of 256x256 pixels. The flattened tile corresponds to the input vector for the network. The detection is performed independently on each tile. In each tile, we first remove detection with a too-low objectness, then we proceed to a first NMS to deal with overlapping detections. As the tiles overlap, a second NMS is subsequently used to remove redundant detections within the mosaic.
At the end of the mosaic loop, individual catalogs of detections are obtained for each mosaic. The following process is used to handle overlaps between mosaics. We first look for the overlapping mosaics. We consider that 2 mosaics overlap if the distance between their center is lower than the sum of their diagonals. Then, sources within the overlapping region are sorted based on their objectness. We then iterate over all the sources going from the highest to the lowest objectness. Sources closer to 45 arcseconds than the current source are removed. This threshold in distance was determined empirically.

2.3.3 Detection performances

To evaluate the performance of the detection by the network, we will compare the source catalog obtained with the source catalog of the RACS obtained by a classical method. For this, we perform a nearest-neighbor cross-match, defining a match between two sources if their separation is less than 25 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 RACS catalog as a reference. The recall is defined

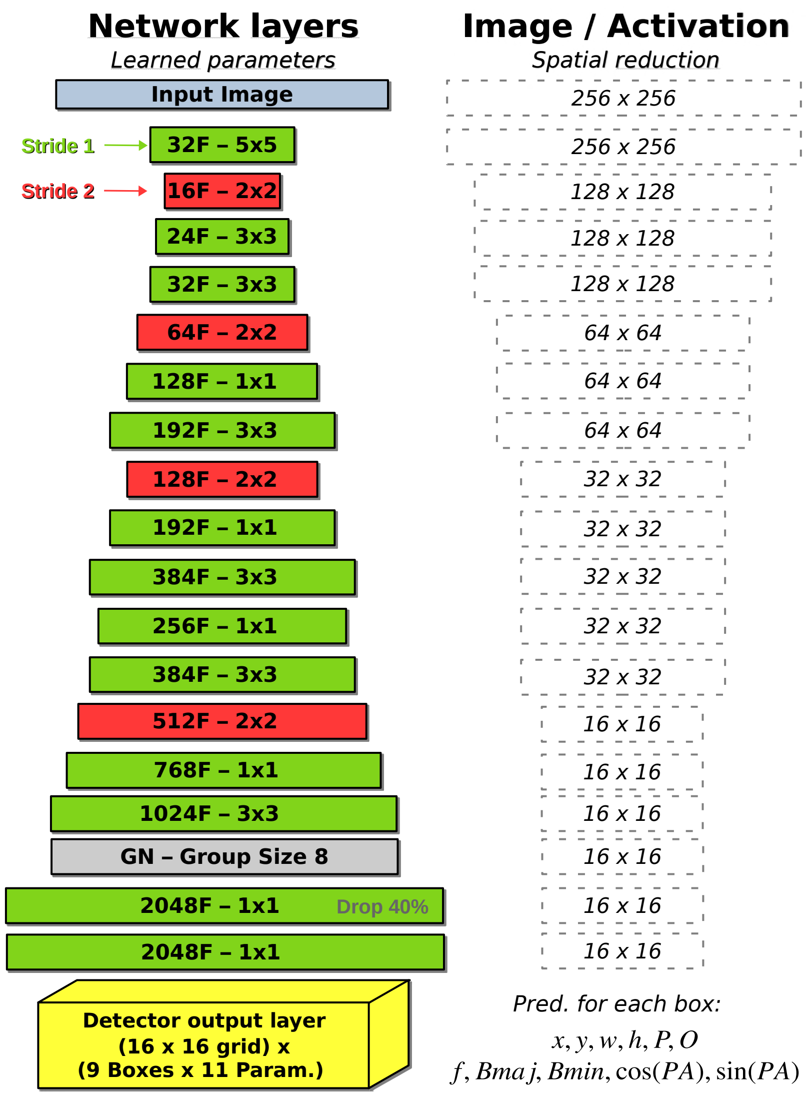


Figure 9: Schematic representation of our YOLO-based CNN. On the left is represented the layers with the number of filters and their size, and on the right side is represented the dimension of the output at each layer. The color represents the stride used, when the stride is equal to 1 (in green), there are no changes in spatial dimension for the output while if the stride is equal to 2 (in red) the output dimension is divided by 2 on both axes (4 in total for the 2D case).

as:

$$\text{recall} = \frac{\text{N}_{\text{match}}}{\text{N}_{\text{ref}}} \tag{18}$$

where N_{match} is the number of cross-matched sources, and N_{ref} is the number of sources in the reference catalog. The precision is defined as:

$$\text{precision} = \frac{\text{N}_{\text{match}}}{\text{N}_{\text{net}}} \tag{19}$$

where N_{net} is the number of sources detected by the network.

