Galaxy Detection Machine Learning Project

Alexander Khalil Arwadi, Aurelio Noca, Louis Jaugey
Supervised by Emma Tolley, LASTRO
CSS-433 (Machine Learning), School of Computer and Communication Sciences, EPFL

***Abstract*—Artificial intelligence has proven to be increasingly useful in all types of industries and scientific fields. In particular, machine learning applications can be used for the purpose of object detection and classification in images. In this paper, we use the Square Kilometer Array (SKA) Science Data Challenge 1 (SDC1) data set to train a model that can detect galaxies in images. The importance of this task is in automating the process of galaxy detection in telescope images, especially that the number of galaxies is extremely high in these types of images. For this purpose, we try different approaches based on convolutional neural networks (CNN). The scope of the project covers the classification of images based on the presence or absence of galaxies in them. In this paper, we study the correlation between the noise threshold used for detecting a galaxy, and the performance of our binary classifier.**

I. Introduction

SKA data challenges are regularly issued to the science community for multiple purposes such as source finding, source property characterization or source population characterization. The data provided by SKA are images of the universe taken by the world's largest radio telescope. These images may consist of real data from currently operating radio facilities or of simulated SKA data.

For this project, we use the SKA SDC1 [1] data set to train a machine learning model that is capable of detecting galaxies in these images. For this purpose, the pre-processing techniques used to clean the data are presented. Then, the details of the model and the machine learning approach used are discussed. Finally, the performance of the algorithm is evaluated along with an analysis of our findings.

II. Data preprocessing

A. Data description

The SKA SDC1 data set consists of nine simulated SKA images files, spanning over 3 frequencies of the Mid Band (560 MHz, 1.4 GHz, 9.2 GHz). Each frequency has three associated integration times : 8h, 100h, 1000h. The images are 32768×32768 pixels, centered at right ascension angle of $RA = 0°$ and declination of $DEC = -30°$. The simulated observations are stored as FITS images, a format commonly used in astronomy [2]. Each frequency also has an associated training set file containing the properties of the galaxies present in a small subset of the images, such as galaxy centroid positions in pixels, major and minor axis, integrated flux, to cite a few. These characteristics can be used as the labels the model will be trained to predict. The simulated images and the corresponding training files can be found at [1].

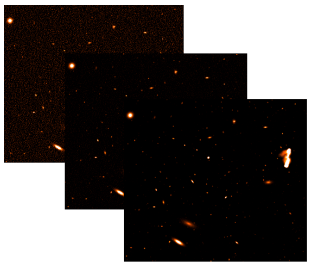


Figure 1: Zoom-in of the 1.4 GHz maps, showing the same region of the sky with different telescope integration: 8, 100, 1000 h left to right [1]

B. Data cleaning

Firstly, we choose to use only the images with integration time of 1000h as they present less noise than the others. The training set files contain galaxies that are not uniformly spread out over the entire span of the images. They are actually all contained within $\sim 1\%$ of the simulated field, near the center. It is important to separate this part of the image from the rest, as it could lead to a model trained on wrongly labeled data. This area covered in labeled galaxies is split into a set of smaller $N \times N$ pixels images. They are then standardised using Z-normalisation, as statistical normalisation has been proven to increase the performance of classification models by removing statistical skewness [3]:

$$Z = \frac{x - \mu_{noise}}{\sigma_{noise}} \tag{1}$$

The 3 training set files also contain the different characteristics defining each galaxy. We dispose of the properties we do not need and only keep the `FLUX` (integrated flux of the galaxy in [Jy]), `BMAJ` (major axis in [arcsec]), `BMIN` (minor axis in [arcsec]), `PA` (principle angle in [degree]), the `X` and `Y` (position in [pixel]), as well as keep the Galaxies with a `SELECTION` value of 1.

