Observatoire de Paris - PSL

M2 Space Science and Technology: Internation Research Track

Internship report: Detection and characterization of galaxies with deep learning in radio data

Author:
Adrien ANTHORE

Supervisor:
David Cornu

M2 Internship

LERMA UMR 8112, Paris

July 20, 2024

Abstract

Today's interferometers produce large and complex data that increasingly challenge classical source detection methods. With the upcoming Square Kilometre Array (SKA), the astronomical community is looking for new innovative methods to perform this task. Members of the MINERVA project at the Paris Observatory have developed a new deep-learning approach to source detection and characterization that has demonstrated state-of-the-art performance on simulated radio continuum images from the SKA first data challenge (SDC1). The objective now is to demonstrate that the method can be applied to observed data with the same efficiency, thus preparing the analysis of the upcoming SKA data. In this report I present a way to directly apply the detection method trained on simulated data to the observational data from LOFAR that is already capable of detecting the most obvious sources with satisfactory confidence. To further improve these results, I constructed a complementary high-confidence training dataset based on a classical model detection method and performed transfer learning on the detector. A catalog of 11,105,581 sources was derived from the LOFAR field with 90% recall and 50% precision when compared to the catalogs derived by classical methods on these data.

Abstract

FR: Les interféromètres actuels produisent des données volumineuses et complexes qui posent de plus en plus de problèmes aux méthodes classiques de détection des sources. Avec l'arrivée prochaine du Square Kilometre Array (SKA), la communauté astronomique est à la recherche de nouvelles méthodes innovantes pour accomplir cette tâche. Les membres du projet MINERVA à l'Observatoire de Paris ont développé une nouvelle approche d'apprentissage profond pour la détection et la caractérisation des sources qui a démontré des performances de pointe sur des images simulées de continuum radio provenant du SKA first data challenge (SDC1). L'objectif est maintenant de démontrer que la méthode peut être appliquée aux données observées avec la même efficacité, préparant ainsi l'analyse des prochaines données du SKA. Dans ce rapport je présente comment appliquer directement la méthode de détection entraînée sur des données simulées aux données d'observation de LOFAR, ce qui permet déjà de détecter les sources les plus évidentes avec une confiance satisfaisante. Pour améliorer encore ces résultats, j'ai construit un ensemble de données d'entraînement complémentaire à haute confiance basé sur une méthode classique de détection et j'ai effectué un apprentissage par transfert sur le détecteur. Un catalogue de 11 105 581 sources a été dérivé du champ LOFAR avec une complétude de 90% et une précision de 50%, en utilisant comme référence les catalogues dérivés par des méthodes classiques sur ces données.

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

Contents

1 Introduction

Radio data have a wide range of astrophysical applications at all scales, from the inner Solar System to very high redshifted radiations. At the extragalactic scale, it provides unique information on the properties of galaxies and their evolution or the origin of large structures through the study of the cosmic dawn and reionization. These advances in the study of the large structures of the Universe are made possible by the emergence of new giant interferometers (e.g., LOFAR, ASKAP, MeerKAT, ...). These interferometers aim to achieve very high sensitivity and resolution, allowing them to place new constraints on the early stages of the Universe and better constrain the evolution of astronomical objects over cosmological timescales. Today's interferometers already produce large, complex, and high-dimensional data at the PB scale. These data are complex by nature and are characterized by various features, including instrumental effects and compound morphologies. The raw radio data are a set of complex *visibilities*, the direct Fourier transform of this data gives the dirty image. To obtain the final image, one must perform a deconvolution of the dirty image, which can be done by an algorithm, such as CLEAN (Högbom, 1974), which is most of the time optimized for point-like sources. This process aims to remove noise from the image, but due to various limitations, artifacts remain in the data. After calibration of the data, the product obtained is illustrated in the figure 1.

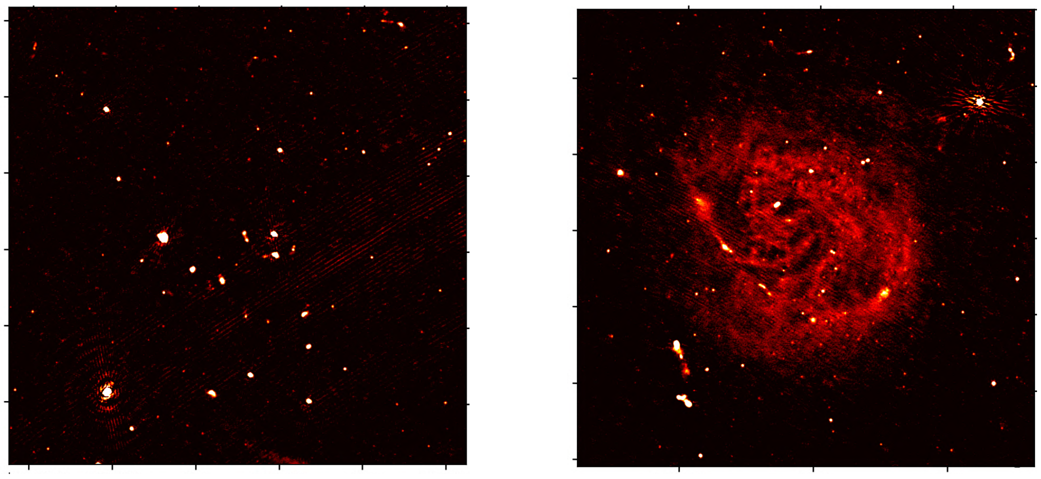

Figure 1: 0.5°×0.5° cutouts from the LOFAR field. The figure shows on the left frame a crowded field with a majority of point-like sources, and on the right frame a field centered on the diffuse extended emission from M101. The pixel values are normalized surface brightness with an arbitrary unit.

With the construction of the Square Kilometre Array (SKA) in 2028, radio data will further increase in volume and complexity. The latest projections for the SKA[1] show a drastic change in the scale of the data, with 700 PB of archived data per year and a raw data output of around 1 TB per second. To help prepare the astronomical community for handling such large data volumes, the SKAO (SKA Observatory) has launched a series of Data Challenges (SDC) to provide simulated data products that should be representative of the SKA. These challenges aim to compare analysis tools on controlled datasets and to stimulate 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. On the one hand, the SDC1 consists of 9 2D continuum images of the same 5°×5° field at 3 frequencies (560 MHz, 1.4 GHz, 9.2 GHz) and 3 telescope integrations (8, 100, 1000 h), each image being 4 GB in size. The SDC2, on the other hand, consists of a 3D HI emission cube over a sky area of 20 square degrees and is 1 TB in size. It has been simulated with a telescope integration of 2000 h with a bandwidth of 950-1150 MHz, sampled at a resolution of 30 kHz. This corresponds to a redshift interval of z = 0.235-0.495.

Source detection is a common task in astronomy and is typically the first analysis performed on a newly acquired image product. In radio data, this task is usually performed with different classical methods,

[1]https://www.skao.int/en/explore/big-data/362/ska-regional-centres

such as SExtractor (Bertin and Arnouts, 1996) or PyBDSF (Mohan and Rafferty, 2015). Those methods exhibit significant limitations, because of the data complexity and diversity. It is common to find false detections in the produced catalogs due to instrumental artifacts that look like small or elongated sources, or complex morphologies that are either single sources or multiple blended sources. Those cases are illustrated in figure 2.

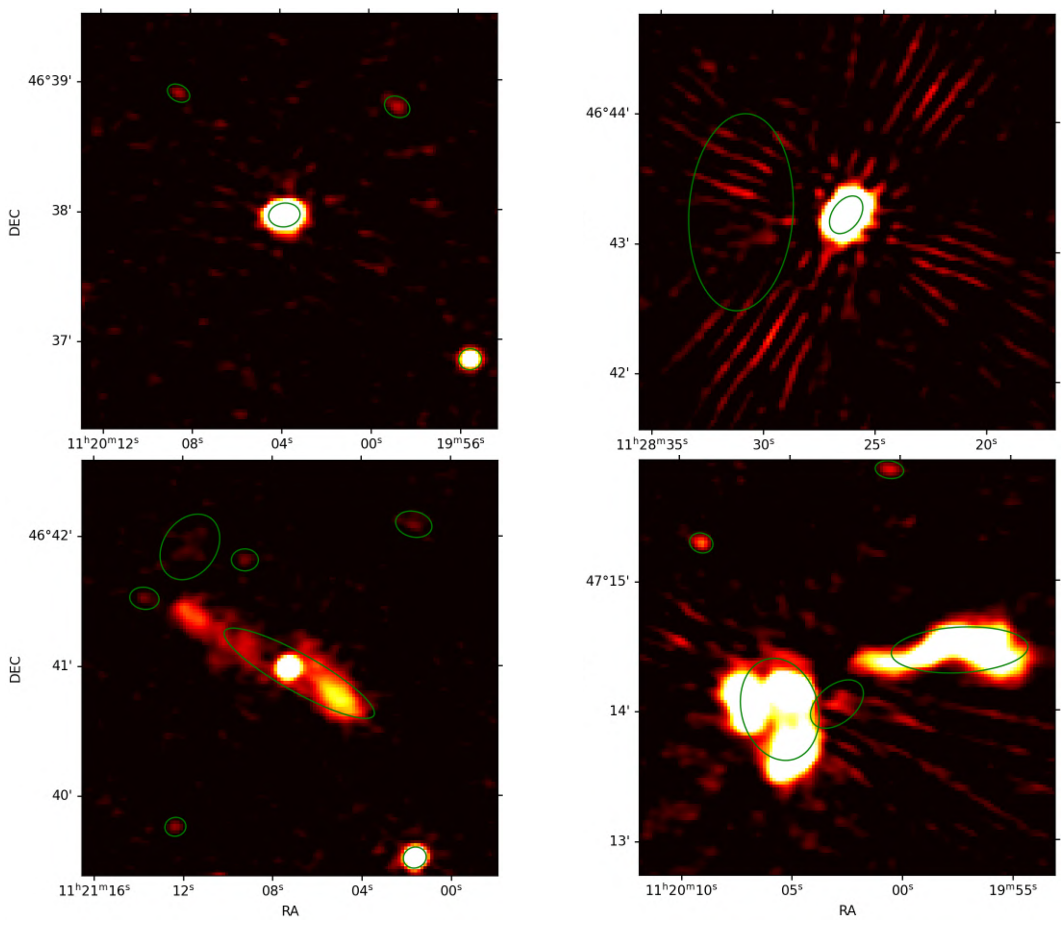

Figure 2: $3.2' \times 3.2'$ close up on typical detections case in LOFAR data. The detections from Shimwell, et al. (2022) are the green ellipses. The figure shows in the top left frame the simple case with few point sources and no strong artifacts; in the top right frame small sources with bright artifacts around; in the bottom left an ANG with jets; and in the bottom right frame a blended emission. The pixel values are normalized surface brightness with an arbitrary unit.

Nevertheless, these widely adopted classical methods have the advantage of being easy to interpret because they are deterministic and simple rules built using our physical knowledge about the objects and their emission in radio frequencies. They make it possible to extract the properties of the objects detected directly, and they make it possible to produce segmentation maps on the data.
With the considerable increase in the quantity of data, classical methods are challenged and become less effective, struggling to scale with this growth. For this reason, new innovative analysis methods need to be developed in parallel with the SKA deployment, which is possible thanks to the SDCs.

To participate in SDC2, members of the MINERVA (MachINe lEarning for Radioastronomy at obserVatoire de PAris) project at the Paris Observatory developed a deep learning approach for source detection and characterization that would be applied to both 2D and 3D data sets, YOLO-CIANNA (Cornu et al., 2024). This challenge has proven to be particularly difficult, as only nine of the forty international teams registered for the SDC2 achieved a positive result, i.e., a detection purity of more than 50% based on the challenge matching criteria. With its new method, the MINERVA team achieved first place on the leader-board, by detecting 30% more sources than the SoFIA (Serra et al., 2015) team that was using a state-of-the-art classical tool for HI detection while preserving the same purity of 95%. Based on this success, the MINERVA team also applied its method to the previous SDC1 dataset and detected more than twice the number of sources identified by the best solution submitted at the time of the challenge while still preserving a purity of 94%. Combined with a more accurate source characterization, it resulted in a score improvement of +139% using the challenge metric. In addition to its detection and characterization performances, the method is also computationally efficient as it produces detection results in under 1% of the corresponding integration time required to produce the corresponding data.