The method has been applied to 20 of the 799 mosaics from the RACS data, the methods held ∼54% recall with ∼65% precision in the detection. An example field is shown in figure 10.

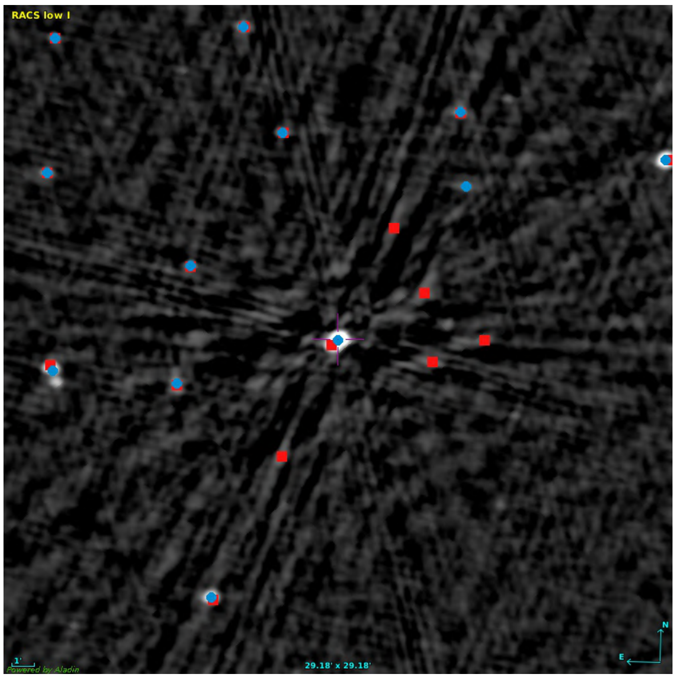


Figure 10: Example field of 29.18’$\times$ 29.18’ from the RACS with detection from the network. The squares in red represent the predictions from the network. The circles in blue represent the reference catalog.

2.4 Result analysis

The reason we chose the LoTSS and RACS is that they are comparable to the simulated data from SDC1. The LoTSS consists of high-resolution 120-168 MHz images covering 27% of the northern sky with 841 mosaics observed 8 hours each at a 6 arcseconds resolution. From this data, a catalog of 4,396,228 radio source candidates has been derived. The RACS consists of 799 mosaics observed 15 minutes each that could be convolved to a common resolution of 25 arcseconds. It covers the southern sky up to +41 deg declination at 887.5 MHz. From this survey, a source catalog of approximately 2 million (excluding $|b| < 5°$ around the Galactic plane). Both surveys are 2D continuum images as the simulated images from the SDC1 but with different properties. This is why we wanted to test the performance of our method on this data with a network only trained on the data from the SDC1.
The inference work on LoTSS had already been started and held satisfactory results. We might believe that the network could have trouble dealing with the meager signal-to-noise data from the RACS. Indeed, each field on the RACS is observed for 15 minutes, whereas each field on the LoTSS data is observed for 8 hours. With that considered, the network proved that it can handle noisy images since it showed once again great performance in the detection with a reasonable amount of detection. However, there are still a considerable amount of undetected sources and false detection for which we have possible explanations. There are two main kinds of issues in our detection. The first kind of issue comes from the interferometric artifacts around the bright sources. Around bright sources, there is a considerable amount of false detections that come from the artifacts shown in figure 13. We can compensate for these false detections by rejecting the detections at a certain distance around the bright sources The second kind of issue is the missing detections. They can come from the morphology of the object, for example AGN-like object with jets, or from sources with a very low integrated flux.

Another issue that has decreased the quality of the detection comes from the fact that the network has been trained with much higher-resolution images. To avoid such an issue, it was necessary to apply processing to the mosaics, ensuring that the images presented to the network shared the same PSF size as the simulated images used for training. This would lead to a better prediction of the sizes but also of the position. Although this modification would provide better results, still the detection will miss typical features from the survey. The reason is that they are poorly or not represented in the training dataset from the SDC1. The solution to this problem is to train the network to the specific data with a complementary training dataset. This training dataset will contain a complete catalog of the typical objects present in the surveys studied, allowing the network to take into account the artifacts without filtering, have a better SNR limit at which detection is relevant, and take into account more different morphologies absent from the SDC1.

