Experimental Astronomy (2023) 56:293–331
https://doi.org/10.1007/s10686-023-09893-w

ORIGINAL ARTICLE

Radio astronomical images object detection and segmentation: a benchmark on deep learning methods

Renato Sortino[1,2] · Daniel Magro[1,3] · Giuseppe Fiameni[4] · Eva Sciacca[1] · Simone Riggi[1] · Andrea DeMarco[3] · Concetto Spampinato[2] · Andrew M. Hopkins[5] · Filomena Bufano[1] · Francesco Schillirò[1] · Cristobal Bordiu[1] · Carmelo Pino[1,2]

Received: 10 November 2022 / Accepted: 3 March 2023 / Published online: 5 May 2023

Abstract

In recent years, deep learning has been successfully applied in various scientific domains. Following these promising results and performances, it has recently also started being evaluated in the domain of radio astronomy. In particular, since radio astronomy is entering the Big Data era, with the advent of the largest telescope in the world - the Square Kilometre Array (SKA), the task of automatic object detection and instance segmentation is crucial for source finding and analysis. In this work, we explore the performance of the most affirmed deep learning approaches, applied to astronomical images obtained by radio interferometric instrumentation, to solve the task of automatic source detection. This is carried out by applying models designed to accomplish two different kinds of tasks: object detection and semantic segmentation. The goal is to provide an overview of existing techniques, in terms of prediction performance and computational efficiency, to scientists in the astrophysics community who would like to employ machine learning in their research.

Keywords Deep learning · Source finding · Object detection · Transformers · Astrophysics

✉ Renato Sortino
renato.sortino@inaf.it

1 Osservatorio Astrofisico di Catania, INAF, Via Santa Sofia 78, Catania 95123, Italy

2 Department of Electrical, Electronic and Computer Engineering, University of Catania, Catania, Italy

3 Institute of Space Sciences and Astronomy, University of Malta, Msida MSD2080, Malta

4 NVIDIA AI Technology Centre, Milan, Italy

5 Australian Astronomical Optics, Macquarie University, 105 Delhi Rd, North Ryde NSW 2113, Australia

1 Introduction

In recent years, technological advancement in astronomy and astrophysics has marked the need for innovative tools and techniques to process the huge amount of data and images captured from different instruments for radio astronomy observations [1, 2]. The several types of collected data (images, radio signals, etc.) need to be processed according to the specific tasks for extracting and evaluating useful information to support scientific research. In particular, in the context of radio-astronomical surveys, the task of object detection and instance segmentation is crucial for extracting information from images to support astrophysics research to catalog and identify the contained objects [3].

In contrast to optical instruments, which capture images of the sky's brightness distribution directly, radio interferometers employ interferometry to calculate the two-dimensional discrete intensity distribution of the sky, known as visibility data. A Fourier transform of the visibility data is then performed to produce an image of the sky. The result of this process is the convolution of the true sky brightness with the point spread function (PSF) of the interferometric array, commonly referred to as the dirty image. Due to the incomplete sampling of the interferometric visibility data, the PSF has strong spurious sources that affect the entire image. This can make it difficult to recover the true sky's brightness distribution from interferometric data [4].

Various approaches have been proposed to identify and extract visual information from images, but the majority of these methods are based on classical image processing techniques that show several limitations in terms of accuracy and classification and for other tasks that involve specific features post-analysis (e.g. for identifying extended sources or sources with a complex multi-component morphology).

To overcome these limitations, deep learning models represent the evolution of such approaches and yield interesting results extensively explored in several domains. In particular, object detection and semantic segmentation models based on deep learning are currently used in different domains, such as automotive [5–8], medical imaging [9–11], video surveillance [12, 13], and robot navigation [14–16]. The radio-astronomical domain has not yet been exhaustively explored with the application of the mentioned methods; therefore, this work represents an attempt to gather performance and computational requirements about several state-of-the-art approaches to be used as a reference for future work.

In this work, we propose a benchmark to evaluate and compare the performance of multiple object detection and semantic segmentation models based on deep learning (e.g. Mask-RCNN, U-Net, Tiramisu, etc.). We apply these models to astronomical radio images collected from several surveys to detect and classify sources and provide a comprehensive overview of these approaches.

We have performed tests on a dataset consisting of over 10, 000 images containing objects belonging to one of three classes (compact, extended, and spurious sources), extracted from different radio-astronomical surveys images taken with SKA precursors/pathfinders: the Australian Telescope Compact Array (ATCA), the Australian Square Kilometre Array Pathfinder (ASKAP) and the Very Large Array (VLA).

For each model, we evaluated the performance by calculating the F1 score, reliability (precision), and completeness (recall) for each detection. We also explored the

performance of subsets of our dataset according to the signal-to-noise ratio (SNR) i.e. the ratio between the peak luminous flux of the objects and the noise component of the image. This allows us to evaluate the detection abilities of the model on faint sources and with a varying degree of noise in the image.

The analysis conducted in this work shows state-of-the-art object detection and segmentation models applied to radio-astronomical images, providing a baseline for future work.

2 Related works

The earliest source detection technique was the visual inspection manually carried out by trained astronomers. Needless to say, with the sheer scale and volume of data from modern-day telescopes and surveys, such an approach is intractable and infeasible, as there is far too much data to be manually looked at by astronomers.

Algorithmic techniques represent the first form of automating the source finding task and include a variety of methods based on thresholding and peak detection. To cite a few, Duchamp [17] allows user-controllable preprocessing followed by a threshold; AEGEAN [18] exploits spatial correlation on data to fit a predictive model; PySE [19] selects islands of high pixel values after removing the background noise, then deblends these islands before fitting a 2D Gaussian model; PyBDSF [20] gathers several image decomposition techniques and offers tools to compute the properties of extracted sources.

The natural evolution from this was the use of classic computer vision techniques. For instance, [21] applies Latent Dirichlet allocation, a generative statistical model, to image pixels, thus clustering them into either background or source pixels. Riggi et al. [22] developed another similar technique, which instead performs the source segmentation (or at a lower level, clustering), using the k-means and Self Organizing Maps (SOM) [23] algorithms based on pixels' spatial and intensity values. While these techniques sometimes obtain good results, they are not as capable of generalizing on unseen data as deep learning models, which explains the shift of more recent works towards deep learning techniques for automated source detection.

ConvoSource [24] is one such deep learning technique, which uses a relatively lightweight CNN, made up of 3 convolutional layers, dropout, and a fully connected (dense) layer to generate the final output: a binary mask. This output, of course, cannot differentiate between different classes, as it is a binary mask, and thus only performs binary classification. DeepSource [25] is another CNN-based model, made up of 5 convolutional layers, with ReLU activations, residual connections, and batch normalization. This model differs from ConvoSource in that the earlier layers are used to boost the signal-to-noise ratio of the input, effectively improving the quality of the image, with a post-processing technique responsible for identifying the predicted sources. These models use rather simplistic CNNs, and thus the models will not be as capable of recognizing high-level features as state-of-the-art object detection techniques, leading to less than satisfactory performance on more complex or fainter objects. CLARAN [26] is based on the Faster R-CNN object detector [27], fine-tuned from weights trained on the ImageNet dataset [28], with some architecture changes,

such as the RoI Pooling layer replaced by differentiable affine transformations. Astro R-CNN [29] applies Mask R-CNN [30], an instance segmentation technique, the evolution of Faster R-CNN, to perform object detection on a simulated dataset.

Mask Galaxy [31] is yet another implementation that uses Mask R-CNN. It fine-tunes a model trained on the COCO dataset [32] with astronomical data. This model, however, was only trained to detect one class.

HeTu [33] uses a combination of residual blocks [34] and a Feature Pyramid Network (FPN) to locate objects in radio images and classify them among four categories.

While these works prove that great strides have been made in the development of automated source finders, there is still a large margin for improvement. Most of these works, except for HeTu and CLARAN, are trained on simulated datasets, which can limit the capability of these models to generalize to actual telescope data as they can inherit the bias of the acquisition instrument.

Object detection methods in the radio astronomical field employ especially architectures based on Mask R-CNN [29, 31, 35] and similar convolutional architectures, using FPNs and ResNets [26, 33]. In recent years, in the computer vision field, several approaches have pushed forward the state of the art of object detection methods, either improving the existing architectures [36–39] or employing the transformer architecture, introduced first in the NLP field [40] and then adapted to the vision domain [41–43]. One of the families of architectures widely employed as object detectors is the one based on YOLO [44]. YOLOv4 [36] and YOLOv7 [37] present similar architectures, with the latter being the improvement of the former in terms of both performance and efficiency. These models are fully convolutional networks based on FPNs with added modules to improve performance and model size. Similarly, the EfficientDet [39] models build on the EfficientNet [45] model to adapt a well-established convolutional architecture to the object detection task. Transformers introduce the multi-head attention mechanism, which allows the processing of long sequences and has been successfully applied to the vision domain by treating images as sequences of patches and applying the attention mechanism to such sequences. DETR [42] is one of the first methods to have applied transformers to object detection by treating the task as a bipartite set matching one, where they match a fixed set of object queries to the detected objects in the image. YOLOS [46] simplifies DETR's architecture by removing the decoder and treating the image patches and the object queries as a single sequence, resulting in a more lightweight model. A task similar to object detection is semantic segmentation which, instead of predicting bounding boxes, estimates a segmentation mask classifying each pixel of the image. Typical semantic segmentation approaches make use of a U-Net [47] architecture, made of a downsampling encoder, a bottleneck, and an upsampling decoder. The downsampling and upsampling paths are connected by skip connections that avoid gradients from becoming too small and help to keep low-resolution features. Improvements, such as the deep supervision mechanism [48], contribute to making these models more robust by learning more informative features using an objective function on the hidden layers of the upsampling path. U-Net++ [49] proposes a combination of architectural improvements by improving the skip connection pathways and extending the deep supervision mechanism, while Tiramisu [50] employs DenseNets [51] instead of ResNets [34] improving the performance using fewer parameters. Other approaches [11, 52] pro-

pose hierarchical decoding for segmentation, which, instead of employing the pipeline of downsampling and upsampling path, decode features at multiple resolutions and combine them at the end.

To the best of our knowledge, no study provides an overview of deep learning models, especially relative to semantic segmentation ones, applied to radio-astronomical images. The only work exploring semantic segmentation in radio-astronomy is carried out in [53]. Thus, one of the main contributions of this work will be the standardized comparison of a significant number of state-of-the-art detection and segmentation approaches on a dataset composed of real images from several telescopes. Additionally, this work should serve as a baseline of object detection and segmentation approaches for the radio-astronomical community to orient any scientist who recognizes that deep learning architectures may suit their case study. A complete summary of other comparable approaches can be found in [54, 55].

3 Dataset: radio astronomical images

To train and validate the models we made use of a dataset containing 10952 image cutouts extracted from different radio astronomical survey images taken with the Australian Telescope Compact Array (ATCA), the Australian Square Kilometer Array Pathfinder (ASKAP), and the Very Large Array (VLA) complemented with radio galaxies coming from the Radio Galaxy Zoo (RGZ) project [56].

Each raw data sample from the surveys comes in a large file size ($\sim$ 4GB), which is intractable by deep learning models as it would require an excessive amount of resources, so we extract cutouts from each sample. Each cutout may contain multiple objects of the following three classes:

- *Extended Sources*: Radio galaxies were taken from the Data Release 1 (DR1)[1] (Wong et al., in preparation) of the Radio Galaxy Zoo (RGZ) project [56], using 1.4 GHz radio observations at 5" resolution from the Faint Images of the Radio Sky at Twenty cm (FIRST) survey [57]. The data samples consist of extended sources with a 2- and 3-component morphology. Another sample of such sources was extracted from the 1.2 GHz ASKAP-36 SCORPIO survey at $\sim$9.4"$\times$7.7" resolution (see [58] for a description of the survey). This category includes both extended emission sources and multi-island sources with two, three, or more components. All the sources are likely extragalactic, since HII regions, planetary nebulae, and supernova remnants have been removed.
- *Compact sources*: Compact radio sources with single island morphology were taken from the ASKAP-15 and ATCA SCORPIO surveys, reported above.
- *Imaging artifacts*: A collection of imaging artifacts around bright radio sources was obtained from different radio observations: ASKAP EMU pilot survey at $\sim$12.5"$\times$10.9" resolution [59], 912 MHz ASKAP-15 SCORPIO survey at $\sim$24"$\times$21" resolution and the aforementioned 1.2 GHz ASKAP-36 SCORPIO sur-

[1] https://cloudstor.aarnet.edu.au/plus/s/agKNekOJK87hOh0 from https://github.com/chenwuperth/rgz_rcnn/issues/10

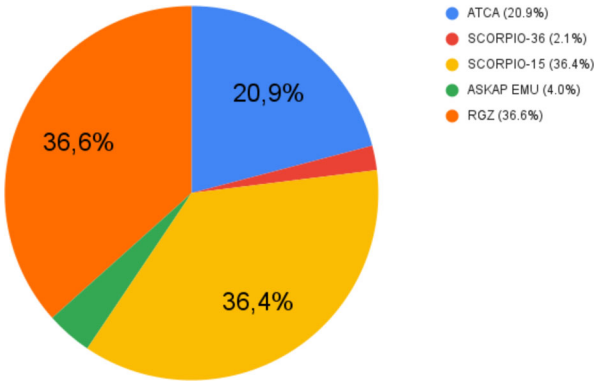

Fig. 1 Pie chart of the origin of the images in our dataset

vey, 2.1 GHz ATCA SCORPIO survey at ∼9.8"×5.8" resolution [60]. Traditional algorithms extract these artifacts as real sources while they are spurious.