The simulated data attempt to approximate real data products from instruments, but they remain biased by the quality of the models, and their realism may be questionable. Therefore, the direct application of the method to observational data is not trivial and requires considering the properties of the different instruments to which we wish to generalize the method. The team is interested in generalizing the method to 2D radio continuum images from LOFAR (LOw Frequency ARray, van Haarlem et al. (2013)) data, specifically the LoTSS (LOFAR Two-metre Sky Survey). These data are representative of the specificity and complexity of contemporary radio data, as illustrated in figure 3.
This internship aims to build an effective implementation of the YOLO-CIANNA method on the LoTSS images. This work involves the construction of automatic and efficient pipelines for inference, the construction of a labeled training set specific to LoTSS data, and large deep-learning detector training.

This report consists of two main parts. In the 2 section, I will present MINERVA's deep learning method for source detection and characterization in the context of the SDC and its direct application to observational data. In the 3 section, I will present our method for improving the results of the direct application with a specification of the method. Eventually, I'll discuss the results in the 4 section, highlighting the progress of the project and the prospects for improving the detections.

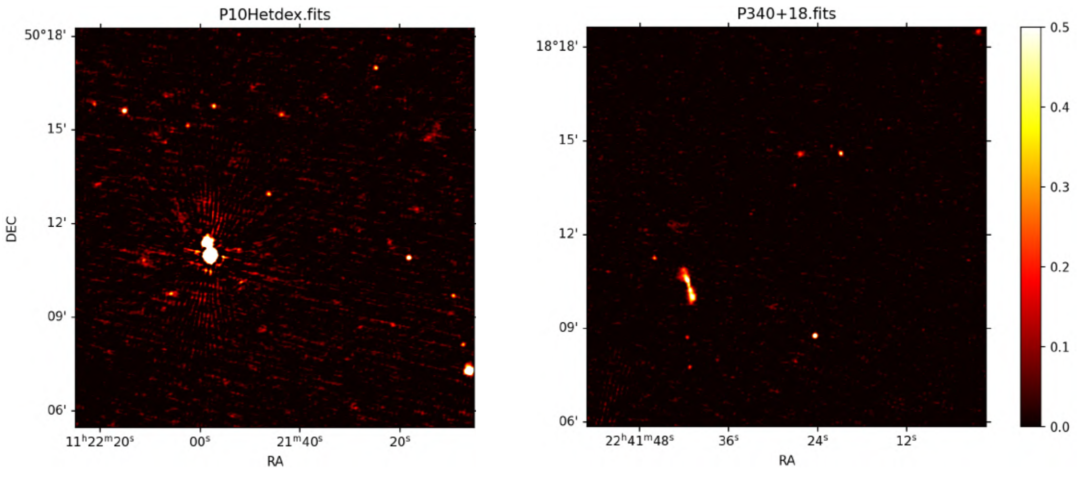

Figure 3: Example fields of $12.8' \times 12.8'$ from the LoTSS DR2 data. This figure shows an example field with instrumental flaws on the left frame and a field with few artifacts on the right frame. The pixel values are normalized surface brightness with an arbitrary unit.

2 Pure deep learning approach

2.1 Context and objectives

The goal of the method we present in this section is to identify and position sources in radio images and to extract their key characteristics, such as source size (major and minor axis), integrated flux, and position angle (PA). This type of detection has been performed several times for a large panel of instruments using classical methods. As mentioned in the Introduction (Sect. 1), the increase in volume and dimensionality of data is challenging for today's widely used method. In this context, the MINERVA team has developed its detection and characterization method using deep neural networks: YOLO-CIANNA

(Cornu et al., 2024). It is inspired by a method designed and developed for everyday images, typically the datasets Pascal VOC or COCO. However, these images often have implicit features. For instance, the presence of sharp-edged objects, masking of objects, low noise, limited dynamic range, lack of blending, and sufficient sampling for detectable objects. On the other hand, astronomical images may have considerable dynamic range, be noisy, contain complex and pronounced instrumental effects, have crowded areas, blended sources, insufficient sampling, or degenerate object types or classes. To adapt this method to astronomical images, the team made extensive structural modifications to account for the specificity of the astronomical source detection task. The specific method was implemented within CIANNA (Convolutional Interactive Artificial Neural Networks by/for Astrophysicists)[2] which is a general-purpose artificial neural network framework developed by David Cornu. The concepts behind machine-learning methods, and machine-learning for images are explained in Annex (Sect. 5.1). The following section focuses on the description of specific deep learning methods for object detection and, therefore, makes use of concepts such as supervised machine learning training, dense artificial neural network layer structure, and convolutional neural network layer structure for image analysis.

2.2 YOLO-CIANNA method

Object detection is a common task in computer vision, and it can be done in many different ways. In the past few decades, Deep Neural Networks (DNN) have been identified as one of the most efficient family of methods for this specific objective. In fact, there are different sub-types of suitable DNN approaches for object detection, which can be separated into two main groups that express the object detection task differently. The first group is composed of segmentation methods that aim to find all the pixels that belong to an object, usually by predicting masks at a resolution lower or equal to the input image resolution. The second group is composed of methods that predict bounding boxes for each object. In that second case, an object's bounding box is defined as the smallest possible rectangular region that contains all the pixels belonging to a given object in the image. Among the bounding box detection methods, we can distinguish two subgroups: multiple-stage methods (also called region-based), such as the R-CNN approach (Girshick et al., 2013), and single-stage methods (also called regression-based), the best known of which is YOLO (You Only Look Once, Redmon et al. (2015)). These different architectures have specific advantages depending on the context. Overall, regression-based methods are known for speed as they are widely used for real-time object detection. They also offer a more straightforward formalism to object detection than region-based or segmentation methods. The deep learning method developed by the MINERVA team was originally inspired by the YOLO architecture. David Cornu built the method 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. However, it now contains several tweaks and additions tailored to astronomical source detection that are not represented in any other of the widely adopted detection frameworks.

2.2.1 General method description

A classical YOLO network consists (from V2 and onwards) of a fully convolutional network (only composed of convolutional layers, see Sect. 5.1.2) in which a YOLO layer is added at the end. The role of the YOLO layer is to encode the network output as a regular grid that maps the different parts of an input image. During training, each grid cell is tasked to detect and characterize all the possible objects centered on that specific grid cell. The griding of the image is shown in the first frame of figure 6. The use of a fully convolutional architecture implies that each grid cell corresponds to an independent pixel in the output activation maps that is structurally centered on the corresponding input region.

For all grid cells, the characteristics of the detected boxes are stored as flat feature vectors. They predict values that allow the reconstruction of the bounding box's geometrical properties like the central coordinates of the boxes (x,y) and the sizes of the boxes (w, h). In practice, each grid cell must predict a

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

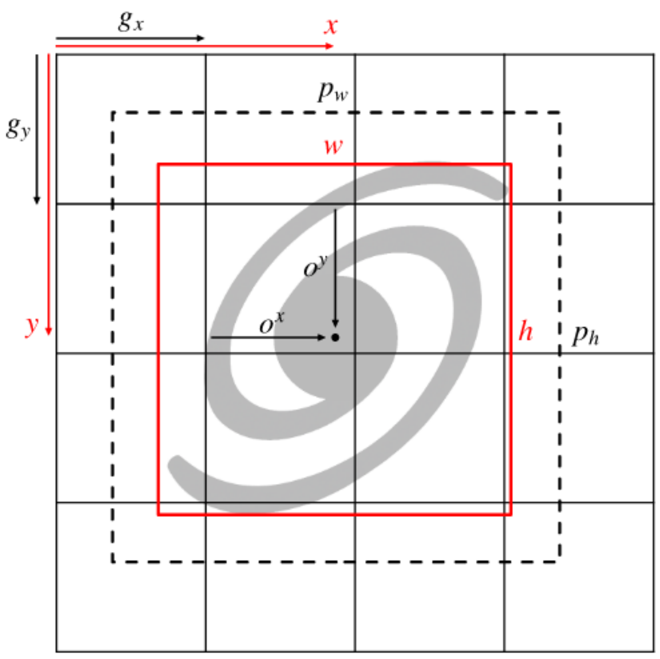

Figure 4: Illustration of the YOLO bounding box representation based on the output grid construction. The dashed black box corresponds to the theoretical prior, while the red box represents the network's predicted size. Source: Cornu et al. (2024)

4-element vector (o_x, o_y, o_w, o_h) from which we can derive the geometrical box properties as follows:

$$x = \sigma(o_x) + g_x, \tag{1}$$
$$y = \sigma(o_y) + g_y, \tag{2}$$
$$w = p_w e^{(o_w)}, \tag{3}$$
$$h = p_h e^{(o_h)}. \tag{4}$$

where (g_x, g_y) corresponds to the relative position of the current cell in the grid, and $(o_x,\ o_y)$ is the position inside the current 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 a set of predefined size priors (p_w, p_h) for each box. These priors need to be defined before the training based on the box size distribution in the training sample. These different geometrical elements are depicted in figure 4, 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. In addition to the geometric properties of the boxes, the output vector also predicts a detection score for each prediction. This quantity is defined as

$$O = P \times IoU, \tag{5}$$

where O is the objectness score, P is the probability that the predicted box contains an object, and the IoU (Intersection over Union) is a geometrical-based matching score in the range $[0,1]$ that estimates the proximity between the predicted and the target bounding boxes. The output vector can also encode the characteristic elements of a classification task, so every predicted box also outputs a class probability. Finally, in YOLO-CIANNA, the output vector has been modified to output an arbitrary number of linear parameters that can be used to estimate additional properties of the detected object, such as the flux of the detected astronomical sources. The typical format of a single output vector for one box is given in figure 5.

During the training phase, the association between the target boxes and the current prediction of the network is done mostly independently for each grid cell. During what is called the association process, the network will try to find the best current prediction corresponding to each target in the image based on an IoU score. Then, a loss function is used to act on the network parameter in a way that reduces the difference between the predicted parameters and those of the target (Mean Square error for the geometry

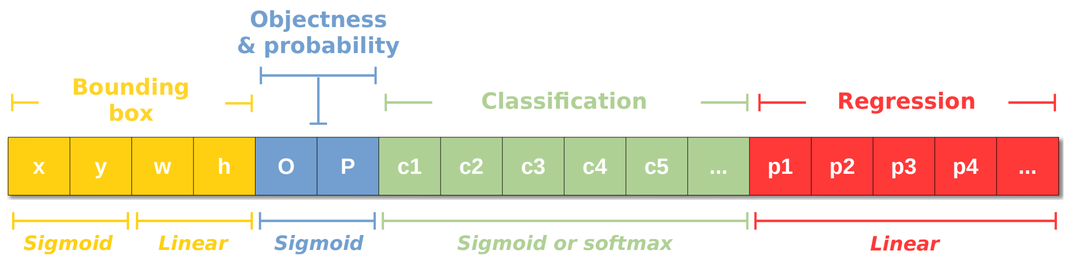

Figure 5: Output vector from a single grid output grid cell for a single detection unit. The elements are colored by type (geometry, probability, classification, and other properties), and the corresponding activation functions are indicated. Multiple identically shaped vectors are repeated on the same axis for multiple detection units. Source: Cornu et al. (2024)

and the regression parameters, and a cross-entropy for the classification parameters). Regarding the objectness score, the target value is defined from the IoU between the best predictions and their associated target, and their probability of representing an object is considered to be 1. For all other predicted boxes that are not the best prediction, only the objectness score is modified by setting the target objectness to 0.