3 Classical detection

In this section, I will describe the classical process employed for object detection, highlighting our objective with this process and the results of the method on the surveys. Eventually, I'll discuss the result and provide further explanations of our observations.

3.1 Context and objectives

The objective of the classical detection is to build a complementary training data set to enhance the detection provided by the Machine Learning method. With this complementary training, the network will learn the specificities of the survey studied. There are already source catalogs derived from classical methods, however, we find some detections questionable, and we want to use them to evaluate our ML method. If we used catalogs, we would train our network to find the same types of sources as them, which would bias our analysis. Hence the need to perform a classical detection by another method taking into account the specific needs of an ML training dataset for source detection, which is not the case for existing catalogs. During my previous internship, I developed a method based on the astrodendro python package[2]. This method served as a proof of concept on the LoTSS survey and has been enhanced during my LIU to be generalized to different surveys, especially RACS, and also to be more efficient overall.
We aim to construct a pure and complete source catalog that is as much as possible representative of the survey data. This catalog will then be used as a complementary training dataset for our network. The quality of the network's detection capabilities is intrinsically tied to the quality of its training data. The network will detect the same kind of sources as those seen during its training. Therefore, our search for a catalog of high-reliability classical sources is motivated by the importance of providing a solid foundation for our ML detection, ensuring that the network can identify sources with precision and accuracy.
Both RACS and LoTSS source catalogs use another classical method with the Python package PyBDSF[3] (Python Blob Detector and Source Finder).

3.2 Method description

The dendrogram method works as follows: starting from the highest flux value, it iterates downwards with a determined step size until it reaches a selected baseline value, as seen in figure 11. At each iteration, pixels are joined together, depending on their proximity as well as their flux level, to create a structure. The smallest structures are called leaves, and a group of nearby leaves form a trunk. We can choose the minimum number of pixels to create a leaf. For our detection, we only consider the leaves as sources.

In the astrodendro package, the parameters described previously are implemented in the built-in method for the detection. The parameters are described like so:

- *min_value*: the flux baseline value;
- *min_delta*: the flux iteration step;
- *min_npix*: the minimum number of pixels to consider a structure.

3.2.1 Parameter Selection

All detection methods will inevitably make false detections in the mosaics produced by ASKAP or LoFAR. This is due to numerous artifacts in the field with enough local contrast to be identified as sources.

[2]https://dendrograms.readthedocs.io/
[3]https://pybdsf.readthedocs.io/

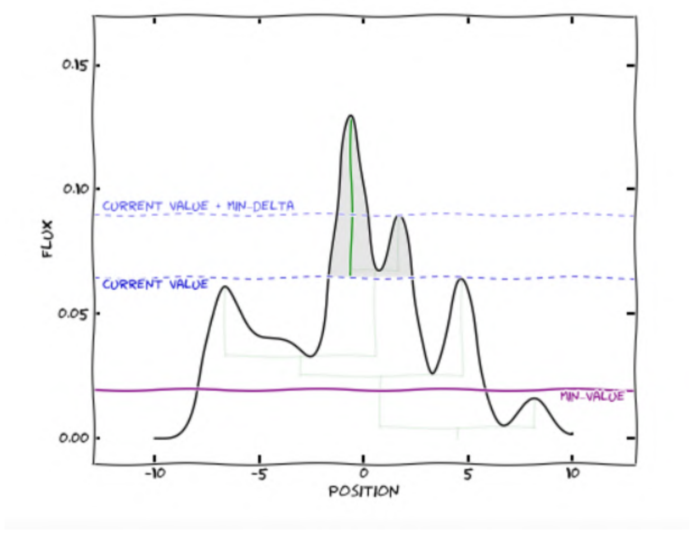


Figure 11: Illustration of the Astrodendro algorithm. - Source: https://dendrograms.readthedocs.io/en/stable/algorithm.html

For example, in figure 13 we can see the the artifacts we met while working on radio interferometric data. Therefore it is common to search for optimized parameters of the method 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, we have decided 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 method to achieve a satisfactory balance between recall and purity involves carefully selecting the Astrodendro parameters from the observed field and instrument properties. Then, a refinement process is implemented to enhance the overall detection quality.