We present a pie chart of the surveys we used to compose our dataset in Fig. 1. Throughout the paper, we will omit 'source' when it becomes redundant and only use their categorization, i.e. one among *'compact'*, *'extended'*, and *'spurious'*.

The whole dataset, aggregating images from the aforementioned surveys, consists of a total of 36, 398 objects, then split into 3 subsets: training (70%), validation (10%) and test (20%) as shown in Table 1. We use a dataset composed of images acquired using different telescopes, which can help the models better generalize over different image sources. Using images from only one survey can result in a higher bias induced in the network, as analyzed in [35], Section 4.2.1. We perform a similar analysis on the impact of training on a subset of the data, split by telescope, and report the results in Section 5.1.3

We extracted image cutouts (single-channel, 132×132 pixels, FITS format) from the reference data using the CAESAR tool [22]. The RGZ dataset already provides source cutouts in the same format and bounding boxes for extended source objects. We use a dataset composed of image cutouts to train and test the models, as training on the images at the original size would be computationally infeasible. The models we chose for our analysis support image sizes up to 1333 × 1333 pixels. Moreover, we resize the images before feeding them to the model.

Table 1 Number of object samples and images for each subset

Object category	Train	Valid.	Test	Total
Spurious Source	934	133	267	1,334
Compact Source	20,603	2,943	5,886	29,432
Extended Source	3,940	562	1,125	5,627
# of images	7,653	1,064	2,235	10,952
# object per image (avg.)	3.32	3.34	3.26	3.30

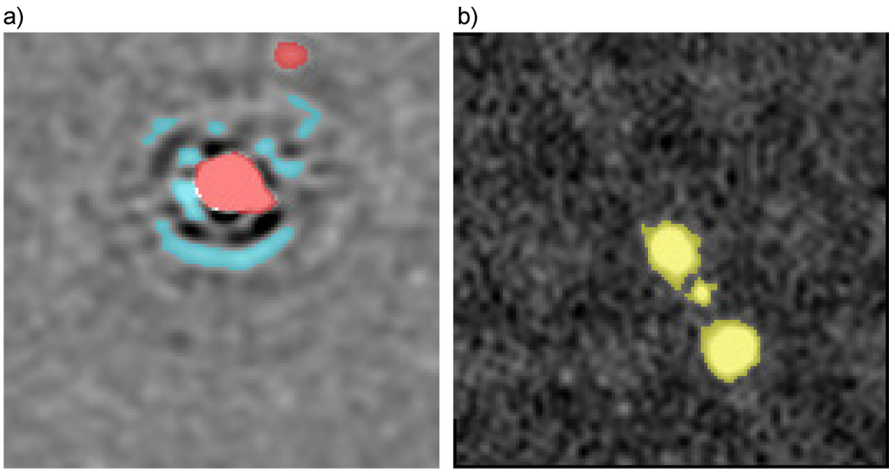

Fig. 2 Image and annotation sample pairs used in training. (a) Compact sources (in red) and spurious sources (in blue). (b) Extended source sample (in yellow)

While bounding boxes are sufficient for training object detection models, we needed to use segmentation masks to supervise the training of the semantic segmentation models. For this reason, a raw object segmentation was preliminarily produced with the CAESAR source finder and later refined by visual examination, needed in particular for spurious sources and bright sources near spurious ones, as source finders often detect them as belonging to the same island. Images belonging to the RGZ survey have been annotated using both infrared and radio data, while annotations of data originating from other surveys are based only on radio data. Samples of annotated images are reported in Fig. 2

Raw and processed data are kept under version control, using the Data Version Control (DVC) framework[2].

3.1 Data format

Input data and annotations represent a crucial element in any machine-learning-based algorithm, as this strongly influences the training procedure, as well as the final performance of the model. After collecting the images and information about segmentation masks, we need to organize our information so that it can be managed by deep learning models. We introduce a data format to be shared among the models, to avoid the need to adapt the data loading pipeline each time we need to train a new model.

In our case study, radio-astronomical data and annotations are stored separately in two different formats. The images come as FITS [3] files, while the annotations are stored in JSON files, one for each image, where all the individual FITS files containing each object mask are specified. Before being fed to any model, images need to go

[2] https://dvc.org/

[3] https://fits.gsfc.nasa.gov/fits_documentation.html

through a pre-processing phase. We first remove any inconsistent values (i.e. NaN values) from the FITS data by setting them to the minimum pixel value in the image matrix. Then, we apply a Z-Scale [61] normalization with a contrast value of 0.3 on each image to improve the object visibility before converting the FITS data to PNG. Finally, we convert the data into a PNG image by rescaling all values to the [0, 255] range and replicating the single-channel image on three channels. Such transformation is necessary as both object detection and semantic segmentation models are designed to operate on RGB image data. We did not apply any background subtraction step to the resulting images, as the network is expected to learn the noise pattern from the data. We acknowledge that these steps can introduce some degree of information loss, as the range of possible values is reduced, and fine-grained details may be lost. It is in our interest to find a better way to preprocess the data, modifying the architecture of some models to exploit the full value range of the images in FITS format, skipping the conversion to PNG.

During training, we employ image augmentation to prevent the model from overfitting. We apply a random number of augmentations (0 to 2) to each image before being fed to the model during training. The available augmentation operations are: (1) *random horizontal flip* (the image is flipped along the x-axis with probability p); (2) 90 deg *rotation*; and (3) *random resize and cropping* (after resizing the image to a set of possible fixed values, a random area, with a fixed size, is cropped from the image). The number and type of operations applied to a particular image are random and different for each epoch. To avoid confusion, in this work, when we mention epochs, we refer to machine learning training epochs. While training a model, we say it completes an epoch when the whole training dataset has been used to update the model parameters. Training a model requires multiple epochs, meaning that the model will "see" the same dataset multiple times.

For each image, we extract information about the segmentation mask of each object and compute the maximum and minimum of the segmentation masks along the 2D coordinates (x,y) to get the coordinates of the respective bounding box coordinates. Finally, we aggregate the bounding box and segmentation mask information into a single JSON object and we link it to the corresponding image by adding, to each object in the image, the "image id" field. In addition to bounding box and segmentation mask annotations, we add to each object description other fields, namely: *category id* to define the type of object for each mask; *area* that indicates the extension covered by the object, in terms of original pixel size, trivially calculated by multiplying the width and height of the bounding boxes; *iscrowd* that specifies if a region contains a set of packed objects of the same class. We add these attributes to conform to the COCO [32] format, as many object detection models, including the ones we tested, rely on this kind of annotation. A visualization of a final JSON annotation file is shown in Fig. 3.

4 Architectures

We investigate several deep learning models, performing object detection and semantic segmentation, to identify and classify sources. We mainly explore architectures based

```
{
  "images": [
    { "id": 0, "width": 132, "height": 132, "file_name": "sample8_sidelobe0022.png" },
    [...]
    { "id": N, "width": 132, "height": 132, "file_name": "sample8_sidelobe0030.png" }
  ],
  "annotations": [
    {
      "id": 0, "category_id": 1,
      "image_id": 0, "iscrowd": 0, "area": 36,
      "bbox": [ 68, 0, 9, 4 ],
      "segmentation": [
        [ 68, 0, 68, 1, 69, 2, 70, 3, 71, 3, 72, 4, 73, 4]
      ]
    },
    [...]
    {
      "id": N, "category_id": 2,
      "image_id": 0, "iscrowd": 0, "area": 28,
      "bbox": [ 27, 30, 4, 7 ],
      "segmentation": [
        [ 28, 30, 27, 31, 27, 4, 27, 33, 7, 34, 28, 35]
      ]
    }
  ],
  "categories": [
    { "id": 1, "name": "galaxy" },
    { "id": 2, "name": "source" },
    { "id": 3, "name": "sidelobe" }
  ]
}
```


Fig. 3 An example of annotation file in JSON (COCO-like) format

on CNN and Transformers [40]. Most CNN-based object detection models rely on Region Proposal Networks [27], which are computationally expensive modules that propose several regions that could potentially contain objects and then select only a few of them. This technique is more efficient than the selective search [62] algorithm used in previous approaches. Single-stage detection algorithms [63] are CNNs that avoid the use of RPNs by dividing the image on a grid and assigning a confidence score to one or more proposed predictions within each grid cell. Transformers are a more recent family of architectures, originally designed for Natural Language Processing tasks which have been extended to the vision domain [41]. This last work paved the road for a whole family of models, which have been applied to several tasks in the vision domain, such as image classification [64–66], object detection [42, 46, 67, 68], and knowledge distillation [43, 69].

Semantic segmentation models follow a different architecture. They are characterized by encoder-decoder pipelines, where images are first encoded to a low-dimensionality representation by downsampling the feature maps at each encoding stage to generate its latent representation. Then the image is expanded back to the original image size in the decoding path to obtain the binary masks that classify the

objects. There are many variants that extend this framework by applying skip connections [47], deep supervision [48], or hierarchical decoding [11].

4.1 Detection models

Object Detection defines all deep neural networks designed with the goal of identifying and locating objects within an image. Such a task is carried out by defining bounding boxes and associating labels to the predicted object. Deep learning-based approaches generally employ convolutional neural networks (CNNs) to perform end-to-end object detection [27, 30, 39, 44], but, recently, transformer-based ones gained relevant popularity [42, 46, 67, 68]. CNN-based object detection models typically consist of two modules: a backbone network and several prediction heads. The backbone generates a low-resolution, feature-rich image representation, usually employing widely affirmed classifier architectures (e.g. ResNets). Feature maps are then fed to a series of convolutional layers that learn to predict a bounding box and a label for each detected object.

Most object detection networks employ a region proposal network (RPN), introduced in [27]. This network generates a set of N bounding boxes at different aspect ratios for each point on the convolutional feature map. The output of the backbone network is then used to determine whether each bounding box belongs to the foreground or the background by providing the feature map cropped by the anchor box to a small CNN. Then, starting from these anchor boxes, the objective function is computed on the offset between the anchor boxes and the ground truth. Usually, multiple proposals will yield a high score for the same object, especially for large objects, so further filtering is needed. Non-Maximum Suppression (NMS) computes the Intersection over Union (IoU) between the highest-scoring predicted box and the next high-scoring boxes and removes the ones with IoU higher than a threshold. Such a threshold is a hyperparameter. This family of models is known as two-stage detectors, as they process the input in two separate phases.

One-stage detectors skip the region proposal network by dividing the input image into N_g grids. The model predicts B bounding boxes and a confidence score for each grid. NMS is applied to the highest-scoring boxes to filter redundant predictions. These detectors allow for a more efficient prediction at inference time in terms of computational resource requirements, at the cost of model performance.

These methods limit the proposal of bounding boxes to a fixed number, which may seem a limitation. As we are working on image crops that contain fewer than fifteen objects, the number of proposed boxes is higher than the maximum number of objects.

Figure 4 shows the general framework of these two families of object detectors.