At inference time, all the possible boxes in each grid cell are always predicted regardless of their chance to represent an object, but they can be filtered based on their objectness score. Then, 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 NMS algorithm works as follows: the box with the highest probability is selected and placed in a closed list, 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. The middle frame of Figure 6 shows the full list of predicted boxes with the thickness of the lines proportional to their objectness score. The right frame of this image shows the end result after both the objectness filtering and NMS post-process.

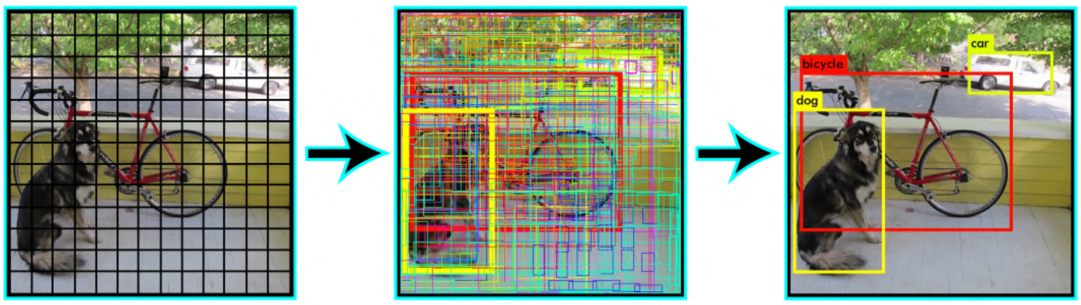

Figure 6: Representation of an image with the output grid and the detection process. The left frame is the input image on which the YOLO gris has been super-imposed. The middle frame is the input image on which all the prediction boxes are represented, the thickness of the box lines represents the score (thicker is higher). The right frame represents the best prediction boxes selected by the NMS. Credit: Redmon et al. (2015)

2.2.2 Specific setup for source detection

When applied to astronomical source detection in radio continuum data from the SDC1, the methods predict both the central position and the box size (Bounding box in figure 5) for each object but also a list of complementary parameters (Flux, PA, Bmaj, Bmin) that are predicted by the network as a regression (Regression in figure 5) 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 then 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)).

The model used by the MINERVA team in Cornu et al. (2024) is a fully convolutional network with 17 convolutional layers and a YOLO layer at the end. Figure 7 shows a schematic of the network. This architecture is the result of a combination of inspiration from different flavors of YOLO (V2, V3, lite, etc.) networks and specific knowledge regarding the properties of the source detection task, the corresponding input images, and instrumental properties (size of the objects, instrumental effects, noise levels, etc.). The final architecture was fine-tuned to optimize the SDC1 score. The detector predicts 9 independent boxes per grid cell, each characterized by size priors (p_w, p_h), which are parameters of the network.
This model demonstrated state-of-the-art performance, detecting more than twice as many sources as the best solution submitted, with a +139% score improvement in the SDC1 data (Cornu et al., 2024) as mentioned in the introduction (Sect. 1).

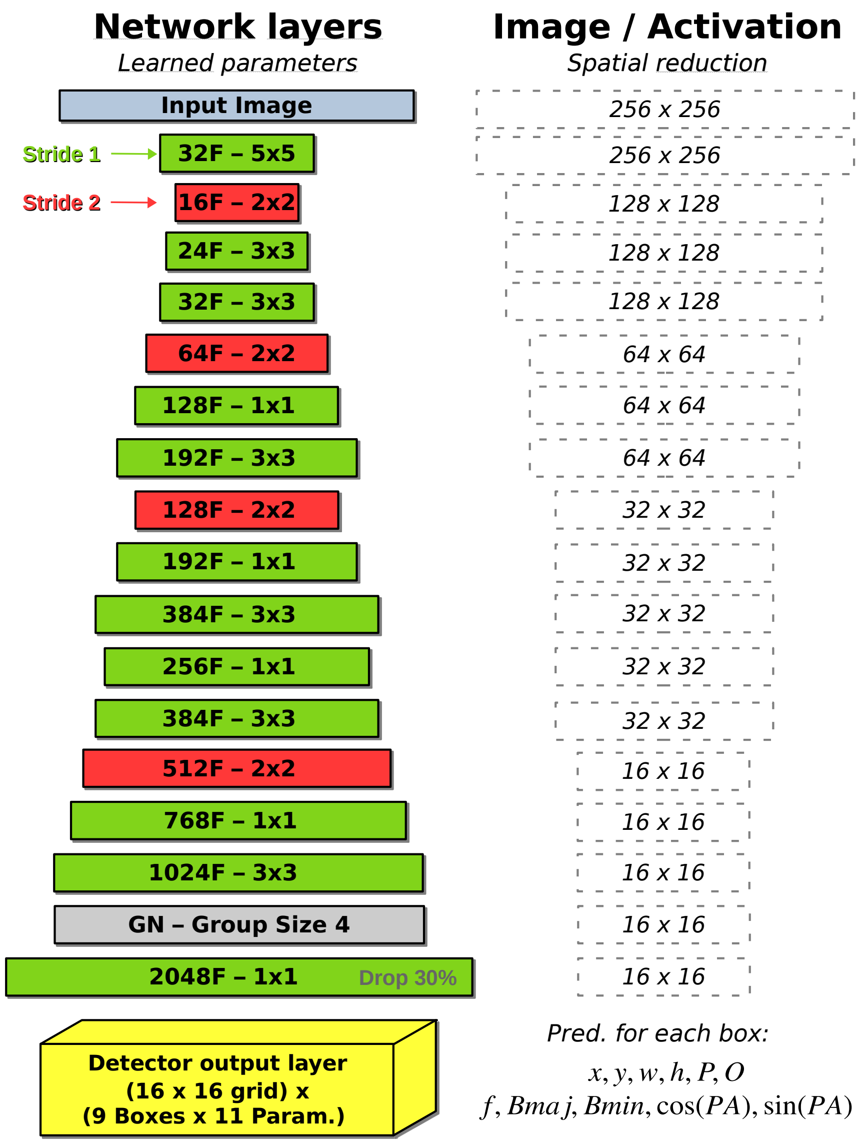

Figure 7: Schematic representation of our YOLO-based CNN. On the left is represented the layers with the number of filters and their size, and on the right side is represented the dimension of the output at each layer. The color represents the stride used, when the stride is equal to 1 (in green), there 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). Source: Cornu et al. (2024).

2.3 From simulations to observations

The first light of the SKA is planned for 2028, but there are a certain number of instruments already in operation that are "pathfinders" or "precursors". The SKA precursors are telescopes at the SKA sites testing technologies that will be used by the SKA telescopes (ASKAP, MeerKAT, MWA, and HERA). The SKA pathfinders are telescopes and systems that perform technology and science studies related to the SKA (LOFAR, NenuFAR, VLA, ALMA, ...). The current objective of the research group I joined during this internship is now to generalize the YOLO-CIANNA method to these precursor and pathfinder instruments.

To fulfill this objective, the best approach would be to use the detector architecture that worked best on simulated data (Fig. 7) and train it from scratch exclusively on observational data. However, in order to train on observational data, we would need to construct a complete, confident, and large enough labeled dataset to train the significant number of free parameters of our detector, which is a very difficult task. The larger the model, the larger the training sample size must be, otherwise the model will overtrain. Then, the network detector would likely miss rare cases that are not very common in the data but still exist. Moreover, if training the network requires already labeling the full survey with high confidence, it would defeat the very purpose of building an automatized detector. Nevertheless, observational data have the advantage of containing all the instrumental effects and observational characteristics that we seek to constrain in order to achieve a good detection purity.
On the other hand, we have available simulated radio continuum data offered by SKAO during the data challenges and on which the detector model has already been trained. With simulation models, we can virtually generate as many labeled examples as necessary to constrain statistical learning models and also compensate for the effects of rarities. However, they remain biased by the quality of their underlying physical and instrumental modeling. In this context, one of the best solutions is to consider that simulations can provide an efficient pre-training dataset for large models. Even if additional fine training on real data is still required to remove bias from the simulation, the amount of observed data that needs to be accurately labeled can be greatly reduced.

In the rest of this report, we will present how we tried to adapt the detection network to perform the detection in the LOFAR data. We will first present the results of a direct application of the trained detection network to the simulated data from the SDC1 to the LoTSS field. For this purpose, we will discuss how we transformed the LoTSS images so that their properties are closer to those of the SDC1 on which the network was trained. We will then present in section 3 our approach for building a high-confidence training sample on a subset of the LoTSS fields to further train the network on the specificities of this dataset, and present and discuss the corresponding prediction results.

2.3.1 Adapting the LOFAR data

LOFAR is a giant interferometer with over 50,000 antennas spread across Europe in 52 stations, with a "dense core" in the Netherlands. Among the scientific projects on LOFAR, we are interested in the extragalactic radio continuum survey LoTSS. The second data release, LoTSS DR2 (Shimwell, et al., 2022), consists of 6 arcseconds resolution images with a pixel size of 1.5 arcseconds, taken at 120-168 MHz and derived from 3451 h of the LOFAR High Band Antenna (HBA), covering 27% of the northern sky. This data is divided into 841 circular images or mosaics, that overlap.
From this data Shimwell, et al. (2022) was able to produce a catalog of 4,396,228 sources, the majority of which had never been detected at these frequencies. This was refined by Hardcastle et al. (2023) through visual inspection, optical/IR counterparts, and other processes, resulting in a catalog of 4,116,934 radio sources, of which 85% have optical or infrared identification and 58% have a good redshift estimate.

Now, in order to apply the network trained on the SKAO SDC1 to LoTSS data, we need to transform the LoFAR images so they resemble the training dataset. The SDC1 data comprises 9 continuum images corresponding to 560, 1400, and 9200 MHz, each declined in 8, 100, and 1000 h of integration times. The SDC1 image closest to what can be found in the LoTSS data is the set of 560 MHz images, but we will only focus on the 1000 h integration that results in the most crowded image. This image is a 5.5×5.5 degrees field at a 1.5 arcseconds resolution with a pixel size of 0.6 arcseconds. It contains more than five million sources among which Bonaldi et al. (2021) estimated that only around 758,000 sources are above the noise level by 5σ specifically for this frequency and integration time.

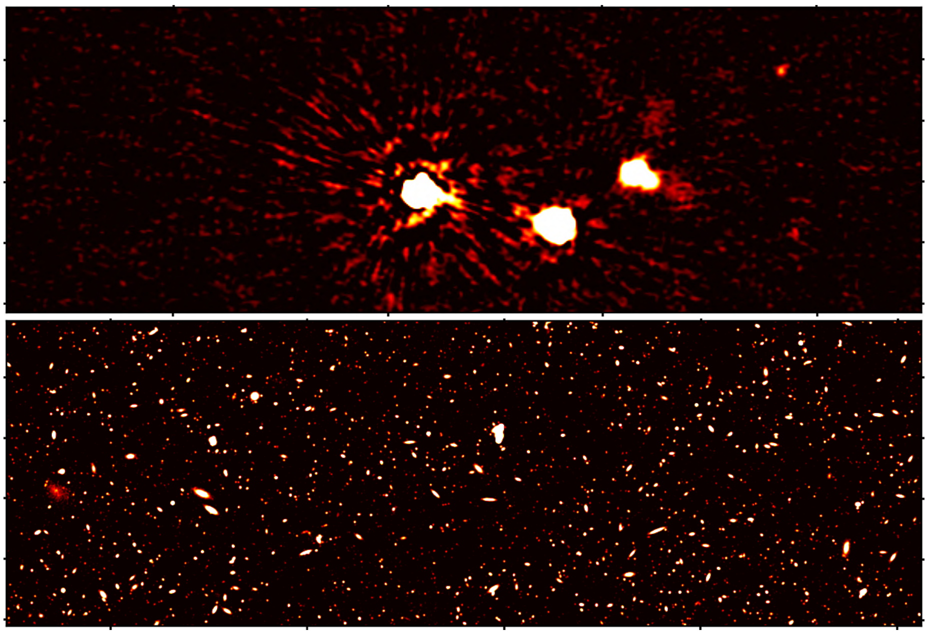

Figure 8: Comparative figure of the LoTSS field, on the top frame, and SDC1 field, on the bottom frame. The two images are $15' \times 5'$ subcuts. The pixel values are normalized surface brightness with an arbitrary unit.