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 according to the pixel size and the beam of the instrument:

$$min\_npix = 2 \times \frac{\text{beam\_size}}{\text{pix\_size}} \tag{20}$$

where beam_size is the size of the beam, and pix_size is the size of one pixel in the sky. With the RACS data, we have a typical beam_size = 25 arcsecs, pix_size = 2.5 arcsecs, so we obtain *min_npix* = 20 pixels. With the LoTSS data, we have beam_size = 6 arcsecs, pix_size = 1.5 arcsecs, so we obtain *min_npix* = 8 pixels.

The parameter *min_delta*, the step of each iteration, is chosen from the instrument flux sensitivity. This information can be found in the datasheet of the instrument or, for RACS, in McConnell et al. (2020). I take triple the value find as the *min_delta*, for instance with RACS has a sensitivity of 0.25 mJy/beam, which leads to a *min_delta* value of 0.75 mJy/beam. For the LoTSS data, we have *min_delta* = 0.3 mJy/beam.

The parameter *min_value*, the minimum flux value of the pixel taken into account, its value is chosen based on the pixel value's distribution for a specific field. This value is derived from the flux distribution of the pixel in the studied mosaic. It is taken as the 90th percentile of the distribution, for the RACS it is in the order of magnitude of 0.7 mJy/beam. For the LoTSS data, it is in the order of magnitude of 0.1 mJy/beam.

3.2.2 Purity enhancement

If we apply the classical detection with the parameter described previously, we notice that there is a considerable amount of detection that can be considered doubtable. Our objective is to limit the false detections as much as possible, mostly from artifacts in the images. For that purpose, we filter the detections based on two criteria.

The first criterion is the eccentricity of the detection. The artifacts have a rather elongated shape, which means that some false detections have a rather large eccentricity. We decided to remove the detections which have an eccentricity above 0.9.

The second criterion is based on the localization of the artifacts in the field. The majority of artifacts are around bright sources and the extent of their distribution increases with the brightness of the source. To remove such detections, we use a rejection radius around bright sources, which is a function of their flux:

$$\mathrm{R}_{RACS} = 1365 \times e^{\log_{10}(flux)} - 3650$$

$$\mathrm{R}_{LoTSS} = 4 \times e^{\log_{10}(flux)} + 2$$

those functions were empirically derived from our observations on the global flux's distribution in the catalogs produced by Astrodendro as well as the typical radius within which we can find the artifacts around bright sources. To avoid removing potential true detection, we don't remove sources that have an integrated flux above 10 mJy.

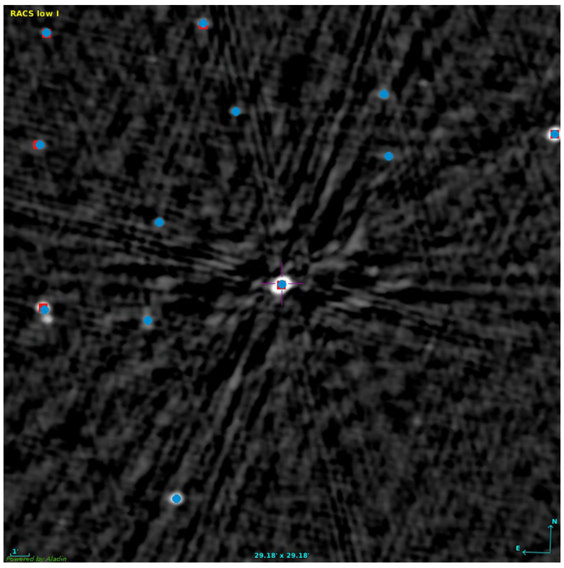
3.3 Result analysis

I have applied this method to the same 20 mosaics from the RACS data used for our ML detection, and I have reapplied this method to 800 of the 841 LoTSS mosaics to see the change from the old version of the pipeline.
To evaluate the performance of the detection by Astrodendro, we will compare the catalog obtained with the source catalog of the RACS and the LoTSS with a nearest neighbor cross match, using the same metric described previously, equation 18 and 19, with $\mathrm{N}_{\mathrm{net}}$ replaced by $\mathrm{N}_{\mathrm{dendro}}$, the number of sources detected by astrodendro. We are using a 25 arcseconds separation for the RACS cross-match and a 6 arcseconds separation for the LoTSS cross-match.
On the RACS data, the methods held ~60% recall with ~90% precision in the detection. On the LoTSS data, it held an average $\sim 70 \pm 10\%$ recall with $\sim 50 \pm 23\%$ precision. For the best fields from both surveys, the fields with a high signal-to-noise ratio and with few artifacts present in the field, the method can reach peak performance with ~70% recall and ~90% precision. Example fields are shown in figure 12.