Transformer-based methods employ a simpler approach by using the backbone network in the same way as CNN-based ones and feeding its output to a transformer [40], as used in [42], or skip the use of the CNN backbone and use an encoder-only transformer [46]. The final prediction is given by a series of prediction heads, typically linear layers. The loss function employed in these models is a bipartite matching loss, introduced in [42]. Transformers require more computational resources and are more difficult to lead to convergence, but can yield more robust models. Note that some

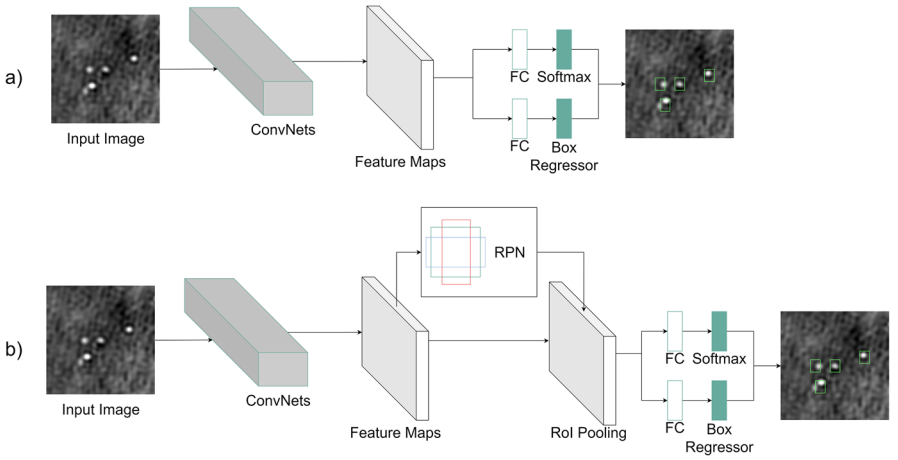

Fig. 4 General pipeline designs of a one-stage object detector a) and a two-stage object detector b), the latter employing RPN and RoI Pooling as additional operations to propose and filter regions

approaches have similar names and can generate confusion. YOLO (You Only Look Once) is a single-stage detector based on CNNs. YOLO9000, YOLOv3, and YOLOv4 are gradual improvements of the same architecture. YOLOS (You Only Look at One Sequence) is a separate model with a completely different architecture.

4.1.1 Mask R-CNN

We used a Mask R-CNN-based model adapted for working on radio-astronomical images [35]. Mask R-CNN [30] is a deep learning model that builds on Faster R-CNN [27], which, in turn, builds on R-CNN [70]. Mask R-CNN performs instance segmentation on images, i.e. it combines object detection, classification, and semantic segmentation, in the form of a per-pixel mask for each object, differentiating between overlapping objects.

The first component of Mask R-CNN, which is also one of the newly introduced features compared to Faster R-CNN, is the Feature Pyramid Network (FPN) [71]. The FPN, also referred to as the backbone, serves as a feature extractor for objects at multiple scales, with shallower layers detecting lower-level features and deeper layers detecting higher-level features. The bottom-up pathway is typically a ResNet [34] (ResNet101 in our case), chosen for their residual connections, which add output and input of a series of neural layers to solve the vanishing gradient problem [72]. This problem relates to the gradients becoming too small in deeper layers due to chained multiplications by small numbers. The top-down pathway consists of a convolution at each layer, followed by a lateral connection from the bottom-up pathway. This lateral connection allows the FPN to combine the top-down pathway's high-level features with the bottom-up pathway's low-level features. The output of the FPN is then passed to the Region Proposal Network (RPN).

The RPN scans predefined areas of the image, referred to as anchors, and picks out regions likely to contain an object of interest. This component does not look

directly at the input image again but uses features already extracted by the FPN, making it efficient. For each anchor, the RPN determines whether it is background or foreground, meaning whether it contains an object of interest and refinements, which are more precise 'outlines' of the object. The regions proposed with the highest confidence are refined and passed on to the next component.

The regions proposed by the RPN are of varying shapes and sizes, which is problematic for the architecture of the network, and are thus resized to a fixed, predefined size. This step is called ROI Pooling. In Faster R-CNN, this is accomplished using ROIPool, which crops and resizes the feature map. However, this can result in the loss of significant data, along with spatial misalignment of mask pixels when overlayed with the original image. Mask R-CNN instead utilizes ROI Align, which samples the feature map at different points and uses bilinear interpolation to achieve the desired size.

The final component, and what particularly sets Mask R-CNN apart from its predecessors, is the Fully Convolutional Network (FCN), also known as the 'mask branch'. This branch is a fully convolutional network that can retain spatial information and generates a per-pixel mask for the proposed regions and thus for the input image.

Given the nature of the data, we carry out some data preprocessing steps specific to radio astronomical data, which are not present in common implementations of Mask R-CNN. For example, setting 'NaN' pixels to the image minimum and applying Z-Scale [61] normalization. Moreover, A significantly intensive Hyper-Parameter Optimization process is also carried out to fine-tune the model's architecture for the data it is made to handle. Furthermore, when being used for detection, we apply post-processing steps, such as merging overlapping or connected objects of the same label or retaining only the object with the highest confidence in other cases.

4.1.2 Detectron2

Detectron2 [38] is Facebook AI Research's next-generation library that implements state-of-the-art object detection and instance segmentation algorithms. It is a rewrite of Detectron [73] that started with maskrcnn-benchmark [74]. It supports multiple tasks such as bounding box detection, instance segmentation, keypoint detection, densepose detection, and others. It also provides pre-trained models that can be easily loaded and used on new data sets. It allows for custom state-of-the-art computer vision technologies to be easily plugged in and includes robust models, e.g. Faster R-CNN [27], Mask R-CNN [30], RetinaNet [75], and DensePose [76], and also features several new models, including Cascade R-CNN [77], Panoptic FPN [78], and TensorMask [79].

For detecting objects, our experiments made use of the Base (Faster) R-CNN with Feature Pyramid Network (Base-RCNN-FPN), which is the basic bounding box detector extendable to Mask R-CNN for instance segmentation. Faster R-CNN with FPN backbone is a multi-scale detector capable of detecting objects at different scales, accomplishing high accuracy and efficiency. A generic Region-Based Convolutional Neural Network (R-CNN) is composed of three main components. The first is a region proposal network that generates candidate regions (bounding boxes) using computer vision techniques. The second one is the feature extraction module which uses convolutional neural networks to extract the features from the candidate regions. Finally,

the last component is a classifier that predicts the classes of the proposed candidates using the extracted features.

Specifically, for Faster R-CNN, these components are:

1. Backbone Network: extracts feature maps from the input image at different scales. Base-RCNN-FPN output features are called P2 (1/4 scale), P3 (1/8), P4 (1/16), P5 (1/32) and P6 (1/64).
2. Region Proposal Network: detects object regions from the multi-scale features. 1000 box proposals (by default) with confidence scores are obtained.
3. Box Head: crops and warps feature maps using proposal boxes into multiple fixed-size features, and obtains fine-tuned box locations and classification results via fully-connected layers. Finally, 100 boxes (by default) in maximum are filtered out using non-maximum suppression (NMS).

Mask R-CNN extends on Faster R-CNN by adding another branch in parallel for pixel-level object instance segmentation. The branch is a fully connected network applied on ROIs to classify each pixel into segments with little overall computation cost. It adds a mask head parallel to the classification and bounding box regressor heads. With respect to Faster R-CNN, one major difference is the use of the RoIAlign layer, instead of the RoIPool layer, to avoid pixel-level misalignment due to spatial quantization.

Mask R-CNN performs better than the existing state-of-the-art single-model architectures. It is simple to train, flexible, and generalizes well in many applications.

4.1.3 YOLO

Characterized by a fully convolutional network that simultaneously predicts a set of bounding boxes and the associated class probabilities, YOLO [44] falls into the category of one-stage detectors. Such a model is characterized by faster execution time in comparison to two-stage detectors, at the cost of a certain degree of accuracy. The loss of accuracy is implicit in the design of the model, as YOLO detects objects by analyzing patches of the entire image. This causes a bias in the prediction of each object, which depends on the context of the patch where the object is located, meaning that the same object, put in another context, might be misdetected. At the core of this method is the following idea: the input image, whose resolution is $P \times P$, is divided into an $S \times S$ grid, and each cell is treated as responsible for detecting an object if the center of such object falls in that cell. Note that $S << P$, so each grid cell contains multiple pixels. The authors set $S = 7$. Each cell predicts a fixed number B of bounding boxes, and a confidence score for each of them, which determines how confident the box is to contain an object and also how accurate it is at covering the object. The confidence score is given by

$$Pr\,(Object) \times IoU \tag{1}$$

where $Pr(Object)$ refers to the probability of that box containing an object, and IoU measures the intersection over union between the prediction and the ground truth. This quantity should be as high as possible when an object is contained in the cell.

If the model predicts no objects in a bounding box, the confidence score should be 0. 5 values are predicted for each bounding box: 4 coordinates ($xyhw$) to locate the bounding box in the image and the confidence score. Furthermore, for each grid cell, a class probability is computed for each category. Even if a grid cell contains multiple bounding boxes with different objects, the model will only predict a class for that cell, which translates into a loss of accuracy, especially for small objects in a group.

The architecture of YOLO is made up of 24 convolutional layers and two fully connected layers. The first 20 convolutional layers are pre-trained on the ImageNet [28] dataset on the classification task; the whole pipeline is fine-tuned with the detection task (i.e. the predictions are bounding boxes). Training is carried out using the sum-squared error as the objective function, as it is easy to optimize. Many grid cells do not contain any objects and will have a zero confidence score. This may cause an increase in the magnitude of the gradients from cells that do contain an object, leading to unstable training and divergence. To address this problem, the loss for the bounding box coordinate prediction is increased, while the loss relative to the confidence score is decreased for boxes that do not contain objects. Although it represents an improvement in terms of performance and methodology when compared to two-stage approaches, YOLO presents the following limitations:

- Limited flexibility since each grid cell predicts a fixed number of bounding boxes and a single class, which makes it impossible to distinguish among objects of different categories in the same grid cell or to predict more objects than the maximum number of boxes within a cell;
- Issues to generalize and recognize the same object in a different aspect ratio than the one provided in the training data, as its inductive bias is conditioned by the grid that divides the image;
- Errors are not treated proportionally to the box size but in terms of their absolute value, which is a problem as a small error on a large box does not affect prediction accuracy, as the same error value on a smaller box does.

There are multiple versions of this approach, each of which represents a relevant improvement in terms of computational performance and prediction accuracy relative to the older version. The first improvement of the architecture is represented by YOLO9000 [80], which, among others, provides the following enhancements:

- Batch normalization [81], which contributes to stabilizing training;
- A higher resolution classifier, capable of recognizing features at a higher scale;
- Multiscale training, which processes batches at different resolutions to make the model more robust to scale and aspect ratio variations. This is made possible by employing Feature Pyramid Networks (FPN) [71].

A further improvement in performance and computational speed is given by YOLOv3 [82], which improves the backbone by adding residual connections and optimizes some training hyperparameters to make the model converge faster. In our benchmark, we use two versions of this architecture: **YOLOv4** and the latest **YOLOv7**. YOLOv4 [36] boosts YOLOv3 performance by operating on the following:

- Expand the receptive field of the detector by increasing the number of convolutional layers in the backbone, integrating the Cross-Stage Partial Network [83] into the backbone, and adding to this the Spatial Pyramid Pooling [84];

- Replace the FPN in YOLOv3 with the PANet [85] module

YOLOv7 [37] builds upon this architecture and includes a series of architectural improvements to boost performance and resource requirements. Some of the improvements are the following:

- Extended efficient layer aggregation to perform layer fusion that makes the operation less resource-demanding by reducing the path of the gradients in the computational graph;
- Auxiliary supervision heads in intermediate layers to improve the features' quality.

4.1.4 EfficientDet

EfficientDet [39] represents a family of one-stage detectors whose general architecture follows that of YOLO and aims to improve efficiency and performance by exploring several architecture variations to reach state-of-the-art performance more efficiently. This work realizes the enhancement by operating mainly on two aspects. First of all, most object detectors make use of multi-scale features to gather information from feature maps at different resolutions and fuse them to output the prediction. Such approaches do not distinguish between feature maps at different scales, so they treat them equally. Yet, each feature map highlights different properties of the image, so they should be processed accordingly. To tackle this problem, the authors introduce the Bi-directional Feature Pyramid Network (Bi-FPN). Scaling up the model performance is another concern addressed by this work. Most approaches rely on scaling up the backbone or the image size, while the authors claim it is necessary to scale the box and class prediction heads accordingly. Finally, they also modify the backbone architecture, choosing EfficientNet [45], as it is less burdensome in terms of GFLOPs and model size.