We are also able to compute the area each galaxy occupies on the image using the major and minor axes. The studied area of the simulated field is full of galaxies of different sizes and intensities, some of which are not labeled as they are too faint. All of these make up the background noise present in each image. For a galaxy detection model, it is interesting to

filter out some galaxies under a certain threshold of intensity and train the model on the remaining galaxies. For this, we define a filter threshold

$$f = \mu_{noise} + k * \sigma_{noise} \tag{2}$$

where f [Jy/pixel] is the filter flux density threshold, $k \in \{0, 0.1, 0.2, ..., 0.9, 1\}$ and μ_{noise} [Jy/pixel] and σ_{noise} [Jy/pixel] are the mean and standard deviation of the center area of the images containing the training examples. To apply this threshold to the remainder of the training set files, we compute the area of the galaxies to obtain the average flux density of each galaxy:

$$A = \pi \cdot \frac{B_{MAJ}}{2} \cdot \frac{B_{MIN}}{2} \tag{3}$$

For each frequency, there is a conversion factor between arcsecond and pixel given by 0.60, 0.24, and 0.037 arcsec/pixel for 560 MHz, 1.4 GHz and 9.2 GHz respectively.

Hence, the average flux density $\bar{f}$ of a galaxy is

$$\bar{f} = \frac{FLUX}{A} \tag{4}$$

C. Data labeling

To detect a galaxy, we must find a way of defining the position and the shape of the galaxy. This is done by using bounding boxes. Each bounding box needs a center position (x, y), a width w and a height h. For an object detection model, one adds the confidence value, a number between 0 and 1 indicating the confidence in the bounding box. Hence, each training image has a label vector of 5 elements:

1) Galaxy presence: 0 for background, 1 for galaxy
2) Galaxy centroid x value
3) Galaxy centroid y value
4) Galaxy bounding box height value
5) Galaxy bounding box width value

The last 4 labels are assigned a NaN value if the image does not contain a galaxy, or in other words, if the first label is 0.

III. Models and Methods

A. The YOLO approach

The You Only Look Once (YOLO) [4] approach is a clever convolutional neural network approach used in computer vision for object detection. Instead of using the sliding windows approach, where one chooses a box and slides it over the image, predicting an output after every stride, the YOLO approach splits the input image into a w x h grid. Each element of that grid can predict its own label vector, as presented in II-C, giving the prediction done for one sub-image. This allows YOLO to be much faster computationally at object detection.

B. Initial Observations

Originally, the images of the simulated field were split into smaller images of size 128 by 128 pixels. These images were cut into a 16 x 16 grid of 8x8 images. A filter f of threshold k=0.1 was applied to remove some of the noise. Results showed that our model was not able to correctly detect galaxies in the images. We hypothesized four possible reasons for this outcome:

- A large number of the galaxies present in the images are very small (on the order of 1 pixel in size), creating a skewness of the data and affecting the model.
- The data set was not balanced enough, with significantly more non-galaxies than galaxies.
- The training set was not precise enough to obtain good training data. Indeed, the training bounding boxes did not fit the galaxies very well.
- Finally, our model may not have been complicated enough to grasp the intricacies of the data set to allow for galaxy detection.

We attempted to mitigate the first and second problem by creating our own loss function and using different costs for the different label elements, in a similar manner to how it was done in the original YOLO paper [4]. After a few weeks of time spent on trying to solve this problem, we decided to focus on a smaller sub-problem : classifying the presence / absence of galaxies in some given images, i.e. binary classification of galaxies.

C. Binary classification of galaxies

Instead of considering larger images of 128 by 128 with a subgrid, we consider each individual subgrid as an image. We now have input images of size 8 by 8 pixels, each one with an associated label $y \in \{0, 1\}$ indicating whether it contains a galaxy (corresponds to 1) or not (corresponds to 0). To mitigate the data imbalance problem, the data set was augmented using the following geometric transformations: rotations by 0, 90, 180 and 270 degrees of the original images, and flipping combined with the rotation by 90, 180 and 270 degrees. This results in 7 new images for each positive sample.