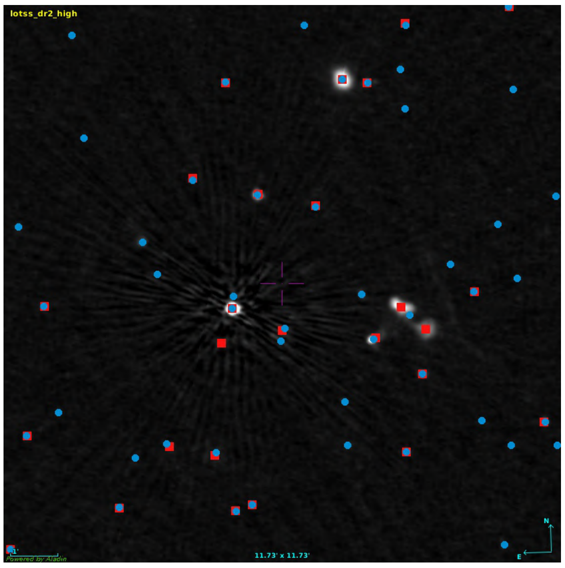
During my previous internship, I performed this classical detection on the same LoTSS field resulting in an average recall of $\sim 60 \pm 15\%$ and an average precision of $\sim 62 \pm 8\%$ in the detection. With the modification brought during this LIU, the average recall raised by around 10 points with a smaller dispersion while the precision decreased by around 10 points but with a much higher dispersion. This comes from the fact that the method produces fewer sources in the fields of better quality (precision and high recall). However, in low-quality fields, the number of sources tends to be higher (high recall but low precision), whereas the old version could sometimes make almost no detection, which is not wanted.

The results from the LoTSS detection here are an average over the analyzed mosaic, contrary to the results from the RACS detection on which I have implemented the same kind of process described in section 2.3.2 to deal with mosaic overlaps and redundant detection over overlapping mosaics. It is planned very soon to relaunch detection on LoTSS with this implementation and the missing mosaics.

While our astrodendro-based method shows satisfactory precision in detection, we want to avoid limiting false detections as much as possible false detections, therefore enhancing recall. Additionally, the goal is to keep as many true detections as possible, even those eliminated based on our filtering criteria. Most of the undetected sources correspond to very low integrated flux sources or sources rejected in areas with artifacts. On the other hand, the false detections come either from AGN-like sources, astroendro has the same trouble cataloging AGN with jets as the network, or they come from particularly bright artifacts that have an integrated flux above the limit set at 10 mJy.



(a) Example field of 29.18’× 29.18’ from the RACS.



(b) Example field of 11.73’× 11.73’ from the LoTSS

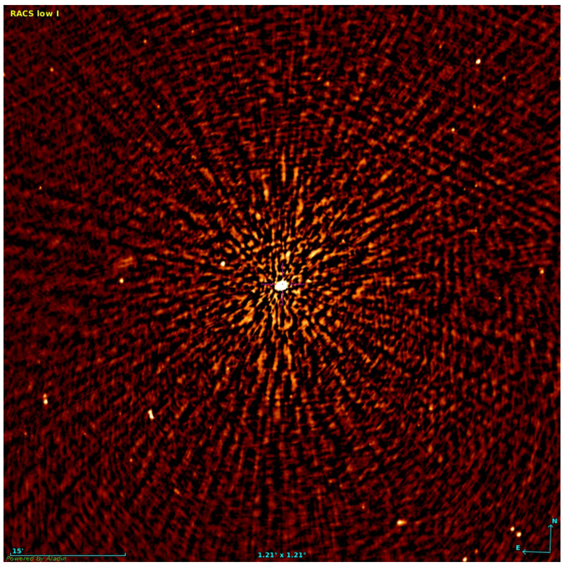
Figure 12: Example fields that show the results of the classical detection on the RACS and the LoTSS.he squares in red represent the detection with our astrodendro method. The circles in blue represent the reference catalog.