The BiFPN module represents the core of the EfficientDet architecture and solves the task of gathering information from the image at different scales. This method improves existing state-of-the-art approaches like PANet [85] and NAS-FPN [86], both requiring a high amount of resources to carry out training and inference, by employing a more efficient strategy to fuse features at different scales. In particular, they improve performance by operating on the following main aspects:

1. Remove cross-scale connections that have only one input, as it has been observed that they contribute poorly to the final output by not providing a significant amount of information for the final prediction;
2. Add a residual connection between the input and the output for features at the same scale, so to fuse more features without adding parameters to the model;
3. Repeat the top-down and bottom-up phases multiple times to gain a bidirectional flow of information, which enhances high-level feature fusion.

Finally, as features need to be treated differently according to their resolution, the fusion strategy is modified. While most approaches use a rescaled sum of all features, this method offers the following ways to combine them:

- Weighted fusion: feature maps are scaled by a learnable weight, and they are summed together. This may cause training instability, as the value of the weight is unbounded.

- Softmax-based fusion: each weight is given to a softmax operation, to rescale it into a range of [0, 1]. Computing softmax on each weight before multiplying it by the feature maps solves the aforementioned problem, but introduces a new one: significantly slower computation.
- Fast-normalized fusion: the operation of softmax is modified slightly by removing the exponential operation and adding an ϵ term to the denominator for numerical stability. This approach is more efficient and avoids the unbounded value problem.

The overall model consists of a backbone based on EfficientNet, a multi-scale feature aggregation network, carried out using BiFPN, and a box/class prediction network, realized by employing fully convolutional networks.

The model can be scaled up or down by setting a coefficient ϕ, which influences the depth and width of the modules. For the backbone, the coefficient determines which EfficientNet architecture to use, from EfficientNet-B0 to B6. The BiFPN network is scaled according to the following equations:

$$W_{BiFPN} = 64 \cdot \left(1.35^{\phi}\right), \qquad D_{BiFPN} = 3 + \phi \tag{2}$$

where W_{BiFPN} indicates the number of channels of the convolutions (width) and D_{BiFPN} refers to the number of layers in the network (depth).

The box and class predictors are scaled as well in the same manner. They share their width with the BiFPN module, but their depth is computed in the following way:

$$D_{box} = D_{class} = 3 + \lfloor \phi/3 \rfloor \tag{3}$$

Finally, the input image resolution must also be scaled so that it can be processed by the BiFPN. For this reason, the input image size has to be divisible by 2^7, thus the following equation:

$$S_{input} = 512 + \phi \cdot 128 \tag{4}$$

The family of EfficientDet models includes EfficientDet-D0 to D7, where the number corresponds to the value of ϕ.

We tested EfficientDetD1 and D2, as testing more complex versions of the models on our relatively simple images could lead to severe overfitting.

4.1.5 DETR

DETR [42] is one of the first approaches to employ transformers to perform object detection. This method poses object detection as a direct set prediction problem, thus relieving the bounding box regression from the burden of Region Proposal [27] and Non-Maximum Suppression. These two steps have a high impact on performance, as they consist in generating thousands of proposals and discarding most of them based on an Intersection over Union threshold value. DETR consists of three main modules: (1) a CNN backbone, typically ResNet50 [34], which serves as a 2D feature extractor of the input image; (2) an encoder-decoder transformer, which computes self- and cross-attention on the features extracted by the backbone and produces a fixed set

of predictions; (3) a feedforward network that assigns either the predicted class or a special **no-object** class to each element of the prediction set.

The core mechanism of the model resides in the transformer module, which is composed of an encoder and a decoder. Before being fed to the encoder, 2D features are projected onto a hidden dimension d by means of a $\times 1$ convolution and embedded with a spatial positional encoding. The transformer is designed to process sequences, so the 2D features of dimension $d \times H \times W$ are flattened into a sequence of $d \times HW$. Then, the encoder computes self-attention on the sequence followed by residual connections, layer normalization [87], and feedforward layers. Such operations are iterated as many times as the number of encoder layers stacked together. The decoder shares part of its architecture with the encoder but is characterized by an additional module: the cross-attention module, placed after the self-attention block. The decoder receives a fixed set of *object queries* as input, which are randomly initialized learnable parameters. Then it computes self-attention on the input, followed by a cross-attention between the object queries and the encoder output. This step is crucial in transformer architectures as it contributes to mapping the information between the source sequence (i.e. the image features) and the target sequence (i.e. the predicted object categories and locations).

Finally, each element of the output sequence from the decoder is fed to a shared feedforward network to predict the bounding boxes and the corresponding class. The decoder always outputs a fixed set of predictions, equal to the number of provided object queries, and each prediction is given a certain class if an object is predicted with a high confidence score, otherwise, the object is associated with a **no-object** class.

An important remark about the decoder's behavior is that it differs from the one defined in [40], which operates by iteratively generating an element of the sequence and feeding that element back to the decoder to generate the next element. In DETR, the decoder executes in a single step, predicting all the outputs in parallel. Since the problem is framed as a direct set prediction one, the loss function to optimize is a matching cost between the ground truth set and the predicted set. The loss optimization follows two steps: (1) the Hungarian algorithm is used to find the optimal assignment among all the possible permutations of the two sets; (2) the Hungarian loss gives the actual loss function between the sets. The loss functions also depend on the bounding box loss, which uses a linear combination of the $L1$ loss and the IoU loss [88]. Additionally, an auxiliary loss is employed at each decoding layer to improve performance.

The main contribution of this work is that, by using attention maps, DETR can more accurately predict complex and overlapping objects, avoiding the computational burden that comes with traditional two-stage detectors. A typical drawback of this model is its non-trivial training, which requires pre-training on extended datasets and learning rate warm-up schedules, as reported in [89], and its limited capability on smaller objects.

We tested the minimal version of DETR among the variants reported in [42], i.e. DETR with ResNet50 as the backbone for feature extraction.

4.1.6 YOLOS

Finally, to complete the overview of the existing models, we also tested a novel object detection approach based entirely on transformers. While DETR still relies on a convolutional backbone, YOLOS employs a fully transformer-based architecture similar to ViT [41], adapted to the object detection task, as the latter is designed to tackle classification tasks. Compared to ViT, YOLOS drops the *[CLS]* token, as its purpose is to classify the input by performing attention on the whole patch sequence, whereas here, the task is to detect objects in the scene and locate them. Instead of using the *CLS* token, YOLOS adds a hundred learnable detection tokens concatenated to the input image patches along the sequence dimension. The encoder then performs self-attention over the whole sequence, and the tokens will be used as output features that will be processed to generate the predictions. These tokens have the same purpose as the object queries in DETR. YOLOS shares some aspects with DETR: in addition to the concept of learnable tokens and the core based on transformers, they employ the same optimization strategy by minimizing a bipartite loss function, using the Hungarian algorithm and the respective loss defined in [42]. Yet, YOLOS introduces a significant change in the transformer architecture: it employs an encoder-only transformer, while DETR presents an encoder-decoder architecture. The lack of a decoder module allows for a smaller number of parameters, thus a lighter model in inference, without significantly impairing prediction performance. Before being fed to the transformer, the image is divided into patches of fixed dimensions. Then each patch is flattened to have a sequence of one-dimensional vectors and projected onto a latent space. A fixed-size sequence of learnable detection tokens, each of which has the same size as the projected patches, is concatenated to the input sequence. This composed sequence is then summed with a sinusoidal positional encoding to maintain positional information within the sequence in the transformer. The transformer computes self-attention between the elements of the input sequence, which in this case is the union of the projected patches and the detection tokens. This step is necessary for injecting information from the input image patches into the detection tokens. The transformer encoder outputs a sequence of the same length as the input, but only the elements corresponding to the detection tokens are used to compute the predicted objects. Such output is provided as input to a 2-layer MLP with GELU [90] activation functions. Finally, two distinct heads, each consisting of a one-layer MLP with a ReLU activation function, predict the bounding box coordinates and categories, respectively. A peculiarity of the approach used in YOLOS is the minimization of injected inductive bias. This means that the model exploits as little as possible the spatial structure and geometry of the input, which makes this approach versatile and capable of efficiently operating on data of different sizes and aspect ratios. The only form of spatial inductive bias implicit in the model is given by the patch sub-division of the image, but, other than this, no spatial convolutions are involved.

YOLOS comes in different variants, which differ mainly in model size, influenced by the embedding size and the number of heads. As data is not too complex, consisting

of grayscale images, we tested the lightest version of the model, i.e. **YOLOS-Tiny**, to avoid having too many parameters, which might cause overfitting. This version consists of 12 encoder layers, an embedding dimension of 192, and 3 heads for the attention mechanism, for a total of $5.7M$ parameters.

4.2 Segmentation models

Semantic segmentation represents a task with the objective of classifying individual pixels of an image to obtain segmentation masks of the same size as the original image, where each pixel is labeled as belonging to its predicted class. This task is different from the object detection one, which predicts bounding boxes, so it generally follows a different pipeline. These models do not employ Region Proposal Networks since they do not need to "find" objects in the image, but classify each pixel. The objective function used to train these models is different as well; generally a form of the Cross-Entropy loss function, as opposed to L1 or MSE distances used in object detection. The general pipeline used for semantic segmentation tasks consists of an encoder-decoder pair. The first represents the downsampling path, where the image is compressed to a latent space representation, while the latter is the upsampling path, which generates the high-resolution binary mask for the input image. In between the encoder and the decoder, a bottleneck layer projects the compressed image representation into the space of the binary mask by means of 1×1 convolutions. In our analysis, we employ the following architectures on our radio-astronomical dataset, as shown in Fig. 5: a basic encoder-decoder architecture, U-Net [47], which adds skip connections to the basic architecture, U-Net with deep supervision, which enhances the loss by computing it also on intermediate upsampled masks, Tiramisu [50], which employs DenseNets [51] instead of convolutions in the downsampling and upsampling blocks, and PankNet [11], which makes use of hierarchical decoding for information mixing between upsampling paths.

Segmentation models were originally designed for medical imaging applications. We think that our use case can benefit from such architectures, as medical images and radio-astronomical data are characterized by a similar distribution.

4.2.1 Encoder-decoder

The baseline for our segmentation models is a simple encoder-decoder pipeline with a bottleneck layer between the two paths, as shown in Fig. 5(a). This model employs a series of convolutional blocks in the encoder and several up-convolutional blocks in the decoder. Each downsampling block is composed of two 3×3 convolutions followed by a rectified linear unit (ReLU) and a 2×2 max pooling to reduce feature resolution. The feature channels are doubled at each downsampling step. In the decoding path, each block performs a 2×2 transposed convolution, which expands the resolution of the feature map while halving the size of the feature channel, followed by 2 layers of 3×3 convolutions and a ReLU. In the final layer, 1×1 convolutions are applied to map the dimension of the feature map to the number of classes.