D. GalaxyNet and loss

The machine learning model used for the purpose of this paper is a two dimensional convolutional neural network. The neural network, named GalaxyNet, used for training and predicting is described in Table I. Batch normalization layers are added as they accelerate training [5].

We use a modified form of the mean square error (MSE) loss defined for a batch of size m :

$$L = \frac{1}{m_1} \sum_{i=1}^{m} \delta(y_i)(y_i - \hat{y}_i)^2 + \frac{1}{m_2} \sum_{i=1}^{m} \delta(y_i - 1)(y_i - \hat{y}_i)^2 \tag{5}$$

where $y_i \in \{0, 1\}$ is the label of example i, $\hat{y}_i$ is the i^{th} prediction, $m_1 = \sum_{i=1}^{m} \delta(y_i)$ is the number of non-galaxies in the batch, $m_2 = m - m_1$ is the number of galaxies in the batch. This allows to mitigate some of the effects of an imbalanced data set.

Table I: Architecture and characteristics of GalaxyNet

Layer	Layer name	Characteristics
1	Conv2d	64 kernels of size 2x2
2	BatchNorm2d	64 features
3	Leaky ReLU	Negative slope: 0.1
4	Conv2d	128 kernels of size 2x2
5	BacthNorm2d	128 features
6	Leaky ReLU	Negative slope: 0.1
7	Maxpool2d	Kernels of size 2
8	Conv2d	128 kernels of size 3x3
9	BatchNorm2d	128 features
10	Leaky ReLU	Negative slope: 0.1
11	Flatten	-
12	Linear	Input size: 128, output size: 128
13	Leaky ReLU	Negative slope: 0.1
14	Linear	Input size: 128, output size: 64
15	Leaky ReLU	Negative slope: 0.1
16	Linear	Input size: 64, output size: 1
17	Sigmoid	General Sigmoid function

IV. Results

80% of the data set is used for training and the validation is performed on the remaining 20%. We use mini-batch gradient descent with a batch size of 1024 and train with a learning rate $\gamma = 0.1$ over 3 epochs followed by $\gamma = 0.01$ over 2 epochs. The training is done twice for all values of $k = \{0, 0.1, 0.2, ..., 0.9, 1\}$: once for the non-augmented data and once for augmented data. The output is predicted to be a galaxy if $\hat{y} > 0.5$ and not a galaxy otherwise. Cross-validation was not performed as it would have generate a huge amount of data. Indeed, as it will be discussed more in detail later, a single value such as accuracy is far from enough to evaluate the performance of this model.

The results obtained are stored in Table II for the non-augmented data set and in Table III for the augmented data set.

A. Evaluation criteria

The criteria used to evaluate our model are the sensitivity/true positive rate (TPR), specificity/true negative rate (TNR), precision (PPV), accuracy (ACC) and the receiver operating characteristic (ROC) curve. Let TP be the true positives, TN the true negatives, FP the false positives, FN the false negatives and TOT the number of sample. Then the previous metrics are defined as:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \qquad \mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} \tag{6}$$

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \qquad \mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TOT}} \tag{7}$$

B. Findings and Analysis

Table II and III summarize most of the results obtained with the non-augmented and the augmented data sets respectively. Note that it was chosen to not augment images from the 560 MHz frequencies when $k = 0$ as it contains enough positive samples to skew the data set in the opposite way when augmented. This is why the total number of samples is smaller than with $k = 0.1$.

The interesting values are the ones linked to positive predictions. Indeed, this algorithm would be used to help detect galaxies, and the negative predictions would be initially discarded. The true positive over false positive ratio is therefore paramount to the evaluation of the model. The results shows that this ratio decreases when k increases. Surprisingly, when $k = 0$ the ratio is lower with data augmentation but it decreases much faster without it. This is also true of the precision. Data augmentation made a more robust model in this regard. The true positive/false positive ratio shows that the model is more interesting to use with lower k values.