To compare the data, we can start by looking at example fields. The fields illustrated in figure 8 have comparable aspects: the point-like sources are of the same type, the luminosity profiles are also of the same type for the extended sources, the number of sources per pixel is relatively close, and therefore the proportion of blending as well. However, there are some notable differences. To begin with, the instrumental characteristics are not the same (resolution, sensitivity, etc.), and the morphological diversity of the sources is not the same (proportion of point-like sources/resolved sources, plasma jets from the active nuclei of galaxies, etc.). Finally, the simulated SDC1 data poorly represents the typical characteristics of interferometry, for instance, the artifacts distributed in rays around sources with high surface brightness in the LOFAR images. Despite these differences, the data appear to be similar enough to apply the network. However, the detection method is based on priors defined with pixel size, and therefore the observations must be reduced to the same sampling. In SDC1, there are 2.5 pixels per resolved element in both directions, whereas in LoTSS, there are 4 pixels per resolved element in both directions. To obtain similar sampling in the two images, the LoTSS images need to be pooled by a factor of 2. This results in images with a pixel size of 3 arcseconds and 2 pixels per resolved element, which is close enough to the sampling of the SDC1 image.

The other important aspect is data normalization. The network takes as input a 2D image in which the pixels contain a surface brightness value with a very wide dynamic range. To enable the network to converge, these values need to be normalized to bring the dynamics of the image towards the dynamics of the initialization of the weights. Astronomical images have large dynamics that need to be preserved as much as possible. For the SDC1, the MINERVA team decided to preserve the 32-bit quantization of the input image and to give more representativity to low flux values (close to the noise value) via a hyperbolic tangent that compresses the high flux dynamics. These two choices were made to maximize the chances of detecting objects with low SNR. It is important to note that all images must be normalized with the same coefficients so that the flux can always be determined. If we normalized each patch by its maximum, then a pixel value of 1 would not represent the same flux in each image, making the network rely only on morphological detection criteria. The choice for the normalization is further discussed in Cornu et al. (2024) and the normalization can be summarized to this function:

$$p_i' = \tanh\left(\alpha \frac{p_i - min_p}{max_p - min_p}\right) \tag{6}$$

where p_i is the raw pixel value in Jy.beam^{-1} clipped between the values min_p and max_p, respectively the

limit below which the pixels necessarily contain noise, and above which the pixels necessarily contain signal, and finally α a multiplicative factor. In the case of transfer learning to LoTSS, the network was trained with a certain distribution of pixel values specific to the SDC1 image. Therefore, it is necessary to renormalize the LoTSS images so that this distribution is as close as possible to that of SDC1 in order for the network to produce a detection catalog that is both relatively complete and not dominated by false detection. We also have to take into account that the observing conditions on LoTSS change depending on the mosaics. Taking fixed values for the whole LoTSS can lead to incorrect normalization of the data. For the SDC1, the normalization parameters are: $\alpha = 3$, $min_p = 1 \times 10^{-7}$ Jy.beam^{-1}, and $max_p = 4 \times 10^{-5}$ Jy.beam^{-1}.
By minimizing the absolute difference between the distribution of the normalized data from the SDC1 and the LoTSS, we found that we can derive the normalization parameters for each LoTSS field with $\alpha = 3$, min_p as the 85th percentile of the pixel flux value, and $max_p = 100 min_p$. This results in the distribution in figure 9. We note that by doing so, we lose the ability of the network to predict the flux of the sources in a way that would be comparable between one mosaic to the other. However, for now, we only seek to evaluate the raw detection capabilities of this direct transfer.

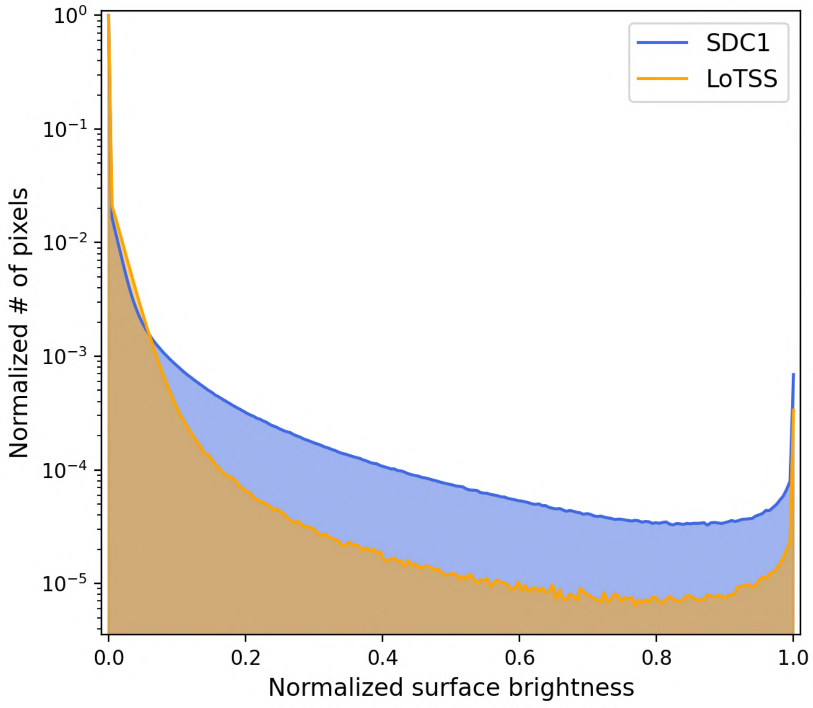

Figure 9: Normalized distributions of the pixels per values of the normalized surface brightness of the SDC1 and LoTSS images. These distributions have been made by taking equivalent subcutis of $1° \times 1°$ in each field. The blue envelope corresponds to the distribution of the pixels from the SDC1 image, and the orange envelope the distribution of the pixels from the LoTSS image.

Now that we have done this analysis and that we have processed the LoTSS images to be correctly interpreted by the network, we can move on to the inference on the entire LoTSS survey.

2.3.2 Direct application

The direct application of the network is the most straightforward approach to applying our network to observational data. We expect that the similarities between LoTSS and SDC1 data will allow most sources to be detected. However, we also expect that features not present in SDC1, such as instrumental artifacts, will produce a considerable amount of false detections. I will use the field shown in figure

3 to illustrate the detection. Our detection will be compared to the PyBDSF detection from Shimwell, et al. (2022), illustrated in figure 10. We evaluate the quality of our detection through its recall (the completeness of the detection), and its precision (the purity of the detection). The recall is defined as:

$$\text{recall} = \frac{\mathrm{N_{match}}}{\mathrm{N_{ref}}} \tag{7}$$

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

$$\text{precision} = \frac{\mathrm{N_{match}}}{\mathrm{N_{net}}} \tag{8}$$

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

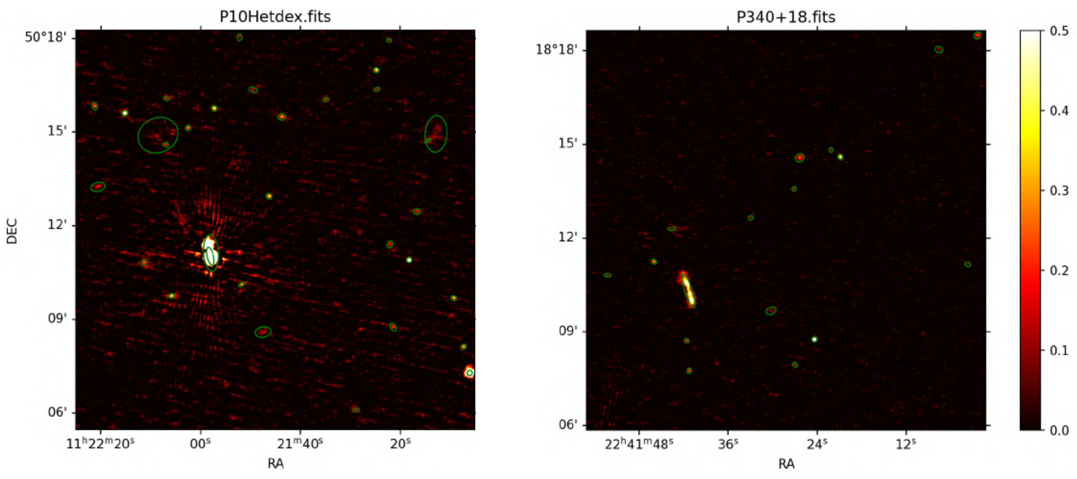

Figure 10: Example fields of $12.8' \times 12.8'$ from the LoTSS DR2 data. The green ellipses correspond to the detection produced by Shimwell, et al. (2022) with PyBDSF. The pixel values are normalized surface brightness with an arbitrary unit.

When we proceed with the prediction, we obtain the detection shown in figure 11. As expected, there are a large number of false detections that come from the instrumental artifacts in the field. However, the network can detect what appears to be real sources in the fields with proper bounding boxes. Another visible problem with these detections is that some sources that appear to be unique are detected by more than one simple box, further degrading the quality of the detection. This detection results in $\sim$50% recall and $\sim$20% precision.

We can see that the false detections are mainly concentrated around bright sources due to artifacts resembling compact or elongated sources. It is relatively easy to distinguish these artifacts from true sources by human inspection, as they have a characteristic pattern around the bright sources. We take into account the context, which the network cannot do as this type of example was missing from its training dataset (SDC1 data). However, CNNs have been shown to perform similarly to human experts in most visual recognition tasks. It would therefore be sufficient to correctly label these cases for the network to learn them. We may remove some of the false detections by post-processing the predictions. The majority of easily identifiable false detections are found at artifact levels around sources with high surface brightness. I, therefore, search these sources and apply a rejection radius around it fixed at an average distance from the extension of these artifacts ($\sim$5 arcmin). I based the withdrawal of the predictions on the objectness to probability ratio of the boxes. The objectness is computed from the product of the probability score and the IoU of the box. The probability simply refers to whether the box contains signals, but some false detections such as artifacts have a brightness very close to a real source and therefore are easily considered as signals by the network. By dividing the objectness by the probability, we obtain the intrinsic geometric quality of the detection, the lower this score is, the less relevant the box is as a prediction and therefore the more likely it is to be a false detection. This process leads to the predictions shown in figure 12. On the first hand, if we look at the field with no artifacts, the prediction remains the same. On the other hand, if we look at the field with artifacts, we see that the area is less crowded. We maintained a

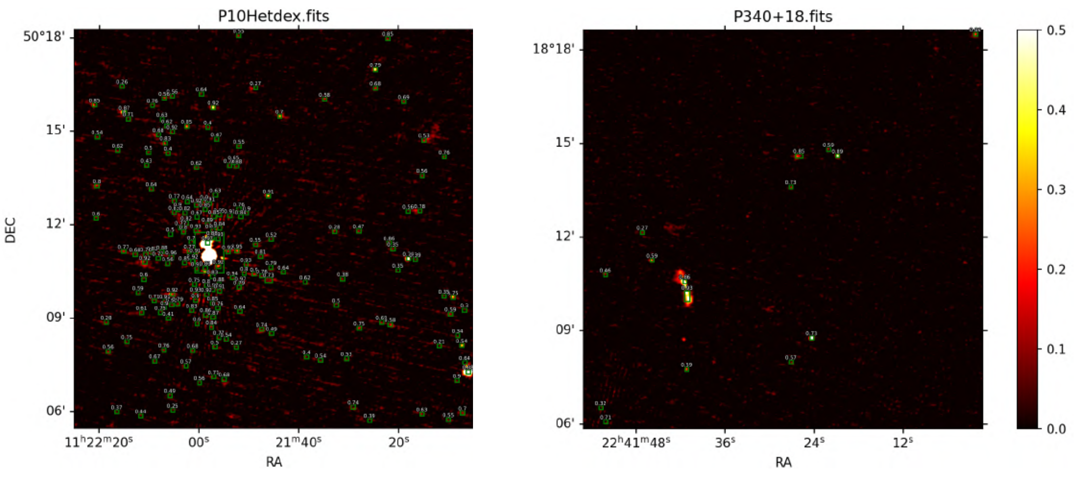

Figure 11: Example of detection in two fields of $12.8' \times 12.8'$ from the LoTSS DR2 data. The detection has been made with the YOLO-Cianna CNN trained on simulated data. The green boxes are the predictions of the network with their probability associated written in white. The pixel values are normalized surface brightness with an arbitrary unit.

reasonable level of detection, with all reliable sources detected, and were able to remove false detections from artifacts present in the field. However, we still have unique sources detected with more than one box. This results in an increase in precision, leading to ~45% recall and ~30% precision.