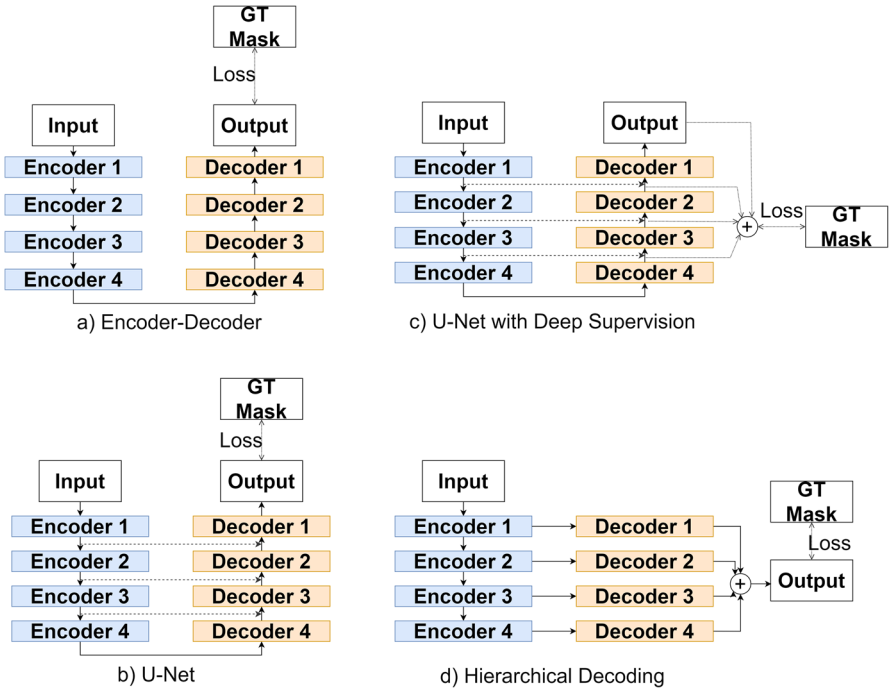

Fig. 5 Different semantic segmentation architectures. a) Classic encoder-decoder approach; b) original U-Net architecture, with skip connections; c) U-Net with added deep supervision for enhanced loss; d) hierarchical decoding that combines features from all decoding stages

4.2.2 U-Net

Deep layers in neural networks are generally hard to train, yielding a model incapable of properly converging to an optimal point. The most common reason behind this behavior is the problem known as vanishing gradient [72], which can occur in deep pipelines and impairs the training procedure. ResNets [34] offer residual connections as a solution to this problem by summing the input and the output of a convolutional block. In this way, the gradients can flow through the shortcut path and avoid becoming too small. U-Net [47] addresses the problem by applying skip connections, as shown in Fig. 5(b). Skip connections have the function of conditioning the upsampling output of the decoding blocks on the feature maps of the downsampling block of the same size. This is accomplished by concatenating the feature map produced by the downsampling block with the input to the corresponding upsampling block.

Another enhancement can be applied by intervening in how the cost of the objective function is computed. In the classic U-Net architecture, the loss is computed only on the last layer's output by applying a cross-entropy loss between this and the ground truth masks. We employ a U-Net variant in our segmentation models by applying the mechanism known as deep supervision [48]. It consists of computing the loss between the output of each upsampling path and the downsized ground truth mask to guide the decoder to generate meaningful masks from the early stages. An example of this architecture is shown in Fig. 5(c).

Finally, among the U-Net-based architectures, U-Net++ [49] improves the skip pathways and the deep supervision mechanism to improve performance and enhance gradient flow between the encoding and decoding paths. While U-Net simply concatenates the output of the downsampling layers with the corresponding upsampling ones, U-Net++ extends this operation by adding dense convolutional layers between the two paths, which yields a more meaningful encoding of the features as well as simplifying the computation of the gradients, thus stabilizing the training. U-Net++ employs deep supervision as well but, instead of applying it to intermediate outputs of the upsampling path, as done in [48], it computes it on the outputs of the convolutional layers of the skip pathways. This enables model pruning at inference time, which makes this model adaptable to environments with constrained computational capabilities.

4.2.3 Tiramisu

While U-Net introduces skip connections to solve the vanishing gradient problem, Tiramisu [50] acts on the structure of the convolutional blocks to make feature maps more representational by injecting higher-resolution information into lower-resolution maps. This is achieved by employing DenseNets [51], which replace the sum operation between the input and output of each convolutional block, typically used in ResNets, with concatenation. This contributes to reducing the network's parameters, as it reuses feature maps from earlier layers, freeing deeper layers from the need to learn redundant features.

4.2.4 PankNet

Finally, we employ PankNet [11], which does not make use of the downsampling-upsampling pipeline but applies encoders and decoders in a multilevel way, as shown in Fig 5(d). Each decoder receives only the corresponding encoder output as input, while the other versions concatenated it to the previous decoder output. In addition to this change in architecture, the output mask is obtained in a different way from what we have seen so far. Output feature maps from all the decoders are summed together, allowing for mixing global and local features, thus enabling a more context-aware prediction. The model is designed to process 3D CT and MRI scans, but since our case study involves 2D images, we convert the 3D convolutions into 2D when using them in our dataset.

5 Performance analysis

This Section presents the metrics and related results computed to evaluate the performance of each model on object detection (see Section 5.1) and semantic segmentation (see Section 5.2). Additionally, in Section 5.3, we present an evaluation of the ability of the models to perform with different data samples according to their signal-to-noise ratio (SNR). Finally, we evaluate the performance of the models from the computational point of view (see Section 5.4). We report the metrics on the entire dataset and

the *extended* category. We are more interested in how the models perform for this particular category as the traditional algorithms are well-affirmed for compact source detection but less efficient on single-island extended sources and incapable of detecting multi-island extended sources. Also, we expect common deep learning models to underperform on compact and spurious sources due to their small size and irregular shape. While analyzing the performance, it is useful to recall the number of objects per class (Table 1) and per SNR bin (Table 6). For further comparison, we report metrics also for the *compact* category.

5.1 Object detection

5.1.1 Detection metrics

We evaluate the performance of several detection models on the radio images dataset and compute the following metrics:

- Reliability (Precision)

$$Reliability = \frac{TP}{TP + FP} \tag{5}$$

- Completeness (Recall)

$$Completeness = \frac{TP}{TP + FN} \tag{6}$$

- F1-Score

$$F1 = \frac{2 \times R \times C}{R + C} = \frac{2 \times TP}{(2 \times TP) + FP + FN} \tag{7}$$

  where we refer to R for Reliability and C for Completeness
- mAP@50

$$mAP = \sum_{c} AP_c \tag{8}$$

To compute such metrics, we first need to compute the number of true positives *(TP)*, false positives *(FP)*, and false negatives *(FN)*. In the object detection context, a true positive is the combination of a predicted bounding box having an Intersection over Union *(IoU)* greater than a specified threshold and a correctly predicted category for the detected object. *IoU* is defined as the ratio between the intersection of the predicted bounding box and the corresponding ground truth and the union of the two. If this ratio is greater than the threshold and the class associated with the bounding box is correct, the prediction is a true positive. If the *IoU* is less than the threshold, or the object is misclassified, we count a false positive. If the ground truth bounding box has no associated prediction, we consider it a false negative. In our case, we use an *IoU* threshold of 0.9 for reliability, completeness, and f1-score metrics. For *mAP@50*, we use a threshold of 0.5, as specified in the name. We chose a different threshold for *mAP@50*, as this is the most common metric used to evaluate object detection models. The Mean Average Precision at 0.50 *mAP@0.5*, defined in Eq. 8, indicates the mean over the average precision (or reliability) for all classes above the specified threshold

of 0.5. To compute such a metric, predictions need first to be ranked, in descending order, and by confidence score value. This is the reason why we can't use this kind of metric for segmentation models, as they intrinsically lack a confidence score for the predicted object. After ranking the N predictions, starting from the one with the highest confidence score, we compute the area under the precision-recall curve for each class c, that we indicate with AP_c. Finally, we compute the mean of the average precision metrics over all classes.

We trained our models with different batch sizes, as some models require higher memory consumption, especially the ones using transformers. The batch size determines how the model is updated: a batch size of B means that the model parameters are updated after performing the forward pass on B elements. We trained our detection models for 300 epochs, using, for testing each model, the weights yielding the best validation loss.

5.1.2 Detection results

In Table 2, we report the results for the described models. Looking at the reliability column, YOLOv4 appears to be the best-performing model, but this high score is counterbalanced by low completeness (~ 0.5). This means that, while the model yields predictions with an IoU score of more than 0.9, such precise predictions concern half of the true objects in the images. The model is missing half of the predictions, so the reliability metric is not enough to evaluate performance. This particular case (high reliability, low completeness) is also the effect of choosing a high IoU threshold (0.9) which makes all the predictions with an IoU below that threshold be considered false negatives, thus increasing the denominator of the Completeness formula (Eq. 6).

A more indicative metric that takes into account both reliability and completeness is the f1-score, so it seems more reasonable to use this metric as a comparison between the methods.

All the methods have been trained from scratch, except DETR, Detectron2, and YOLOS. These models have been fine-tuned starting from weights pre-trained on the COCO [32] dataset. We refer to fine-tuning as training the whole model end-to-end, initialized with pre-trained weights instead of random weights. The choice of not training DETR and YOLOS from scratch stems from the difficulty of reaching convergence with transformers when trained from scratch, compared with the case where it is fine-tuned, as shown in Table 3. Detectron2 has been pre-trained for comparison with Mask R-CNN, which, in contrast, is trained from scratch. This way, we evaluate the impact of training the model starting from learned features against training from randomly initialized weights.

Detectron2 and DETR yield the best results, but this comes at a high computational cost (see Section 5.4). If a compromise between performance and computational budget has to be met, the single-stage detector YOLO or one of the EfficientDet versions may be more suitable.

The models perform a lower score in reliability and completeness compared to computer-vision-based techniques (e.g. CAESAR [22], AEGEAN [18]) but they come with the capability of distinguishing between multiple kinds of sources and detecting extended sources and artifacts. This is the main advantage of using deep learning for

Table 2 Detection metrics

Model	BS	Reliability			Completeness			F1-Score			mAP
		Compact	Extended	Total	Compact	Extended	Total	Compact	Extended	Total	Total
Mask R-CNN	32	48.7%	88.8%	52.0%	82.3%	77.0%	79.5%	61.2%	82.5%	62.9%	70.2%
Detectron2	64	59.8%	62.9%	59.1%	**83.7%**	**90.9%**	**83.9%**	69.7%	74.3%	69.4%	**83.9%**
DETR	2	75.0%	84.6%	76.4%	76.6%	84.9%	76.8%	**75.8%**	84.8%	**76.6%**	79.0%
Yolo v4	64	**97.4%**	**95.9%**	**97.2%**	48.3%	85.5%	50.2%	64.5%	**90.4%**	66.2%	53.8%
Yolo v7	32	87.5%	87.7%	87.4%	60.0%	86.6%	61.0%	69.1%	87.2%	71.7%	61.6%
YOLOS	2	55.9%	78.1%	58.0%	75.0%	84.8%	75.5%	64.1%	81.3%	65.6%	76.3%
EffDet-D1	64	96.1%	0.0%	64.9%	42.2%	0.0%	33.7%	58.6%	0.0%	44.4%	53.5%
EffDet-D2	32	96.7%	0.0%	69.8%	48.5%	0.0%	39.1%	64.6%	0.0%	50.1%	53.8%

YOLOv4 shows the best reliability, but this high value is given by a high IoU threshold. BS stands for batch size. Best results in bold

Table 3 Impact of pretraining for transformer-based models

Model	FT	Rel	Comp	F1 Score	mAP@50
DETR	No	15.3%	18.4%	16.7%	13.4%
	Yes	**76.4%**	**76.8%**	**76.6%**	**78.9%**
YOLOS	No	48.1%	25.9%	33.7%	26.4%
	Yes	**58.0%**	**75.5%**	**65.6%**	**76.3%**

FT stands for fine-tuning: "No" means the model has been trained from scratch, and "Yes" that it is fine-tuned from weights on the COCO Dataset. Best results in bold

source detection, as shown by other object detection methods applied to radio astronomy, for instance, CLARAN [26] and HeTu [33]. Also, computer vision algorithms compute metrics regardless of the predicted object's category, and, in our case, this can be the reason why our performance is lower than this type of approach.

In Section 5.3, we also analyze the performance of these approaches on different Signal-to-Noise Ratio ranges to explore how much the models are capable of dealing with hard-to-detect sources and understand how much faint sources impair detection performance.

5.1.3 Results by telescope type

We perform an analysis of the impact of the image origin on the bias inducted in the models. We split our dataset into three subsets, by telescope type: VLA, ATCA, and ASKAP, in the same way as in [35]. We then evaluate our best-performing object detector, YOLOv7, by creating three separate instances of the model, originating from different training sessions, one for each subset. We test each instance of the model on each subset and on the mixed dataset. We report the performance in terms of F1-score in Table 4. From this analysis, it emerges that each subset injects some degree of bias in the model, and this seems to be especially the case for the VLA subset. The model trained on single subsets yields poor performance when tested on different subsets, due to the distribution shift caused by the different telescopes of origin. When trained on the whole dataset, the model learns the biases of several telescopes, being capable of a higher degree of generalization.