Of course, we want the algorithm to be at least better than luck. As the accuracy shows, this is the case for lower values of k but not for higher k. In fact, this probably comes from the fact that the accuracy is boosted in lower k values by the true positives because the algorithm is more confident in predicting galaxies, since the data set contains a lot of them. As for the true positive/false negative ratio and the precision, the accuracy decreases much faster without data augmentation for increasing k. This again shows that the model is better with low values for k. It might be interesting as a future experiment to continue to increase the threshold value k, as it may result in having only a specific kind of galaxy. The model may then be able to perform better than with more diverse galaxy types.

The training set is expected to be a major component in the model's mixed results. Indeed, looking at the pictures with a naked eye, one can see that many galaxies had a size quite different from what the provided data told. For this reason, some galaxies whose provided size is much smaller than their actual size can easily satisfy the filter condition, since the flux density increases with smaller area, even if they shouldn't. This probably leads to a big source of confusion for the model, since some big and clear galaxies are labeled as negative examples while point (size of 1 pixel) galaxies are labeled as positive. This is a possible explanation to why the model performs so much better when no filter is applied (i.e. when $k = 0$). The model may also not have been rich enough to correctly grasp the intricacies of the data that was presented to it.

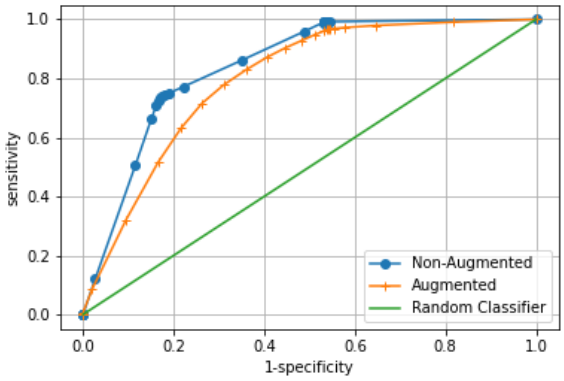
ROC curves visible on fig. 2a shows a clear improvement in comparison to a random classifier, when $k = 0$. As Table II and III suggested, the model seems better without data augmentation with $k = 0$ (the perfect classifier would be a

Table II: Model results using the non-augmented test set

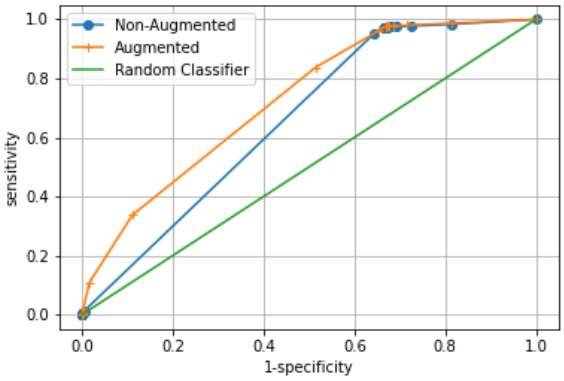
k	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
True Positives (TP)	23048	9079	5977	4281	3086	2310	1734	1437	1145	844	699
True Negatives (TN)	58043	47999	39892	34221	34221	34017	34041	33841	34302	34331	34360
False Positives (FP)	14385	44115	56574	64392	65599	66586	67155	67659	67478	67775	67891
False Negatives (FN)	7493	1776	526	75	63	56	39	32	44	19	19
Total Positives	30541	10855	6503	4356	3149	2366	1773	1469	1189	863	718
Total Negatives	72428	92114	96466	98613	99820	100603	101196	101500	101780	102106	102251
TP/FP ratio	1.60	0.21	0.11	0.07	0.05	0.04	0.03	0.02	0.02	0.01	0.01
Sensitivity (%)	75.47	83.64	91.91	98.28	98.00	97.63	97.80	97.82	96.30	97.80	97.35
Specificity (%)	80.14	52.11	41.35	34.70	34.28	33.81	33.64	33.34	33.70	33.62	33.60
Precision (%)	61.57	17.07	9.56	6.23	4.49	3.35	2.52	2.08	1.67	1.23	1.02
Accuracy (%)	78.75	55.43	44.55	37.39	36.23	35.28	34.74	34.26	34.42	34.16	34.05
Train + test samples	514843										