Although we can find algorithmic ways to improve the results, we reach a limit where radio data is no longer sufficient to improve the quality of our training catalog. To increase the quality of the training dataset, a perspective we have is the use of existing catalogs in other wavelengths. The idea is to try to reintroduce sources eliminated by the processing while keeping aside the eliminated sources whose parameters are credible to be existing sources. We then proceed to a cross-match with optical/IR counterparts in the region. The matched sources can then be reintegrated into our catalog, thus increasing its purity. We could also remove AGN’s jet detection from known AGN with jet catalogs.

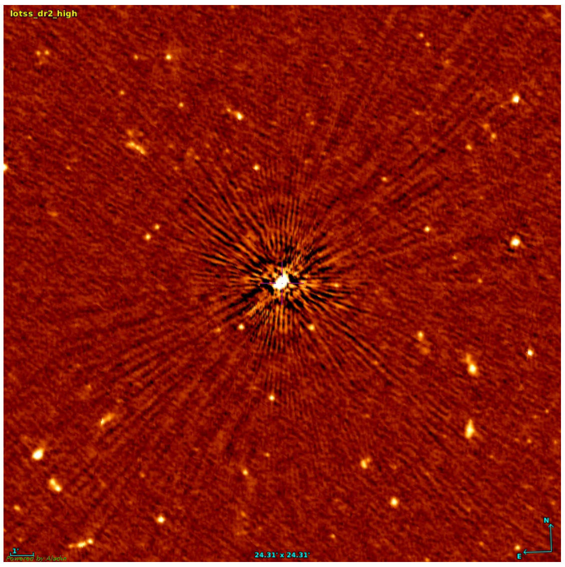
4 Discussion and conclusions

In this report, I have presented the results of applying the MINERVA CNN-based detection method to large radio-continuum surveys from SKA precursors, LoTSS, and RACS. I showed that the direct application of a network trained on the simulated data from the SDC1 to the LoTSS and RACS data presents behaviors that lead it to underperform in some specific contexts. I presented different post processes that proved efficient in improving the quality of the solution by removing false detection. On the RACS data, we are reaching ∼54% recall with ∼65% precision in our ML detection with the RACS catalog taken as reference. Although the detection achieves satisfactory performance, the network cannot manage all the specificities of the surveys. While some of these specificities represent real instrument limitations, some false detection or missed detection could be identified by visual inspection, which is a strong indication that a network structure should be capable of improving the current result with the proper complementary training. As could be expected, the training on only the simulated data from the SDC1 is not sufficient to represent all the features present in the real data. This is why it is necessary to apply specific training to the network, to make it learn the typical features present in the studied survey.
To perform this training, we are constructing complementary training datasets for both surveys. Those training datasets are derived from the classical method of the dendrograms. We are reaching $\sim 70 \pm 10\%$ and ∼60% recall with $\sim 50 \pm 23\%$ and ∼90% precision in our classical detection respectively for the LoTSS and the RACS. However, we are reaching the limited quality of our training catalogs that we can obtain using only radio data. To reach the expected quality for training the network, we project to use counterparts in other wavelengths to maximize the number of true sources in our training data and have the best representation of the radio sources in the surveys.

5 Annex



(a) 1.21°× 1.21° section of the RACS field centered on a bright source with many interferometric artifacts.



(b) 24.31'× 24.31' section of the LoTSS field centered on a bright source with many interferometric artifacts.

Figure 13: Examples of LoTSS and RACS fields with large interferometric artifacts displayed in Aladin software. The artifacts are the lines distributed around the central sources, which can span several degrees.

References

A. Bonaldi, T. An, M. Brüggen, S. Burkutean, B. Coelho, H. Goodarzi, P. Hartley, P. K. Sandhu, C. Wu, L. Yu, M. H. Zhoolideh Haghighi, S. Antón, Z. Bagheri, D. Barbosa, J. P. Barraca, D. Bartashevich, M. Bergano, M. Bonato, J. Brand, F. de Gasperin, A. Giannetti, R. Dodson, P. Jain, S. Jaiswal, B. Lao, B. Liu, E. Liuzzo, Y. Lu, V. Lukic, D. Maia, N. Marchili, M. Massardi, P. Mohan, J. B. Morgado, M. Panwar, P. Prabhakar, V. A. R. M. Ribeiro, K. L. J. Rygl, V. Sabz Ali, E. Saremi, E. Schisano, S. Sheikhnezami, A. Vafaei Sadr, A. Wong, and O. I. Wong. Square kilometre array science data challenge 1: analysis and results. *Monthly Notices of the Royal Astronomical Society*, 500(3):3821–3837, Oct. 2020. ISSN 1365-2966. doi: 10.1093/mnras/staa3023. URL http://dx.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.