Table 4 Performance on YOLOv7 when trained on different subsets of our dataset

Test (→) Train (↓)	Mixed	VLA	ATCA	ASKAP
Mixed	**71.7%**	**87.09%**	**90.25%**	**87.04%**
VLA	18.56%	84.43%	0.00%	0.27%
ATCA	38.83%	01.32%	59.85%	51.26%
ASKAP	51.26%	02.40%	48.03%	81.33%

Each row refers to an instance of YOLOv7 trained on the specified subset, while columns (2-5) specify the subset where each instance has been evaluated. "Mixed" stands for the whole dataset. Results are reported in terms of F1-score. Best results in bold

5.2 Semantic segmentation

5.2.1 Segmentation metrics

Similarly to how we computed the metrics for the detection models, we introduce reliability, completeness, and the f1 score for the segmentation models. These quantities are computed in the same way, as reported in Eqs. 5, 6 and 7. The main difference between the metrics for the two families of models resides in how we compute true positives, false positives, and false negatives. While for object detection we consider the IoU between the predicted and ground truth boxes to determine the nature of the prediction, for segmentation models, we compare masks. If the model predicts, for pixel i, a class of $\hat{k}$, k is the true class, we will count a true positive if $k = \hat{k}$, a false positive if $k \neq \hat{k}$, and a false negative if $k \neq 0$ and $\hat{k} = 0$. Given the different nature of the metrics, it is not informative to compare object detection models directly with semantic segmentation models, so we will evaluate them separately. We trained our models with a batch size of 32, using Adam [91] as an optimizer, with a learning rate of 10^{-4}, using a cross-entropy loss function. Training has been carried out for 300 epochs, and we selected the weights for the model at the minimum validation loss for evaluation.

5.2.2 Segmentation results

Table 5 reports the comparison between our segmentation models. These results show how the performance increases when progressively adding incremental enhancements, starting from the baseline encoder-decoder model up to the Tiramisu architecture, as we described in the previous sections. This demonstrates that on radio-astronomical data, it is possible to achieve the same improvements that would have been obtained on a canonical, more affirmed dataset. The lower performance achieved by the last model, PankNet, can be due to the following factors: 1) the change in the decoder architecture, replacing the upsampling path with hierarchical decoding; 2) the fact that the model is originally designed for segmenting 3D volumes instead of 2D images. From this analysis it emerges how these models perform better on extended sources with respect to compact sources. This can be caused by the fact that, dealing with deep networks, these can lose more fine-grained details in the feature extraction process, resulting in smaller objects not being detected. The difficulty of detecting smaller objects, especially in the semantic segmentation task, is a common problem [92–94] and this can be exacerbated by the fact that in our case we are using image crops at 132×132 resolution, causing smaller compact sources information to be encoded in a restricted number of pixels.

5.3 SNR performance

In addition to performance in the entire dataset, we evaluated the ability of the models to perform with specific data samples. We divided our test set into three subsets based on their signal-to-noise ratio (SNR), calculated by dividing the peak flux value of each

Table 5 Semantic Segmentation metrics. Best results in bold

Model	Reliability			Completeness			F1-Score		
	Compact	Extended	Total	Compact	Extended	Total	Compact	Extended	Total
U-Net (no-skip)	72.2%	84.4%	76.8%	65.2%	70.5%	68.5%	68.5%	76.8%	72.4%
U-Net	79.4%	86.7%	82.5%	70.4%	76.1%	72.5%	74.6%	81.1%	77.1%
U-Net+DS	79.8%	**93.8%**	83.1%	74.3%	88.3%	76.5%	77.0%	91.0%	79.7%
U-Net++	81.4%	91.5%	84.7%	82.4%	90.1%	80.4%	79.5%	92.7%	84.3%
Tiramisu	**82.2%**	90.9%	**86.5%**	**90.4%**	**97.1%**	**91.4%**	**86.1%**	**93.9%**	**88.9%**
PankNet	77.6%	85.4%	79.5%	68.1%	75.8%	72.5%	72.5%	80.3%	75.8%

Table 6 Object category distribution among SNR splits

Category	SNR < 5	5 < SNR < 20	SNR > 20
Spurious	0	0	267
Compact	175	1505	4206
Extended	5	156	1046

The first two splits lack a significant number of spurious sources, so no significant result should be expected for such splits

cutout by the amount of background noise, as shown in Eq. 9. We selected 2 thresholds for the SNR to split our test set: 5 and 20. The first one is a commonly used threshold in astrophysics to determine a lower bound for source filtering, i.e. sources below this SNR value are not considered. We set this first threshold to explore how models can predict sources below such an SNR value, thus on difficult samples. The threshold of 20 is arbitrary, with the sole purpose of further exploring how much the SNR value affects the performance of the models. Such a split generates a class imbalance, yet this does not affect training, as we train the models on the whole dataset and use this split only for evaluation, thus in inference. In particular, in the first two splits (i.e. SNR < 5 and $5 <$ SNR < 20), there are no spurious sources, so performance for this class in such subsets will necessarily be 0. Details on how the objects are split among the SNR bins are reported in Table 6.

$$SNR = \frac{F_P}{N_{BG}} \tag{9}$$

where N_{BG} is the 3 Sigma Clip of the background pixels and F_P is the peak flux computed on pixel values. The SNR equation that we used is defined in [95].

Table 7 reports the results computed on the different SNR bins. Models perform poorly in the range lower than 5 because samples in this SNR range are more difficult to detect, even using visual inspection, and because, as shown in Table 6, this bin contains a low number of samples. YOLOv7 seems to excel in this case among object detection models, while, in segmentation, Tiramisu and U-Net perform best for SNR values above and below 5, respectively. YOLOv7 outperforms the other models in this scenario because of its high reliability at the 0.9 IoU threshold, reported in Table 2, which raises the f1-score. Such results are enhanced by the prevalence of point sources rather than extended sources, suggesting a better ability of YOLOv7 to detect small objects compared to the other methods. Among the segmentation methods, Tiramisu seems to be the most robust model, even if it may slightly overfit on fainter objects, for which U-Net seems better suited. More detailed results are reported in Appendix.

5.4 Computational resources utilization

In this section, we evaluate the performance of the presented models from the computational point of view. We performed our experiments on a DGX cluster composed of 3 nodes, each of which is equipped with 8 *NVIDIA A100* GPUs. We trained each model across 8 GPUs, using data parallelization by broadcasting a copy of the model

Table 7 Performance evaluation on different SNR bins for each class

Model	SNR < 5			SNR 5 - 20			SNR > 20		
	Compact	Extended	Total	Compact	Extended	Total	Compact	Extended	Total
Detection Models									
Mask R-CNN	45.2%	66.7%	45.0%	62.0%	79.5%	62.8%	61.8%	83.1%	63.8%
Detectron2	70.8%	38.7%	69.1%	79.6%	65.5%	77.7%	66.3%	76.1%	66.9%
DETR	79.7%	66.7%	79.0%	83.0%	76.2%	82.3%	**72.9%**	86.4%	**75.4%**
YOLOv4	87.3%	**80.0%**	88.5%	89.4%	91.6%	90.6%	72.0%	90.4%	74.6%
YOLOv7	**89.1%**	**80.0%**	**89.7%**	**90.8%**	**92.1%**	**91.0%**	72.4%	**91.7%**	75.2%
YOLOS	51.3%	44.4%	50.9%	54.0%	49.7%	50.9%	41.7%	84.4%	43.6%
EffDet-D1	74.7%	0.0%	73.2%	70.6%	0.0%	62.4%	53.1%	0.0%	39.0%
EffDet-D2	76.2%	0.0%	74.6%	79.3%	0.0%	71.7%	57.9%	0.0%	43.3%
Segmentation Models									
U-Net (no-skip)	82.1%	**66.7%**	80.5%	68.0%	64.9%	66.5%	61.5%	84.8%	63.6%
U-Net	**86.8%**	**66.7%**	**85.1%**	73.3%	68.3%	72.5%	67.1%	87.0%	70.5%
U-Net+DS	83.9%	61.5%	82.5%	73.8%	72.9%	71.6%	70.4%	89.6%	73.8%
U-Net++	84.4%	**66.7%**	84.8%	78.6%	77.4%	75.9%	72.3%	90.2%	75.2%
Tiramisu	82.3%	**66.7%**	81.7%	**86.5%**	**88.9%**	**87.9%**	**90.7%**	**91.6%**	**91.9%**
PankNet	64.0%	63.16%	67.2%	63.0%	74.0%	67.9%	70.0%	77.0%	75.8%

Results are expressed as f1 scores. Best results in bold

Table 8 Computing performance on CINECA DGX machine

Model	ETA*	GFLOPs	# of params
Detection models			
Mask R-CNN	180s	198.0	44M
Detectron2	150s	198.0	43.9M
DETR	130s	85.5	41.5M
YOLOv4	90s	48.3	52.5M
YOLOv7	43s	43.7	29.3M
YOLOS	75s	22.7	5.7M
EffDet-D1	52s	6.1	6.6M
EffDet-D2	77s	11.3	8M
Segmentation models			
Encoder-Decoder	50s	14.9	7.0M
U-Net	56s	16.6	7,7M
U-Net + DS	67s	19.8	7.7M
U-Net++	75s	27.4	9.0M
Tiramisu	97s	32.4	3.5M
PankNet	101s	12.8	25.6M

*ETA refers to one epoch

and its parameters to each process, loading a shard of the training batches. At each forward pass, the gradients are reduced among all processes and the same backward pass is performed on all the copies. Such communication operations introduce some overhead that may limit the speed-up of the training process.

In Table 8, we show the computing performance on the CINECA DGX cluster in terms of the average time it takes to complete a training epoch. Along with this metric, we report GFLOPs (i.e. the number of billions of floating point operations that each model requires to make a forward pass) and the total number of parameters of each model.

Most current state-of-the-art models for object detection tasks employ a two-stage detection process based on the Region Proposal Network [27], which performs highly computationally demanding operations. The high number of FLOPs required by Mask R-CNN and Detectron2 is due to their architecture, as they are based on the same model, i.e. Mask R-CNN [30]. YOLOv4, being a single-stage detector, is more computationally efficient in comparison to Mask R-CNN-based ones, even if it requires more parameters.

DETR [42] comes with a less demanding set of operations compared to two-stage detectors (i.e. Mask R-CNN and Detectron2), as it doesn't use an RPN but still represents a burden on some architectures, as it employs transformers at its core. In particular, the use of both transformer encoder and decoder modules, with both self- and cross-attention mechanisms, is where most of the computational time is spent.

YOLOS uses a transformer architecture as well but requires less computational resources compared to DETR, mainly due to its encoder-only design, exploiting the

sequence processing capability of the self-attention mechanism and avoiding the computational burden of the decoder cross-attention.

Among the segmentation approaches, all the U-Net-based ones and the baseline encoder-decoder model require approximately the same amount of parameters. Tiramisu is an exception, and, thanks to its feature reuse, given by its DenseNet blocks, allows for fewer parameters to be optimized in the training process.

6 Conclusions

With the advent of the Square Kilometre Array, automated techniques are needed to efficiently analyze the data deluge expected to be produced by this new generation of astronomical facilities. Deep learning is being evaluated in radio astronomy and shows promising results in source detection and classification.

This work proposes a benchmark reporting performance and computational requirements of several models, both convolutional and transformer-based, for the tasks of object detection and semantic segmentation on radio astronomical images collected from different surveys and telescopes. The results can guide future activities on several radio surveys based on the peculiarities of the available images and computing capacities. However, since deep learning is rapidly evolving, any newly developed model has been possibly omitted but additional releases of the present work are planned in the future, as well as extending the training dataset and classification capabilities. From the analyses we conducted, we noticed that for the object detection task, YOLO-based methods report better overall performance in comparison to other approaches. Transformer-based approaches, even if they show promising results, especially in the case of DETR [42], come with the drawback of computational burden, as the multi-head attention mechanism increases the number of operations. In addition to this, as shown in Table 3, transformer models require a lot of data to be properly trained, e.g. 300M images [41] which can be a problem when training in the radio-astronomical field, for the limited amount of labelled data available for training deep learning models. Also, for images in an SNR range lower than 20, architectures based on YOLO showed the best performance, which can be ideal for cases where most of the images are characterized by lower peak fluxes. The comparison of semantic segmentation models shows that Tiramisu achieves the highest performance without excessively compromising computational speed. The major drawback of this kind of model lies in the format of the annotations required for training. Providing a segmentation mask that classifies each pixel for each image is not a trivial task, as these masks have to be carefully segmented by experts in the field, while bounding box coordinates are easier to provide.

We also aim to address, in future works, the use of Generative Adversarial Network (GAN) architectures to solve a series of open problems in the radio astronomical field, such as the inability to replicate machine learning experiments due to data privacy and the difficulty of simulating annotated data in a non-time-consuming way.