Table III: Model results using the augmented test set

k	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
True Positives (TP)	45118	49450	39092	29724	21185	14780	12422	10108	8885	7214	6159
True Negatives (TN)	41411	47095	60822	55353	54415	57056	49748	44677	35963	34082	34256
False Positives (FP)	30965	45179	35498	42949	45415	43781	51401	56946	65778	68025	67740
False Negatives (FN)	5697	7726	10575	5642	4024	3781	1981	1137	294	150	163
Total Positives	50815	57176	49667	35366	25209	18561	14403	11245	9179	7364	6322
Total Negatives	72376	92274	96320	98302	99830	100837	101149	101623	101741	102107	101966
TP/FP ratio	1.4571	1.0945	1.1012	0.6921	0.4665	0.3376	0.2417	0.1775	0.14	0.11	0.09
Sensitivity (%)	88.79	86.49	78.71	84.05	84.04	79.63	86.25	89.89	96.80	97.96	97.42
Specificity (%)	57.22	51.04	63.15	56.31	54.51	56.58	49.18	43.96	35.35	33.38	33.59
Precision (%)	59.30	52.26	52.41	40.90	31.81	25.24	19.46	15.07	11.90	9.59	8.33
Accuracy (%)	70.24	64.60	68.44	63.65	60.46	60.17	53.80	48.54	40.43	37.72	37.31
Train + test Samples	615951	747250	729932	668339	625191	596988	577759	564340	554596	547351	541590



(a) ROC curve of the classifier for a noise threshold k = 0 and using the augmented and non augmented data. The samples are obtained by varying the detection threshold between 0 and 1



(b) ROC curve of the classifier for a noise threshold k = 1 and using the augmented and non augmented data. The samples are obtained by varying the detection threshold between 0 and 1

single point at $(0, 1)$ on the top left of the graph). Conversely, data augmentation greatly improves the model for higher k, as seen in fig. 2b. The sensitivity of the model is less affected in this case. This is expected since the data set have very few positive examples for larger k.

A general remark about the galaxy detection problem is that is is very hard to have a non-skewed data set, without using data augmentation. The images have to be split into a small form factor, otherwise the ratio of galaxies/non galaxies per image is dominated by negative examples, making data augmentation counter productive. The drawback of using small images is that one loses the global vision of the picture and the detection becomes harder.

V. Conclusion

In this paper, we tackled the problem of galaxy detection. The attempt at implementing a YOLO-like algorithm was not successful. We therefore decided to pursue with binary classification of galaxies. The algorithm was simplified to take smaller images as inputs. The performance of the model was evaluated for different noise thresholds, as well as with or without data augmentation. It was found that a lower noise threshold allowed for better accuracy and better precision. The model trained on the augmented data set was also observed to be more robust for different noise thresholds.

REFERENCES

[1] Ska science data challenge 1. [Online]. Available: https://astronomers.skatelescope.org/ska-science-data-challenge-1/

[2] D. C. Wells and E. W. Greisen, "Fits - a flexible image transport system," *International Workshop on Image Processing in Astronomy. Proceedings of the 5th. Colloquium on Astrophysics, held in Trieste, Italy, June 4-8, 1979. Editors, G. Sedmak, M. Capaccioli, R.J. Allen.; Publisher, Osservatorio Astronomico di Trieste, Trieste, Italy, 1979. ISBN NONE. LC QB51.3.E43 C64 1979*, p. 445, 1979.

[3] A. Gupta, F. Nelwamondo, S. Mohamed, C. Ennett, and M. Frize, "Statistical normalization and back propagation for classification," *International Journal of Computer Theory and Engineering*, vol. 3, no. 1, pp. 1793–8201, 2011.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi;, "You only look once: Unified, real-time object detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.

[5] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167