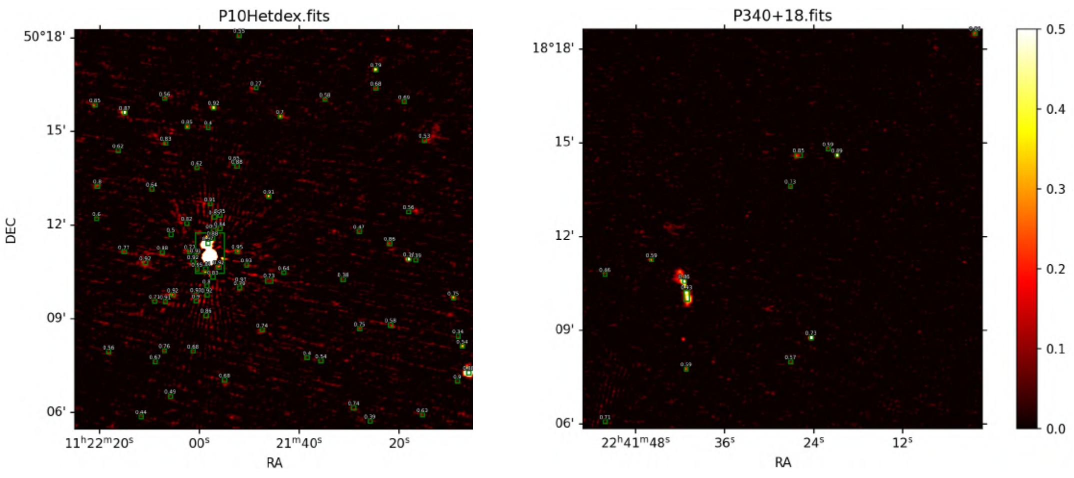

Figure 12: Example of detection in two fields of $12.8' \times 12.8'$ from the LoTSS DR2 data. The detection has been made with the YOLO-Cianna CNN trained on simulated data and post-processed to remove false detections. The green boxes are the predictions of the network with their probability associated written in white. The pixel values are normalized surface brightness with an arbitrary unit.

The results described above suggest that the state-of-the-art model for simulated data can be generalized to perform detection on SKA precursors. However, the direct application of the network to observational data is not enough. Although it predicts reliable sources in the LoTSS field, there are too many false detections and mischaracterized sources. These results are encouraging and suggest that we can drastically improve detection with the proper training dataset.

3 Specific training approach

As mentioned in the previous section, the results suggest that with proper training on LoTSS data, our model (Fig. 7) could produce a higher or equivalent precision and completeness catalog than the current catalogs derived from classical methods.

3.1 Training methodology

In this subsection, I will describe how I chose the training area for the detection, as well as the methodology behind the labeling.

3.1.1 Training sample definition

To train the network, we need to show the detector all the representative objects present in the LoTSS field. We need to explore the LoTSS field to determine the most common elements and their density, which I have done through visual inspection. We find:

- "Bright sources with artifacts", which are sources with high surface brightness with multiple beams distributed around them;
- "AGN-like sources with jets", are multi-lobed sources that look like active galactic nuclei with emission jets or two mixed elliptical sources;
- "Extended unusual", are non-point-like sources with sometimes unusual morphologies, for instance: resolved galaxies or intra-galactic sources;
- "Long-range artifacts" are artifacts that look like those around bright sources, but don't come from a visible source in the current field of view;
- "Strong discontinuity, a sudden change in pixel dynamics in the current field of view;
- "Holes", holes in the data.

Those features are illustrated in figure 13, and the results of the inspection of 30 random regions of 1 degree^2 are reported in table 1.

Features	Statistics
Bright sources with artifacts	8$\pm$3 sources/degree^2
AGN-like sources with jets	3$\pm$2 sources/degree^2
Extended unusual	2$\pm$2 sources/degree^2
Long-range artifacts	10/30
Strong discontinuity	14/30
Holes	4/30

Table 1: List of the common features in the LoTSS field, illustrated in figure 13, established through human inspection of 30 frames of 1 degree^2 in the LoTSS field.

Once we have determined the characteristic features present in the LoTSS, we have to select a training area that covers all those features in a maximum of different contexts. We need an area large enough to include all these features, with a fine image quality, i.e. no rare flaws or objects that astrodendro can't characterize, to produce high precision and recall detection. Among the available images of the LoTSS, we found that the field P9Hetdex01, in figure 14, is a relevant representation of the LoTSS data. In this training area, during the training process, we show the network a 256$\times$256 pixels frame centered on a random position in the area following a uniform distribution.

Now that we have defined the training area, we have to consider the labeling of the sources of the training sample.

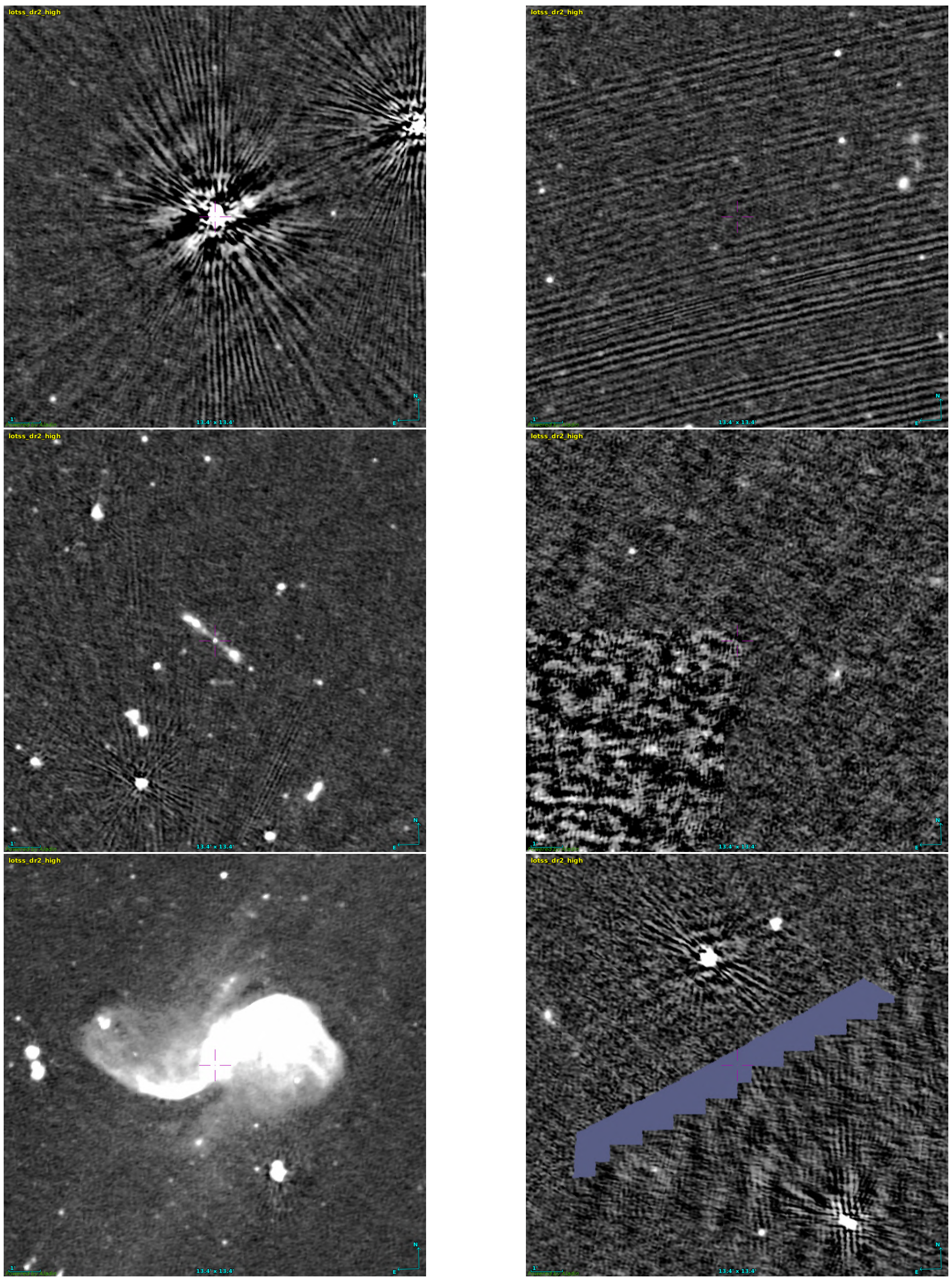

Figure 13: Cutouts of $13.4' \times 13.4'$ centered on the example of the typical features present inside LoTSS data. The top left frame is centered on a bright source with artifacts; the top right frame is centered on long-range artifacts; the middle left frame is centered on an AGN-like source with jets; the middle right frame is centered on a strong discontinuity; the bottom left frame is centered on an extended unusual source, here M106; the bottom right frame is centered on a hole.

3.1.2 Labellisation method

The motivations for producing our labels to train on the LoTSS are that, firstly, we want to produce labels that are optimized for our ML detection purpose, we want to produce the most accurate and complete

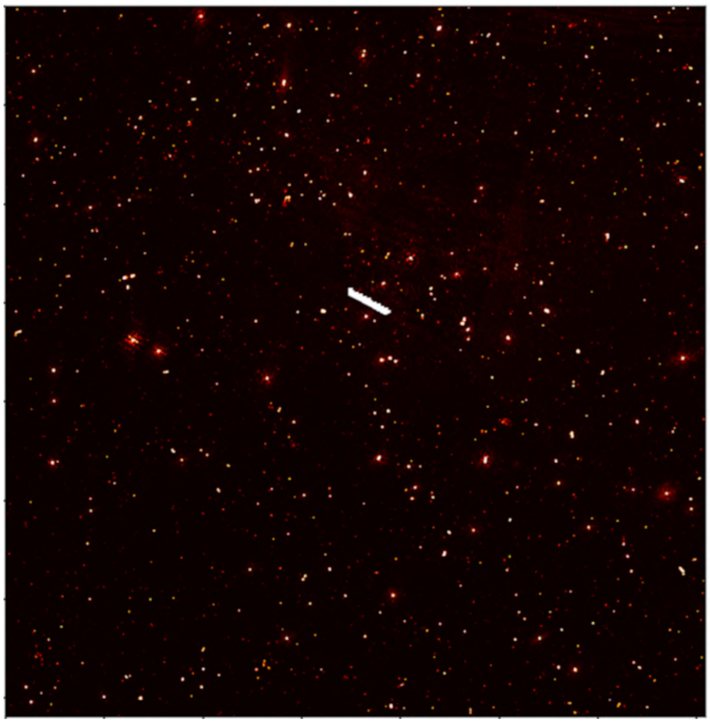

Figure 14: Training area used for the specific training of the YOLO-CIANNA method on LoTSS data. It consists of a 4.2°×3° subfield of the LoTSS data taken out of the mosaic P9Hetdex01. The pixel values are normalized surface brightness with an arbitrary unit.

detection possible to reproduce it on the full field. Second, we want to be able to compare the produced detection to the literature, otherwise, we would train the network to detect the same sources we are comparing to. The catalog released by Shimwell, et al. (2022) is based on the method PyBDSF, it creates a detection catalog from Gaussian fits on the field. The method we chose for source detection is based on the dendrogram algorithm. This gives us alternative labels that allow us to continue to compare ourselves with the literature.