Table 9 Reliability over SNR splits and over the compact and extended classes, as well as over all classes. Best results in bold

Model	SNR < 5			SNR 5 - 20			SNR > 20		
	Compact	Extended	Total	Compact	Extended	Total	Compact	Extended	Total
Detection Models									
Mask R-CNN	31.13%	66.67%	31.45%	47.10%	74.59%	51.15%	50.10%	81.28%	53.18%
Detectron2	57.40%	27.27%	56.72%	69.67%	58.46%	68.81%	53.73%	67.07%	57.51%
DETR	73.30%	66.67%	72.63%	80.60%	72.78%	79.79%	72.89%	86.77%	75.32%
YOLOv4	**87.57%**	**100.00%**	**90.28%**	95.03%	93.74%	**94.76%**	92.07%	94.16%	90.97%
YOLOv7	86.91%	**100.00%**	89.16%	92.52%	**93.91%**	93.64%	94.54%	**94.35%**	**94.33%**
YOLOS	36.79%	50.00%	36.88%	38.83%	72.54%	37.15%	29.19%	84.20%	41.82%
EffDet D1	86.59%	0.00%	84.39%	96.82%	0.00%	82.11%	96.38%	0.00%	58.45%
EffDet D2	87.13%	0.00%	85.44%	**97.72%**	0.00%	86.54%	**96.71%**	0.00%	62.52%
Segmentation Models									
U-Net (no-skip)	86.16%	27.27%	84.83%	56.83%	76.85%	54.89%	54.37%	87.62%	64.81%
U-Net	91.44%	**66.67%**	89.07%	63.97%	78.54%	62.17%	62.51%	89.52%	67.91%
U-Net+DS	91.59%	**66.67%**	**91.40%**	63.52%	**90.52%**	65.13%	68.44%	93.69%	73.48%
U-Net++	**91.68%**	**66.67%**	**91.67%**	71.38%	89.15%	68.73%	71.39%	94.61%	76.31%
Tiramisu	83.18%	50.00%	83.24%	**82.49%**	89.29%	**84.64%**	**94.88%**	**95.71%**	**93.84%**
PankNet	56.66%	**66.67%**	62.39%	54.06%	82.17%	57.13%	66.81%	77.49%	75.36%

Table 10 Completeness over SNR splits and over the compact and extended classes, as well as over all classes. Best results in bold

Model	SNR < 5			SNR 5 - 20			SNR > 20		
	Compact	Extended	Total	Compact	Extended	Total	Compact	Extended	Total
Detection Models									
Mask R-CNN	82.52%	**66.67%**	78.95%	90.82%	85.03%	81.46%	80.52%	84.98%	79.63%
Detectron2	**92.52%**	**66.67%**	88.47%	**92.82%**	74.49%	**89.28%**	**86.69%**	87.85%	**80.07%**
DETR	87.28%	**66.67%**	86.52%	85.53%	79.87%	84.95%	72.89%	85.95%	75.48%
YOLOv4	87.06%	**66.67%**	86.71%	84.46%	**90.55%**	86.72%	59.15%	86.90%	63.19%
YOLOv7	91.40%	**66.67%**	**90.25%**	89.14%	90.36%	88.50%	58.66%	**89.19%**	62.52%
YOLOS	84.95%	40.00%	82.17%	88.50%	37.81%	80.61%	72.86%	84.52%	45.56%
EffDet D1	65.74%	0.00%	64.55%	55.51%	0.00%	50.31%	36.66%	0.00%	29.22%
EffDet D2	67.69%	0.00%	66.17%	66.72%	0.00%	61.26%	41.33%	0.00%	33.14%
Segmentation Models									
Encoder-Decoder	78.46%	66.67%	76.52%	84.69%	56.18%	84.18%	70.88%	82.16%	62.40%
U-Net	**82.52%**	66.67%	**81.50%**	85.85%	60.36%	86.89%	72.52%	84.56%	73.27%
U-Net+DS	77.45%	66.67%	75.16%	88.16%	61.08%	79.42%	72.45%	85.89%	74.12%
U-Net++	78.19%	66.67%	78.89%	87.44%	68.39%	84.74%	73.23%	86.18%	74.12%
Tiramisu	81.52%	**80.00%**	80.22%	**90.81%**	**88.57%**	**91.44%**	**86.82%**	**87.87%**	**89.98%**
PankNet	73.52%	66.67%	72.88%	75.48%	67.31%	83.70%	73.52%	76.51%	76.27%

Acknowledgements We thank the authors of the following software tools and libraries that have been extensively used in this work: CAESAR [22, 96], astropy [97, 98], ds9 [99], DVC. We also acknowledge the contribution made by Andrea Pilzer in improving this work.

Author Contributions Renato Sortino wrote the main manuscript text. Eva Sciacca wrote the abstract, introduction, and conclusions. Simone Riggi and Cristobal Bordiu helped shape Section 3 with insight on the astrophysical side. Daniel Magro wrote the Related Works section as well as Section 4.1.1 describing the Mask R CNN model, and contributed to Tables 2, 6, 7, and A1 by running experiments on Mask R CNN. Giuseppe Fiameni wrote Section 4.1.2 on Detectron2, and contributed to Tables 2, 6, 7, and A1 running experiments on the same architecture. Andrew Hopkins contributed with improvements from the astrophysics perspective in Section 3 and Section 5. All the authors reviewed the manuscript.

Funding The research leading to these results has received funding from the Horizon 2020 research and innovation programme of the European Commission under grant agreement No. 863448 (NEANIAS), from the INAF PRIN TEC programme (CIRASA), and from the MOSAICo (Metodologie Open Source per l'Automazione Industriale e delle procedure di CalcOlo in astrofisica) project, co-funded by the EU, Fondo Europeo di Sviluppo Regionale - Programma Operativo Nazionale Imprese e Competitivitá 2014-2020.

Declarations

Competing Interests I declare that the authors have no competing interests as defined by Springer, or other interests that might be perceived to influence the results and/or discussion reported in this paper.

Appendix: Additional results

We report additional results, in Tables 9 and 10, showing reliability and completeness, split by SNR value, to give a comprehensive view of the performance of our models, and also to make them comparable to traditional source detection algorithms.

References

1. Magro, D., Zarb Adami, K., DeMarco, A., Riggi, S., Sciacca, E.: A comparative study of convolutional neural networks for the detection of strong gravitational lensing. Mon. Not. R. Astron. Soc. **505**(4), 6155–6165 (2021)
2. Ralph, N.O., Norris, R.P., Fang, G., Park, L.A., Galvin, T.J., Alger, M.J., Andernach, H., Lintott, C., Rudnick, L., Shabala, S., et al.: Radio galaxy zoo: Unsupervised clustering of convolutionally auto-encoded radio-astronomical images. Publ. Astron. Soc. Pac. **131**(1004), 108011 (2019)
3. Karypidou, S., Georgousis, I., Papakostas, G.A.: Computer vision for astronomical image analysis. In: 2021 IEEE International Conference on Progress in Informatics and Computing (PIC), pp. 94–101 (2021). https://doi.org/10.1109/PIC53636.2021.9687023
4. Connor, L., Bouman, K.L., Ravi, V., Hallinan, G.: Deep radio-interferometric imaging with POLISH: DSA-2000 and weak lensing. Mon. Not. R. Astron. Soc. **514**(2), 2614–2626 (2022). https://academic.oup.com/mnras/article-pdf/514/2/2614/44147595/stac1329.pdf. https://doi.org/10.1093/mnras/stac1329
5. Treml, M., Arjona-Medina, J., Unterthiner, T., Durgesh, R., Friedmann, F., Schuberth, P., Mayr, A., Heusel, M., Hofmarcher, M., Widrich, M., et al.: Speeding up semantic segmentation for autonomous driving (2016)
6. Feng, D., Haase-Schütz, C., Rosenbaum, L., Hertlein, H., Glaeser, C., Timm, F., Wiesbeck, W., Dietmayer, K.: Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. IEEE Trans. Intell. Transp. Syst. **22**(3), 1341–1360 (2020)

7. Chen, X., Ma, H., Wan, J., Li, B., Xia, T.: Multi-view 3d object detection network for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1907–1915 (2017)
8. Siam, M., Gamal, M., Abdel-Razek, M., Yogamani, S., Jagersand, M., Zhang, H.: A comparative study of real-time semantic segmentation for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 587–597 (2018)
9. Kayalibay, B., Jensen, G., van der Smagt, P.: Cnn-based segmentation of medical imaging data. arXiv:1701.03056 (2017)
10. Asgari Taghanaki, S., Abhishek, K., Cohen, J.P., Cohen-Adad, J., Hamarneh, G.: Deep semantic segmentation of natural and medical images: a review. Artif. Intell. Rev. **54**(1), 137–178 (2021)
11. Proietto Salanitri, F., Bellitto, G., Irmakci, I., Palazzo, S., Bagci, U., Spampinato, C.: Hierarchical 3d feature learning forpancreas segmentation. In: International Workshop on Machine Learning in Medical Imaging, pp. 238–247. Springer (2021)
12. Raghunandan, A., Raghav, P., Aradhya, H.R., et al.: Object detection algorithms for video surveillance applications. In: 2018 International Conference on Communication and Signal Processing (ICCSP), pp. 0563–0568. IEEE (2018)
13. Jha, S., Seo, C., Yang, E., Joshi, G.P.: Real time object detection and trackingsystem for video surveillance system. Multimed. Tools Appl. **80**(3), 3981–3996 (2021)
14. Astua, C., Barber, R., Crespo, J., Jardon, A.: Object detection techniques applied on mobile robot semantic navigation. Sensors **14**(4), 6734–6757 (2014)
15. Hernández, A.C., Gómez, C., Crespo, J., Barber, R.: Object detection applied to indoor environments for mobile robot navigation. Sensors **16**(8), 1180 (2016)
16. Miyamoto, R., Adachi, M., Nakamura, Y., Nakajima, T., Ishida, H., Kobayashi, S.: Accuracy improvement of semantic segmentation using appropriate datasets for robot navigation. In: 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT), pp. 1610–1615. IEEE (2019)
17. Whiting, M.T.: duchamp: a 3D source finder for spectral-line data. Mon. Not. R. Astron. Soc. **421**(4), 3242–3256 (2012)
18. Hancock, P.J., Trott, C.M., Hurley-Walker, N.: Source finding in the era of the ska (precursors): Aegean 2.0. Publ. Astron. Soc. Aust. **35**, 011 (2018). https://doi.org/10.1017/pasa.2018.3
19. Carbone, D., Garsden, H., Spreeuw, H., Swinbank, J.D., van der Horst, A.J., Rowlinson, A., Broderick, J.W., Rol, E., Law, C., Molenaar, G., Wijers, R.A.M.J.: Pyse: Software for extracting sources from radio images. Astronomy and Computing **23**, 92–102 (2018). https://doi.org/10.1016/j.ascom.2018.02.003
20. Mohan, N.R., Rafferty, D.A.: Pybdsf: Python blob detection and source finder. (2015)
21. Friedlander, A., Frean, M., Johnston-Hollitt, M., Hollitt, C.: Latent dirichlet allocation for image segmentation and source finding in radio astronomy images. In: Proceedings of the 27th Conference on Image and Vision Computing New Zealand, pp. 429–434 (2012)
22. Riggi, S., Vitello, F., Becciani, U., Buemi, C., Bufano, F., Calanducci, A., Cavallaro, F., Costa, A., Ingallinera, A., Leto, P., et al.: Caesar source finder: Recent developments and testing. Publ. Astron. Soc. Aust. **36** (2019)
23. Schilliro', F., Romano, P.: Segmentation of spectroscopic images of the low solar atmosphere by the self-organizing map technique. Mon. Not. R. Astron. Soc. **503**(2), 13 (2021)
24. Lukic, V., de Gasperin, F., Brüggen, M.: Convosource: radio-astronomical source-finding with convolutional neural networks. Galaxies **8**(1), 3 (2020)
25. Vafaei Sadr, A., Vos, E.E., Bassett, B.A., Hosenie, Z., Oozeer, N., Lochner, M.: Deepsource: point source detection using deep learning. Mon. Not. R. Astron. Soc. **484**(2), 2793–2806 (2019)
26. Wu, C., Wong, O.I., Rudnick, L., Shabala, S.S., Alger, M.J., Banfield, J.K., Ong, C.S., White, S.V., Garon, A.F., Norris, R.P., et al.: Radio galaxy zoo: Claran-a deep learning classifier for radio morphologies. Mon. Not. R. Astron. Soc. **482**(1), 1211–1230 (2019)
27. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. Adv. Neural Inf. Proces. Syst. **28** (2015)
28. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. Int. J. Comput. Vis. **115**(3), 211–252 (2015)