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

P. Hartley, A. Bonaldi, R. Braun, J. N. H. S. Aditya, S. Aicardi, L. Alegre, A. Chakraborty, X. Chen, S. Choudhuri, A. O. Clarke, J. Coles, J. S. Collinson, D. Cornu, L. Darriba, M. D. Veneri, J. Forbrich, B. Fraga, A. Galan, J. Garrido, F. Gubanov, H. Håkansson, M. J. Hardcastle, C. Heneka, D. Herranz, K. M. Hess, M. Jagannath, S. Jaiswal, R. J. Jurek, D. Korber, S. Kitaeff, D. Kleiner, B. Lao, X. Lu, A. Mazumder, J. Moldón, R. Mondal, S. Ni, M. Önnheim, M. Parra, N. Patra, A. Peel, P. Salomé, S. Sánchez-Expósito, M. Sargent, B. Semelin, P. Serra, A. K. Shaw, A. X. Shen, A. Sjöberg, L. Smith, A. Soroka, V. Stolyarov, E. Tolley, M. C. Toribio, J. M. van der Hulst, A. V. Sadr, L. Verdes-Montenegro, T. Westmeier, K. Yu, L. Yu, L. Zhang, X. Zhang, Y. Zhang, A. Alberdi, M. Ashdown, C. R. Bom, M. Brüggen, J. Cannon, R. Chen, F. Combes, J. Conway, F. Courbin, J. Ding, G. Fourestey, J. Freundlich, L. Gao, C. Gheller, Q. Guo, E. Gustavsson, M. Jirstrand, M. G. Jones, G. Józsa, P. Kamphuis, J.-P. Kneib, M. Lindqvist, B. Liu, Y. Liu, Y. Mao, A. Marchal, I. Márquez, A. Meshcheryakov, M. Olberg, N. Oozeer, M. Pandey-Pommier, W. Pei, B. Peng, J. Sabater, A. Sorgho, J. L. Starck, C. Tasse, A. Wang, Y. Wang, H. Xi, X. Yang, H. Zhang, J. Zhang, M. Zhao, and S. Zuo. Ska science data challenge 2: analysis and results. *Monthly Notices of the Royal Astronomical Society*, 523(2):1967–1993, May 2023. ISSN 1365-2966. doi: 10.1093/mnras/stad1375. URL http://dx.doi.org/10.1093/mnras/stad1375.

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

D. McConnell, C. L. Hale, E. Lenc, J. K. Banfield, G. Heald, A. W. Hotan, J. K. Leung, V. A. Moss, T. Murphy, A. O'Brien, J. Pritchard, W. Raja, E. M. Sadler, A. Stewart, A. J. M. Thomson, M. Whiting, J. R. Allison, S. W. Amy, C. Anderson, L. Ball, K. W. Bannister, M. Bell, D. C.-J. Bock, R. Bolton, J. D. Bunton, A. P. Chippendale, J. D. Collier, F. R. Cooray, T. J. Cornwell, P. J. Diamond, P. G. Edwards, N. Gupta, D. B. Hayman, I. Heywood, C. A. Jackson, B. S. Koribalski, K. Lee-Waddell, N. M. McClure-Griffiths, A. Ng, R. P. Norris, C. Phillips, J. E. Reynolds, D. N. Roxby, A. E. T. Schinckel, M. Shields, C. Tremblay, A. Tzioumis, M. A. Voronkov, and T. Westmeier. The rapid askap continuum survey i: Design and first results. *Publications of the Astronomical Society of Australia*, 37, 2020. ISSN 1448-6083. doi: 10.1017/pasa.2020.41. URL http://dx.doi.org/10.1017/pasa.2020.41.

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. *CoRR*, abs/1612.08242, 2016. URL http://arxiv.org/abs/1612.08242.

J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018. URL http://arxiv.org/abs/1804.02767.

J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. URL `http://arxiv.org/abs/1506.02640`.

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`.