Our method works as follows: Starting from the highest flux value, it is iterated downwards with a given step size until a selected baseline value is reached, as shown in figure 15. At each iteration, pixels are joined together to form structures depending on their proximity and flux level. The smallest structures are called *leaves*, and a group of nearby *leaves* form a *trunk*. We can choose the minimum number of pixels to form a *leaf*. For our detection, we only consider the *trunks* as sources.

In the astrodendro package, the parameters described above are implemented in the built-in detection method. The parameters are described as follows:

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

Every detection method will inevitably make false detections on this type of data. This is due to the artifacts in the field with enough local contrast to be identified as sources. For example, in figure 3 we can see the artifacts we encountered when working with radio interferometric data in the left frame. To build the training set, we seek to have the most reliable detections possible. To do this, we first optimize our choice of astrodendro parameters to obtain the most accurate detection possible.

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

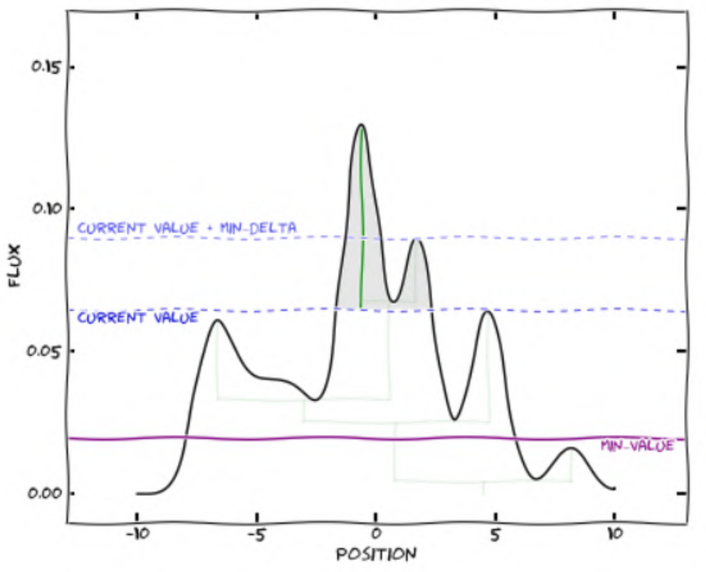

Figure 15: Illustration of the Astrodendro algorithm. - Source: https://dendrograms.readthedocs.io/en/stable/algorithm.html

the pixel size and the beam of the instrument:

$$min\_npix = 2 \times \frac{\text{beam\_size}}{\text{pix\_size}} \tag{9}$$

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.

The parameter *min_delta*, the step of each iteration, is chosen from the instrument flux sensitivity. The LoTSS sensitivity is 0.1 mJy/beam, thus 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 LoTSS data, it is in the order of magnitude of 10^{-2} mJy/beam.

However, this optimization does not remove all possible false detections, typically instrumental artifacts are still retained. Thus, we refine the detection by post-processing the catalog. We reject sources based on their proximity to bright sources, typically we define a rejection radius based on the integrated flux of the bright source considered. This rejection radius is defined as:

$$\text{R} = 12 \times e^{\log_{10}(flux)} + 6 \tag{10}$$

with R given in arcsecs, and $\log_{10}(flux)$ taken from the integrated flux of the source in mJy. We also reject sources based on their integrated flux ($<\sim$ 1 mJy), and their surface brightness ($<\sim 10^2$ mJy/arcsec^2). This greatly increases the accuracy of our detection, but some of the rejected sources may be true detections. With all this process, we reject $\sim$80% of the originally generated sources.
To increase the completeness of our correct detection, we perform cross-matches with catalogs at other wavelengths (i.e. DESI Duncan (2022), and ALLWISE Cutri et al. (2021)), thus recovering sources that were rejected even though they were true positives. The minimum distance to the sources to be tested from the reference catalogs was chosen to limit possible chance associations as much as possible. In our case, the order of magnitude is close to 1%. With this matching process, we re-integrate $\sim$30% of the rejected sources.
You can find an example of the training sample in figure 16. Despite the artifacts, we have a plausible detection, of which the most questionable sources have been confirmed with at least one of our reference catalogs.

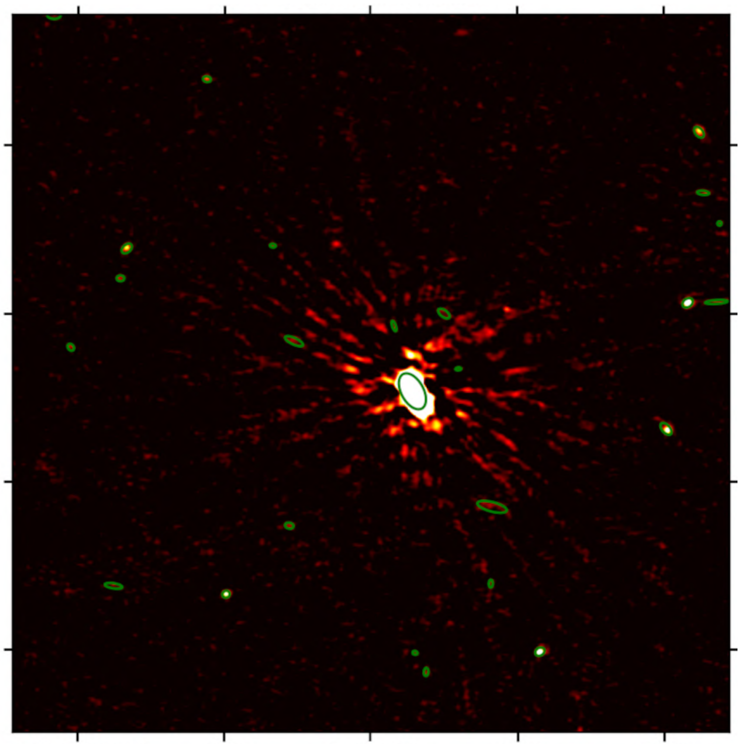

Figure 16: $12.8' \times 12.8'$ sub field in the training area (Fig. 14). The green ellipses correspond to the detections produced by Astrodendro, which are the labeled sources of our training dataset. The pixel values are normalized surface brightness with an arbitrary unit

The catalog produced in the training zone is a complete and precise catalog, with a density of detections comparable to what is established by Shimwell, et al. (2022). By comparing ourselves to this catalog, we obtain up to 70% recall and 80% precision. The next step is to train our detector with this data and carry out the inference on the LoTSS.

3.2 Transfer Learning application

The learning process of supervised methods is further described in Annex (Section 5.1.1). This is what we call the training phase, and a training phase conducted on all the training data is called an epoch. To know at what epoch we can stop training, we monitor the cost function that we try to minimize, and in our case, we can also look for the epoch at which the network is the most efficient compared to Shimwell, et al. (2022) on the training area.

The best performance of the network is shown at 2000 epochs. By testing the recall and precision compared to the training dataset, we obtained ∼80% recall and ∼40% precision. Compared to the Shimwell, et al. (2022) PyBDSF catalog, we achieve ∼90% recall and ∼50% precision. We can see the result in the training area in figure 17.

By comparing to the best previous result obtained with the direct application of the network and post-processing, in figure 12, we see in figure 17 a straight improvement in the network performance. Without any human intervention after the prediction, we see that the detector deals even better with artifacts than before, and the detection is even more complete in the case with low noise/artifacts. we also see that some sources have a better characterization than with direct application, with boxes that better frame the sources, especially fewer single sources characterized by more than one box.

This detection consists of the best performance we obtained through this process, resulting in a catalog of 11,105,581 radio sources all over the LoTSS field. Although this constitutes our best result, we think that we can still go further in the quality of detection.

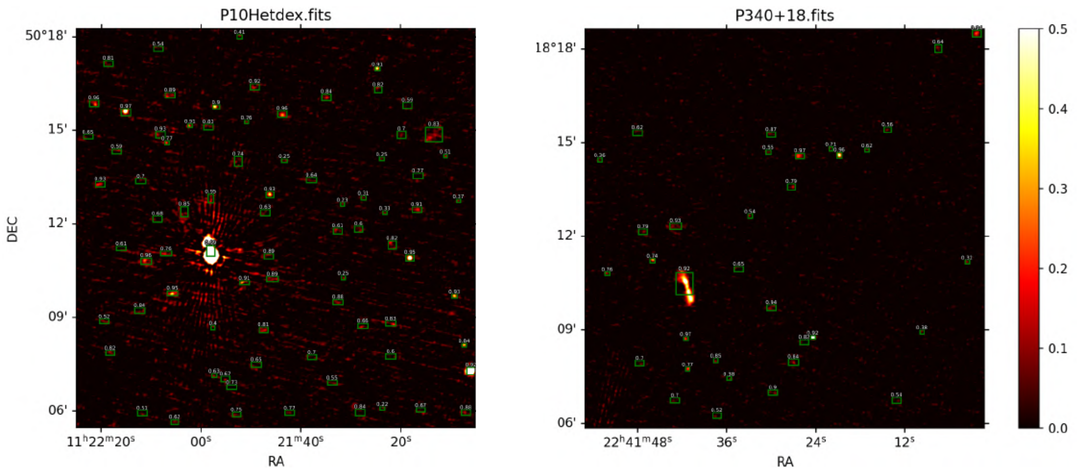

Figure 17: Example of detection in two fields of $12.8' \times 12.8'$ from the LoTSS DR2 data. The detection has been made with the YOLO-Cianna CNN trained on simulated data and the LoTSS data through a transfer learning process. The green boxes are the predictions of the network with their probability associated written in white. The pixel values are normalized surface brightness with an arbitrary unit.

Beyond the satisfactory detection performance itself, this method is much more efficient in terms of computation time than classical methods. To provide our training catalog, we need about 30 min per mosaic for detection and post-processing, and about 4 h for overlap management for the whole field, using publicly available resources (Intel Core vPRO i7 @ 2.50GHz), i.e. ~425 h for the whole LoTSS field, which means ~4 min/deg^2 or 12% of the observation time. With parallel computing, we can reduce this time down to ~50 h for the whole LoTSS field, thus ~30 s/deg^2 or 1% of the observation time. This computation time is underestimated, considering that I perform online queries from large databases and the connection may be disconnected by the query server. We do not have any information on the time needed to generate the Shimwell, et al. (2022) catalog, but considering that our method remains rather simple and fully automatic, we expect it to take at least as long.
Our YOLO-CIANNA method requires a GPU for training and inference, but post-processing is done on the CPU. For a Tesla V100 14.13 TFLOPS GPU, which is an entry-level GPU, the training process takes about 10 s/epoch, i.e. ~6 h for 2000 epochs. The inference process is very fast and is dominated by loading the fits files into memory. It takes about ~5 s per mosaic, with ~4 s for loading the fits file and ~1 s for the inference, which means ~1 h for the whole LoTSS field, i.e. ~0.5 s/deg^2 or 0.03% of the observing time. The construction of the catalog after the inference is longer and takes up to ~3 h, the computation time depends on the number of sources detected. Therefore, the whole process corresponds, in terms of observation time, to ~0.3% with training, and ~0.1% without training.