29. Burke, C.J., Aleo, P.D., Chen, Y.-C., Liu, X., Peterson, J.R., Sembroski, G.H., Lin, J.Y.-Y.: Deblending and classifying astronomical sources with mask r-cnn deep learning. Mon. Not. R. Astron. Soc. **490**(3), 3952–3965 (2019)
30. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2961–2969 (2017)
31. Farias, H., Ortiz, D., Damke, G., Arancibia, M.J., Solar, M.: Mask galaxy: Morphological segmentation of galaxies. Astronomy and Computing, 100420 (2020)
32. Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European Conference on Computer Vision, pp. 740–755. Springer (2014)
33. Lao, B., An, T., Wang, A., Xu, Z., Guo, S., Lv, W., Wu, X., Zhang, Y.: Artificial intelligence for celestial object census: the latest technology meets the oldest science. arXiv:2107.03082 (2021)
34. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
35. Riggi, S., Magro, D., Sortino, R., De Marco, A., Bordiu, C., Cecconello, T., Hopkins, A., Marvil, J., Umana, G., Sciacca, E., et al.: Astronomical source detection in radio continuum maps with deep neural networks. Astronomy and Computing **42**, 100682 (2023)
36. Bochkovskiy, A., Wang, C.-Y., Liao, H.-Y.M.: Yolov4: Optimal speed and accuracy of object detection. arXiv:2004.10934 (2020)
37. Wang, C.-Y., Bochkovskiy, A., Liao, H.-Y.M.: Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arXiv:2207.02696 (2022)
38. Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., Girshick, R.: Detectron2. https://github.com/facebookresearch/detectron2 (2019)
39. Tan, M., Pang, R., Le, Q.V.: Efficientdet: Scalable and efficient object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10781–10790 (2020)
40. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Adv. Neural Inf. Proces. Syst. **30** (2017)
41. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv:2010.11929 (2020)
42. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European Conference on Computer Vision, pp. 213–229. Springer (2020)
43. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: International Conference on Machine Learning, pp. 10347–10357. PMLR (2021)
44. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788 (2016)
45. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning, pp. 6105–6114. PMLR (2019)
46. Fang, Y., Liao, B., Wang, X., Fang, J., Qi, J., Wu, R., Niu, J., Liu, W.: You only look at one sequence: Rethinking transformer in vision through object detection. Adv. Neural Inf. Proces. Syst. **34**, 26183–26197 (2021)
47. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical Image Computing and Computer-assisted Intervention, pp. 234–241. Springer (2015)
48. Zhu, Q., Du, B., Turkbey, B., Choyke, P.L., Yan, P.: Deeply-supervised cnn for prostate segmentation. In: 2017 International Joint Conference on Neural Networks (IJCNN), pp. 178–184. IEEE (2017)
49. Zhou, Z., Rahman Siddiquee, M.M., Tajbakhsh, N., Liang, J.: Unet++: A nested u-net architecture for medical image segmentation. In: Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 20, 2018, Proceedings 4, pp. 3–11. Springer (2018)
50. Jégou, S., Drozdzal, M., Vazquez, D., Romero, A., Bengio, Y.: The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 11–19 (2017)

51. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4700–4708 (2017)
52. Luo, Z., Wang, Y., Liu, S., Peng, J.: Hierarchical encoder-decoder with soft label-decomposition for mitochondria segmentation in em images. Front. Euroscience **15**, 687832 (2021)
53. Pino, C., Sortino, R., Sciacca, E., Riggi, S., Spampinato, C.: Semantic segmentation of radio-astronomical images. In: International Workshop on Artificial Intelligence and Pattern Recognition, pp. 393–403. Springer (2021)
54. Hopkins, A.M., Whiting, M.T., Seymour, N., Chow, K., Norris, R.P., Bonavera, L., Breton, R., Carbone, D., Ferrari, C., Franzen, T., et al.: The askap/emu source finding data challenge. Publ. Astron. Soc. Aust. **32** (2015)
55. Bonaldi, A., An, T., Brüggen, M., Burkutean, S., Coelho, B., Goodarzi, H., Hartley, P., Sandhu, P., Wu, C., Yu, L., et al.: Square kilometre array science data challenge 1: analysis and results. Mon. Not. R. Astron. Soc. **500**(3), 3821–3837 (2021)
56. Banfield, J.K., Wong, O., Willett, K.W., Norris, R.P., Rudnick, L., Shabala, S.S., Simmons, B.D., Snyder, C., Garon, A., Seymour, N., et al.: Radio galaxy zoo: host galaxies and radio morphologies derived from visual inspection. Mon. Not. R. Astron. Soc. **453**(3), 2326–2340 (2015)
57. Becker, R.H., White, R.L., Helfand, D.J.: The first survey: faint images of the radio sky at twenty centimeters. Astrophys. J. **450**, 559 (1995)
58. Umana, G., Trigilio, C., Ingallinera, A., Riggi, S., Cavallaro, F., Marvil, J., Norris, R.P., Hopkins, A.M., Buemi, C.S., Bufano, F., Leto, P., Loru, S., Bordiu, C., Bunton, J.D., Collier, J.D., Filipovic, M., Franzen, T.M.O., Thompson, M.A., Andernach, H., Carretti, E., Dai, S., Kapinska, A., Koribalski, B.S., Kothes, R., Leahy, D., Mcconnell, D., Tothill, N., Michalowski, M.J.: A first glimpse at the Galactic plane with the ASKAP: the SCORPIO field. Mon. Not. R. Astron. Soc. **506**(2), 2232–2246 (2021). https://academic.oup.com/mnras/article-pdf/506/2/2232/39306586/stab1279.pdf. https://doi.org/10.1093/mnras/stab1279
59. Norris, R.P., Marvil, J., Collier, J.D., Kapińska, A.D., O'Brien, A.N., Rudnick, L., Andernach, H., Asorey, J., Brown, M.J., Brüggen, M., et al.: The evolutionary map of the universe pilot survey. Publ. Astron. Soc. Aust. **38** (2021)
60. Umana, G., et al.: Scorpio: a deep survey of radio emission from the stellar life-cycle. MNRAS **454**, 902–912 (2015)
61. Zwitter, T., Munari, U.: An Introduction to Analysis of Single Dispersion Spectra with IRAF, vol. 1, (2000)
62. Uijlings, J.R., Van De Sande, K.E., Gevers, T., Smeulders, A.W.: Selective search for object recognition. Int. J. Comput. Vis. **104**(2), 154–171 (2013)
63. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A.C.: Ssd: Single shot multibox detector. In: European Conference on Computer Vision, pp. 21–37. Springer (2016)
64. Chen, C.-F.R., Fan, Q., Panda, R.: Crossvit: Cross-attention multi-scale vision transformer for image classification. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 357–366 (2021)
65. Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., Jégou, H.: Going deeper with image transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 32–42 (2021)
66. He, X., Chen, Y., Lin, Z.: Spatial-spectral transformer for hyperspectral image classification. Remote Sens. **13**(3), 498 (2021)
67. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. arXiv:2010.04159 (2020)
68. Sun, Z., Cao, S., Yang, Y., Kitani, K.M.: Rethinking transformer-based set prediction for object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 3611–3620 (2021)
69. Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., Zhou, M.: Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. Adv. Neural Inf. Proces. Syst. **33**, 5776–5788 (2020)
70. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587 (2014)

71. Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2117–2125 (2017)
72. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In: International Conference on Machine Learning, pp. 1310–1318. PMLR (2013)
73. Girshick, R., Radosavovic, I., Gkioxari, G., Dollár, P., He, K.: Detectron. (2018). https://github.com/facebookresearch/detectron
74. Massa, F., Girshick, R.: maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch. (2018). Accessed: https://github.com/facebookresearch/maskrcnn-benchmark
75. Lin, T.-Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2980–2988 (2017)
76. Güler, R.A., Neverova, N., Kokkinos, I.: Densepose: Dense human pose estimation in the wild. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7297–7306 (2018)
77. Cai, Z., Vasconcelos, N.: Cascade r-cnn: Delving into high quality object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6154–6162 (2018)
78. Kirillov, A., Girshick, R., He, K., Dollár, P.: Panoptic feature pyramid networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6399–6408 (2019)
79. Chen, X., Girshick, R., He, K., Dollár, P.: Tensormask: A foundation for dense object segmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 2061–2069 (2019)
80. Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7263–7271 (2017)
81. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning, pp. 448–456. PMLR (2015)
82. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. arXiv:1804.02767 (2018)
83. Wang, C.-Y., Liao, H.-Y.M., Wu, Y.-H., Chen, P.-Y., Hsieh, J.-W., Yeh, I.-H.: Cspnet: A new backbone that can enhance learning capability of cnn. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pp. 390–391 (2020)
84. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Trans. Pattern Anal. Mach. Intell. **37**(9), 1904–1916 (2015)
85. Liu, S., Qi, L., Qin, H., Shi, J., Jia, J.: Path aggregation network for instance segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8759–8768 (2018)
86. Ghiasi, G., Lin, T.-Y., Le, Q.V.: Nas-fpn: Learning scalable feature pyramid architecture for object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7036–7045 (2019)
87. Lei Ba, J., Kiros, J.R., Hinton, G.E.: Layer normalization. ArXiv e-prints, 1607 (2016)
88. Zhou, D., Fang, J., Song, X., Guan, C., Yin, J., Dai, Y., Yang, R.: Iou loss for 2d/3d object detection. In: 2019 International Conference on 3D Vision (3DV), pp. 85–94. IEEE (2019)
89. Liu, L., Liu, X., Gao, J., Chen, W., Han, J.: Understanding the difficulty of training transformers. arXiv:2004.08249 (2020)
90. Hendrycks, D., Gimpel, K.: Gaussian error linear units (gelus). arXiv:1606.08415 (2016)
91. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv:1412.6980 (2014)
92. Pihlak, R., Riid, A.: Morphological cross entropy loss for improved semantic segmentation of small and thin objects. Procedia Comput. Sci. **192**, 582–591 (2021)
93. Dong, R., Pan, X., Li, F.: Denseu-net-based semantic segmentation of small objects in urban remote sensing images. IEEE Access **7**, 65347–65356 (2019). https://doi.org/10.1109/ACCESS.2019.2917952
94. Ma, A., Wang, J., Zhong, Y., Zheng, Z.: Factseg: Foreground activation-driven small object semantic segmentation in large-scale remote sensing imagery. IEEE Trans. Geosci. Remote Sens. **60**, 1–16 (2022). https://doi.org/10.1109/TGRS.2021.3097148
95. Condon, J.: Errors in elliptical gaussian fits. Publ. Astron. Soc. Pac. **109**(732), 166 (1997)
96. Riggi, S., Ingallinera, A., Leto, P., Cavallaro, F., Bufano, F., Schillirò, F., Trigilio, C., Umana, G., Buemi, C.S., Norris, R.P.: Automated detection of extended sources in radio maps: progress from the scorpio survey. Mon. Not. R. Astron. Soc. **460**(2), 1486–1499 (2016)

97. Robitaille, T.P., Tollerud, E.J., Greenfield, P., Droettboom, M., Bray, E., Aldcroft, T., Davis, M., Ginsburg, A., Price-Whelan, A.M., Kerzendorf, W.E., et al.: Astropy: A community python package for astronomy. Astron. Astrophys. **558**, 33 (2013)
98. Price-Whelan, A.M., Sipőcz, B., Günther, H., Lim, P., Crawford, S., Conseil, S., Shupe, D., Craig, M., Dencheva, N., Ginsburg, A., et al.: The astropy project: building an open-science project and status of the v2. 0 core package. Astron. J. **156**(3), 123 (2018)
99. Joye, W.A., Mandel, E.: New Features of SAOImage DS9. In: Payne, H.E., Jedrzejewski, R.I., Hook, R.N. (eds.) Astronomical Data Analysis Software and Systems XII. Astronomical Society of the Pacific Conference Series, vol. 295, p. 489 (2003)