4 Discussion and conclusions

In this report, I presented the progress made in applying MINERVA's YOLO-CIANNA method to observational radio data. I have shown that the direct application of the detector, trained on simulated data, to LoTSS data, exhibits behaviors that lead it to underperform in some specific contexts. This detection can be improved by post-processing, but although this improves the results, the quality of the detection remains inadequate. With this process we achieve, thanks to post-process on the detection, ~45% recall with ~30% precision on the LoTSS with Shimwell, et al. (2022) PyBDSF catalog as a reference. Most of the false or wrong detections could be identified by visual inspection, which led us to think that by labeling these cases and retraining on these data, we could improve our results. Precisely, by labeling a subset of LoTSS and training our detector, we can improve detection and achieve a catalog of 11,105,581 sources at ~90% recall with ~50% precision, without post-process on the detection. In addition to the detection performance, this process is very fast and negligible compared to the integration time: ~0.3% of the integration time or ~6 s/deg^2. These performances are reached with fairly accessible computing

materials.
Although there are still errors in detection, we are getting closer to the catalog quality that is state-of-the-art for observational data. We need to continue to improve the training set to further reduce false detections with even higher recall compared to existing catalogs. Another important point for measuring the quality of our detection is the development of cross-match metrics between our proposals and the reference catalogs. Currently, our metric only evaluates the position of the recognition. To get a better evaluation and better matches, it is also necessary to take into account the characterization of the detection. We are in the process of developing a metric that gives a score based on the characterization, but there is still work to be done to tune the thresholds, limits, and scoring.

The general aim of this work is to generalize this approach to existing radio data, but in particular to prepare for the analysis of SKA data, for which powerful analysis methods are essential. Radio data from telescopes such as LOFAR, MeerKAR, and ASKAP have already been analyzed for object detection, and our detection method aims to match or exceed the quality of these catalogs, thus demonstrating its reliability for future data analysis. The first step is to complete the work on LOFAR, then extend it to continuum data, and finally to HI surveys. Perspectives will be added to this work, driven by the overall progress of this project. Typically, the improvement of simulations (T-RECS Bonaldi et al. (2018), ALMAsim Delli Veneri et al. (2022), ...), creating simulations with models very close to the instruments we study will allow a significant improvement in training without the need to label data. We also want to explore new network architectures to create models designed exclusively for astronomical data. Eventually, we seek to apply the method to non-radio surveys (JWST, Euclid, ...) and combine the information at different wavelengths, allowing an even more complete detection.
In addition to detection, the parameters derived from the radio and continuum data (size, flux, etc.) will help us to produce statistics on the detected galaxies. Typically, we are interested in the properties of the galaxies (dynamics, redshift, etc.) as well as their evolution. We are also interested in detecting special objects, such as the active nuclei of galaxies, whose radio emission is highly detectable, high redshift galaxies (z<6), for which instruments such as LOFAR have been designed, and galaxy clusters. To detect specific or "rare" objects, we may use unsupervised learning methods such as Astronomaly Lochner and Bassett (2021), which can extract objects of interest through clustering driven by the user.

5 Annex

5.1 ML methodology

The definition of Machine Learning (ML) 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.

5.1.1 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 then 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 , 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 processed object. 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:

$$h = \sum_{i=0}^{m} X_i \omega_i = b\omega_0 + \sum_{i=1}^{m} X_i \omega_i \tag{11}$$

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 neuron's output. We can 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 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):

$$a_j = g(h_j) = \begin{cases} h_j & \text{if } h_j \geq 0 \\ 0 & \text{if } h_j < 0 \end{cases} \tag{12}$$

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 18.

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 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

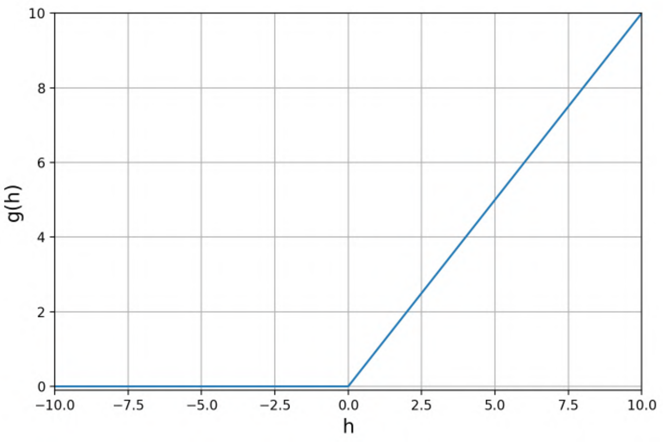

Figure 18: Representation of the ReLU activation function. The graph derives from the equation 12.

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

$$\omega_i \leftarrow \omega_i - \eta (a - t) X_i \tag{13}$$

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 19.

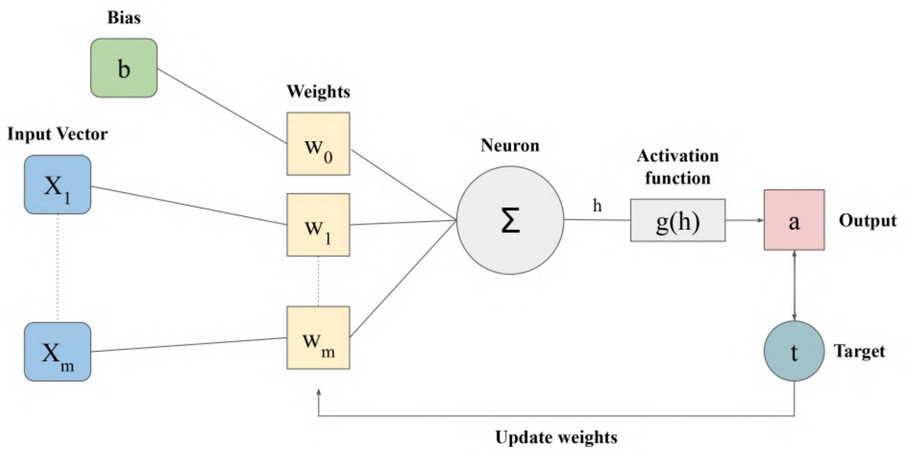

Figure 19: Schematic view of a binary neuron. $X_{[1,\dots,m]}$ are the input features for one object in the training dataset and b is the bias, $w_{[1,\dots,m]}$ are the weights associated with each feature and w_0 is the weight of the bias. Σ represents the sum function and h its results.

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 simultaneously from the same data. 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 become:

$$h_j = b\omega_{0j} + \sum_{i=1}^{m} X_i \omega_{ij} \tag{14}$$

and

$$a_j = g(h_j) = \begin{cases} h_j & \text{if } h_j \geq 0; \\ 0 & \text{if } h_j < 0. \end{cases} \tag{15}$$

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

$$\omega_{ij} \leftarrow \omega_{ij} - \eta(a_j - t_j)X_i \tag{16}$$

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 Cybenko (1989).

The addition of new layers in the network doesn't change much the forward processes, but the update of the weights described by equation (16) 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:

$$E(a,t) = \frac{1}{2}\sum_{k=1}^{N}(a_k - t_k)^2 \tag{17}$$

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 :

$$\omega_{ij} \leftarrow \omega_{ij} - \eta \frac{\partial E}{\partial \omega_{ij}} \tag{18}$$

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

$$\frac{\partial E}{\partial \omega_{ij}} = \delta_l(j)\frac{\partial h_j}{\partial \omega_{ij}} \quad \text{with} \quad \delta_l(j) \equiv \frac{\partial E}{\partial h_j} = \frac{\partial E}{\partial a_j}\frac{\partial a_j}{\partial h_j} = \frac{\partial a_j}{\partial h_j}\sum_k \omega_{kj}\delta_{l+1}(k) \tag{19}$$

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 (19) 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:

$$\frac{\partial h_j}{\partial \omega_{ij}} = a_i \tag{20}$$

the activation of the current layer,

$$\frac{\partial E}{\partial a_j} = (a_j - t_j) \tag{21}$$

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

$$\frac{\partial a_j}{\partial h_j} = \begin{cases} 1 & \text{if } h_j \geq 0 \\ 0 & \text{if } h_j < 0 \end{cases} \tag{22}$$

the derivative of the ReLU activation.

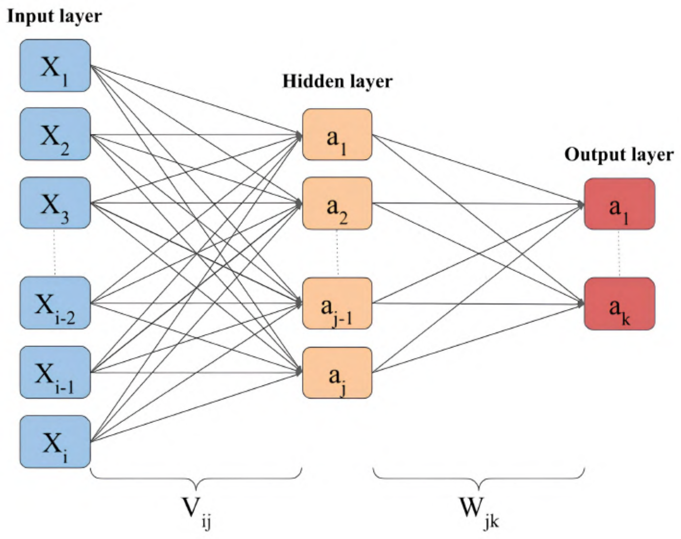

Figure 20: Schematic view of a simple "deep" neural network with only one hidden layer. The blue cells are input dimensions. The beige cells are the hidden layer dimensions. And the red cells are the output dimensions. $X_{[1,\ldots,i]}$ are the dimension for the input vector, $a_{[1,\ldots,j]}$ are 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.

5.1.2 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 21. 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 22. 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 :

$$w_{out} = \frac{w_{in} - f_s + 2P}{S} + 1 \quad h_{out} = \frac{h_{in} - f_s + 2P}{S} + 1 \tag{23}$$

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.

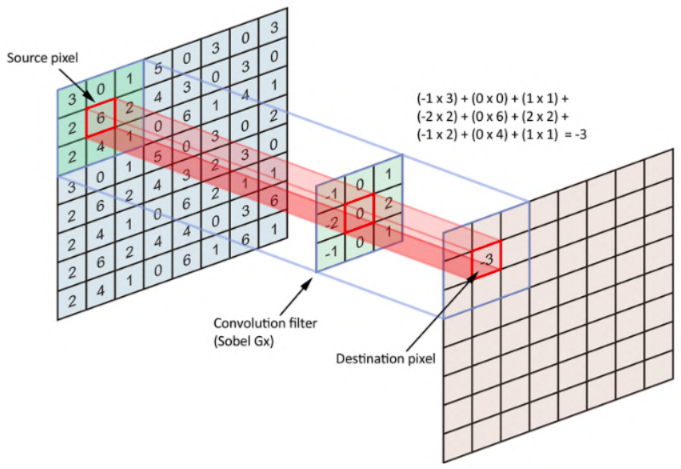
Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
(Sobel Gx)
Destination pixel

Figure 21: Schematic representation of a convolution operation. - Source: https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/

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 22.

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. As an example, a common architecture for those networks is the AlexNet architecture illustrated in figure 23. 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

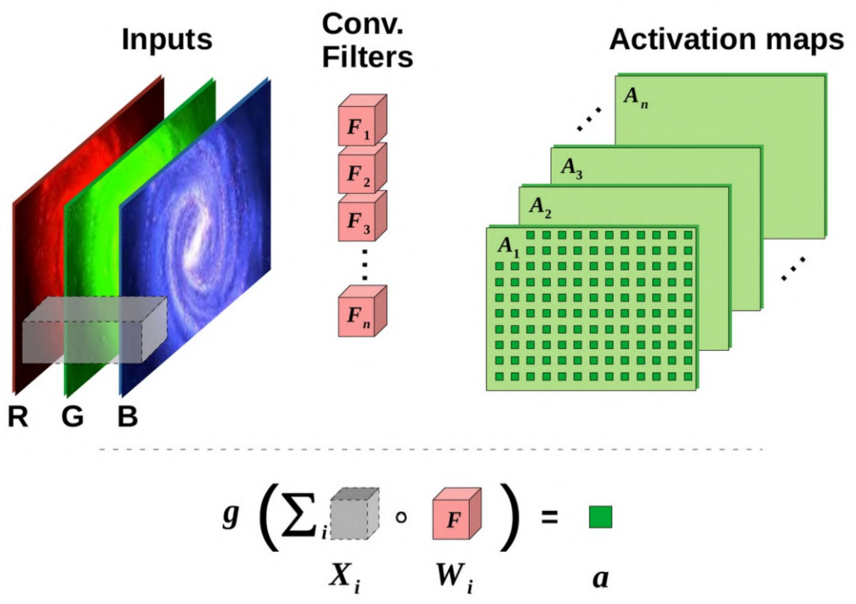

Figure 22: Schematic representation of a convolutional layer. The input consists of a three-channel image (RGB) with n filters. The output consists of n activation maps. - Source Cornu (2020)

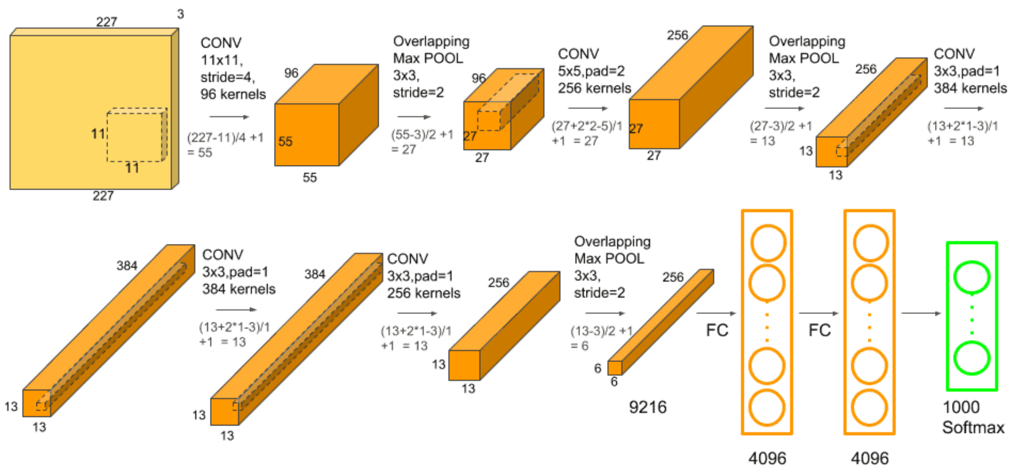

Figure 23: Illustration of the architecture of an AlexNet. It consists of 5 Convolutional Layers and 3 Fully Connected Layers. - Source: https://learnopencv.com/understanding-alexnet/

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 (19) 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.

References

E. Bertin and S. Arnouts. SExtractor: Software for source extraction. , 117:393–404, June 1996. doi: 10.1051/aas:1996164.

A. Bonaldi, M. Bonato, V. Galluzzi, I. Harrison, M. Massardi, S. Kay, G. De Zotti, and M. L. Brown. The tiered radio extragalactic continuum simulation (t-recs). *Monthly Notices of the Royal Astronomical Society*, 482(1):2–19, Sept. 2018. ISSN 1365-2966. doi: 10.1093/mnras/sty2603. URL http://dx.doi.org/10.1093/mnras/sty2603.

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.

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. , 500(3):3821–3837, Jan. 2021. doi: 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.

D. Cornu, P. Salomé, B. Semelin, A. Marchal, J. Freundlich, S. Aicardi, X. Lu, G. Sainton, F. Mertens, F. Combes, and C. Tasse. YOLO-CIANNA: Galaxy detection with deep learning in radio data. I. A new YOLO-inspired source detection method applied to the SKAO SDC1. *arXiv e-prints*, art. arXiv:2402.05925, Feb. 2024. doi: 10.48550/arXiv.2402.05925.

R. M. Cutri, E. L. Wright, T. Conrow, J. W. Fowler, P. R. M. Eisenhardt, C. Grillmair, J. D. Kirkpatrick, F. Masci, H. L. McCallon, S. L. Wheelock, S. Fajardo-Acosta, L. Yan, D. Benford, M. Harbut, T. Jarrett, S. Lake, D. Leisawitz, M. E. Ressler, S. A. Stanford, C. W. Tsai, F. Liu, G. Helou, A. Mainzer, D. Gettngs, A. Gonzalez, D. Hoffman, K. A. Marsh, D. Padgett, M. F. Skrutskie, R. Beck, M. Papin, and M. Wittman. VizieR Online Data Catalog: AllWISE Data Release (Cutri+ 2013). VizieR On-line Data Catalog: II/328. Originally published in: IPAC/Caltech (2013), Feb. 2021.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, Dec. 1989. ISSN 0932-4194. doi: 10.1007/BF02551274. URL http://dx.doi.org/10.1007/BF02551274.

M. Delli Veneri, Tychoniec, F. Guglielmetti, G. Longo, and E. Villard. 3D Detection and Characterisation of ALMA Sources through Deep Learning. *Monthly Notices of the Royal Astronomical Society*, 11 2022. ISSN 0035-8711. doi: 10.1093/mnras/stac3314. URL https://doi.org/10.1093/mnras/stac3314. stac3314.

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

K. J. Duncan. VizieR Online Data Catalog: DESI Legacy Imaging Surveys DR8 photometric redshifts (Duncan, 2022). VizieR On-line Data Catalog: VII/292. Originally published in: 2022MNRAS.512.3662D, June 2022.

R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 11 2013. doi: 10.1109/CVPR.2014.81.

M. J. Hardcastle, M. A. Horton, W. L. Williams, K. J. Duncan, L. Alegre, B. Barkus, J. H. Croston, H. Dickinson, E. Osinga, H. J. A. Röttgering, J. Sabater, T. W. Shimwell, D. J. B. Smith, P. N. Best, A. Botteon, M. Brüggen, A. Drabent, F. de Gasperin, G. Gürkan, M. Hajduk, C. L. Hale, M. Hoeft, M. Jamrozy, M. Kunert-Bajraszewska, R. Kondapally, M. Magliocchetti, V. H. Mahatma, R. I. J. Mostert, S. P. O'Sullivan, U. Pajdosz-Śmierciak, J. Petley, J. C. S. Pierce, I. Prandoni, D. J. Schwarz, A. Shulewski, T. M. Siewert, J. P. Stott, H. Tang, M. Vaccari, X. Zheng, T. Bailey, S. Desbled, A. Goyal, V. Gonano, M. Hanset, W. Kurtz, S. M. Lim, L. Mielle, C. S. Molloy, R. Roth, I. A. Terentev, and M. Torres. The lofar two-metre sky survey: Vi. optical identifications for the second data release. *Astronomy amp; Astrophysics*, 678:A151, Oct. 2023. ISSN 1432-0746. doi: 10.1051/0004-6361/202347333. URL http://dx.doi.org/10.1051/0004-6361/202347333.

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.

J. A. Högbom. Aperture Synthesis with a Non-Regular Distribution of Interferometer Baselines. , 15: 417, June 1974.

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

M. Lochner and B. Bassett. Astronomaly: Personalised active anomaly detection in astronomical data. *Astronomy and Computing*, 36:100481, July 2021. ISSN 2213-1337. doi: 10.1016/j.ascom.2021.100481. URL http://dx.doi.org/10.1016/j.ascom.2021.100481.

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.

N. Mohan and D. Rafferty. PyBDSF: Python Blob Detection and Source Finder. Astrophysics Source Code Library, record ascl:1502.007, Feb. 2015.

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.

P. Serra, T. Westmeier, N. Giese, R. Jurek, L. Flöer, A. Popping, B. Winkel, T. van der Hulst, M. Meyer, B. S. Koribalski, L. Staveley-Smith, and H. Courtois. SOFIA: a flexible source finder for 3D spectral line data. , 448(2):1922–1929, Apr. 2015. doi: 10.1093/mnras/stv079.

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.

M. P. van Haarlem, M. W. Wise, A. W. Gunst, G. Heald, J. P. McKean, J. W. T. Hessels, A. G. de Bruyn, R. Nijboer, J. Swinbank, R. Fallows, M. Brentjens, A. Nelles, R. Beck, H. Falcke, R. Fender, J. Hörandel, L. V. E. Koopmans, G. Mann, G. Miley, H. Röttgering, B. W. Stappers, R. A. M. J. Wijers, S. Zaroubi, M. van den Akker, A. Alexov, J. Anderson, K. Anderson, A. van Ardenne, M. Arts, A. Asgekar, I. M. Avruch, F. Batejat, L. Bähren, M. E. Bell, M. R. Bell, I. van Bemmel, P. Bennema, M. J. Bentum, G. Bernardi, P. Best, L. Bîrzan, A. Bonafede, A. J. Boonstra, R. Braun, J. Bregman, F. Breitling, R. H. van de Brink, J. Broderick, P. C. Broekema, W. N. Brouw, M. Brüggen, H. R. Butcher, W. van Cappellen, B. Ciardi, T. Coenen, J. Conway, A. Coolen, A. Corstanje, S. Damstra, O. Davies, A. T. Deller, R. J. Dettmar, G. van Diepen, K. Dijkstra, P. Donker, A. Doorduin, J. Dromer, M. Drost, A. van Duin, J. Eislöffel, J. van Enst, C. Ferrari, W. Frieswijk, H. Gankema, M. A. Garrett, F. de Gasperin, M. Gerbers, E. de Geus, J. M. Grießmeier, T. Grit, P. Gruppen, J. P. Hamaker, T. Hassall, M. Hoeft, H. A. Holties, A. Horneffer, A. van der Horst, A. van Houwelingen, A. Huijgen, M. Iacobelli, H. Intema, N. Jackson, V. Jelic, A. de Jong, E. Juette, D. Kant, A. Karastergiou, A. Koers, H. Kollen, V. I. Kondratiev, E. Kooistra, Y. Koopman, A. Koster, M. Kuniyoshi, M. Kramer, G. Kuper, P. Lambropoulos, C. Law, J. van Leeuwen, J. Lemaitre, M. Loose, P. Maat, G. Macario, S. Markoff, J. Masters, R. A. McFadden, D. McKay-Bukowski, H. Meijering, H. Meulman, M. Mevius, E. Middelberg, R. Millenaar, J. C. A. Miller-Jones, R. N. Mohan, J. D. Mol, J. Morawietz, R. Morganti, D. D. Mulcahy, E. Mulder, H. Munk, L. Nieuwenhuis, R. van Nieuwpoort, J. E. Noordam, M. Norden, A. Noutsos, A. R. Offringa, H. Olofsson, A. Omar, E. Orrú, R. Overeem, H. Paas, M. Pandey-Pommier, V. N. Pandey, R. Pizzo, A. Polatidis, D. Rafferty, S. Rawlings, W. Reich, J. P. de Reijer, J. Reitsma, G. A. Renting, P. Riemers, E. Rol, J. W. Romein, J. Roosjen, M. Ruiter, A. Scaife, K. van der Schaaf, B. Scheers, P. Schellart, A. Schoenmakers, G. Schoonderbeek, M. Serylak, A. Shulevski, J. Sluman, O. Smirnov, C. Sobey, H. Spreeuw, M. Steinmetz, C. G. M. Sterks, H. J. Stiepel, K. Stuurwold, M. Tagger, Y. Tang, C. Tasse, I. Thomas, S. Thoudam, M. C. Toribio, B. van der Tol, O. Usov, M. van Veelen, A. J. van der Veen, S. ter Veen, J. P. W. Verbiest, R. Vermeulen, N. Vermaas, C. Vocks, C. Vogt, M. de Vos, E. van der Wal, R. van Weeren, H. Weggemans, P. Weltevrede, S. White, S. J. Wijnholds, T. Wilhelmsson, O. Wucknitz, S. Yatawatta, P. Zarka, A. Zensus, and J. van Zwieten. LOFAR: The LOw-Frequency ARray. , 556:A2, Aug. 2013. doi: 10.1051/0004-6361/201220873.