Galaxy Classification and Detection

Benedetta Tessa, Franco Terranova

Computational Intelligence and Deep Learning

Master's Degree in Artificial Intelligence and Data Engineering

University of Pisa

Contents

Chapter 1

Galaxies and Beyond

1.1 Introduction

Deep Learning has proven to be of crucial importance in various fields, including astronomy.
As a matter of fact, many deep learning models have been used for galaxy classification and detection.
Various astronomer study the organization of galaxies, their shape and how matter is distributed in order to better understand how our universe is built, how it evolves bust most importantly its origin. More in particular, neural networks have been widely used in modern telescopes to identify clusters of galaxies, one being the *Hubble Space Telescope*.
For this project, we wanted to implement a classification task able to classify galaxies among the ten well-known classes of galaxies.
A detection task has been also highlighted, with this pipeline in mind:
(i) Given an image, recognize the galaxy from other astronomical entities
(ii) Classify the galaxy into one of 10 main classes.
We will explain in detail these two steps and the reason behind certain choices.
The most used neural networks for this task are by far the CNNs, and we tried to find the best hyper-parameters in terms of accuracy and network size.
Transfer learning is also widely used by researchers, leveraging both fine tuning and feature extraction. For many studies, *VGG-16* it turned out to be one of the most performing architectures whereas for others one of the worst, so we thought it would be interesting to use it to see how well it performed with our dataset.

Chapter 2

Galaxy Classification

2.1 Classification problem

Our goal is to perform a multi-class classification, considering the ten different well-known classes of galaxies.
To this end, we trained a CNN from scratch, comparing various configurations.
The most performing CNN was then compared with a fine-tuned pre-trained CNN, and the best model has been chosen.

2.2 Dataset Description

The dataset used for the analysis is the Galaxy10 dataset `https://data.galaxyzoo.org/`, created with GalaxyZoo Data Release 2, a project carried out from July 2007 to February 2009 where a group of volunteers classified a set of galaxies images. This dataset is composed of 17736 69x69 pixels coloured galaxy images divided into 10 classes:

- Class 0 (1081 images): Disturbed Galaxies
- Class 1 (1853 images): Merging Galaxies
- Class 2 (2645 images): Round Smooth Galaxies
- Class 3 (2027 images): In-between Round Smooth Galaxies
- Class 4 (334 images): Cigar Shaped Smooth Galaxies
- Class 5 (2043 images): Barred Spiral Galaxies

- Class 6 (1829 images): Unbarred Tight Spiral Galaxies
- Class 7 (2628 images): Unbarred Loose Spiral Galaxies
- Class 8 (1423 images): Edge-on Galaxies without Bulge
- Class 9 (1873 images): Edge-on Galaxies with Bulge

2.3 Class Imbalance

The first problem encountered was the presence of a highly imbalanced dataset. Different approaches were taken in order to train a neural network able to learn with the presence of an imbalanced dataset.

- Re-balancing
- Class weights
- Partial re-balancing and class weights

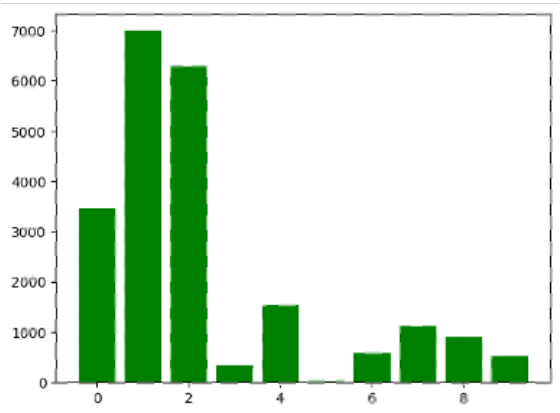


Figure 2.1: Dataset classes distribution

2.3.1 Re-balancing

A first approach involved re-balancing the training set using down-sampling and over-sampling.
The class 5 containing only 17 samples was completely removed from the dataset, moving towards a multi-class classification problem with 9 classes.
This choice has been taken also by other studies applied to this dataset.
Subsequently, the classes 1 and 2 were down-sampled and classes 3,4,6,7,8,9 were over-sampled in order to reach a quite balanced dataset.

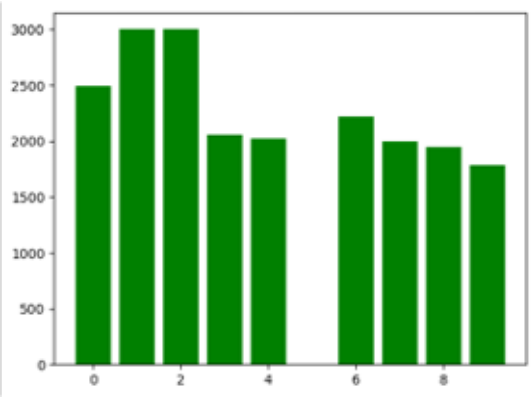


Figure 2.2: Dataset classes distribution

Oversampling has been performed using the following set of parameters, manually found trying different possible data augmentation parameters' values.

- rotation_range = 40
- width_shift_range = 0.2
- width_shift_range = 0.1
- shear_range = 0.2
- zoom_range = 0.2
- horizontal_flip = True
- fill_mode = nearest

2.3.2 Class weights

A second approach involved the usage of the original dataset and class weights in order to train our model while taking in consideration an imbalanced dataset.
We essentially want to work on the loss function by assigning a higher weight to the loss encountered by the samples associated with minor classes.
The weights for a class c are computed in the following manner:

$$weight[c] = \frac{N}{C * IMAGES[c]}$$

With N being the total number of images in the training set, C being the total number of classes and $IMAGES[c]$ the number of images belonging to the class c in the training set. Here's the value of the weight for each class:

- 0: 0.6999
- 1: 0.3471
- 2: 0.3826
- 3: 6.7764
- 4: 1.5521
- 5: 4.1965
- 6: 2.1851
- 7: 2.6959
- 8: 4.5710

2.3.3 Partial Re-balancing and Class weights

A third approach involved re-balancing partially our training set using down-sampling and over-sampling, while using class weights to deal with the presence of a not fully balanced dataset.
Using this approach, we reduced the amount of artificial data present in our dataset while obtaining quite comparable class weights and a quite balanced dataset.
Down-sampling and over-sampling were performed to the same classes and the same parameters listed in the previous section, but in a lower amount.

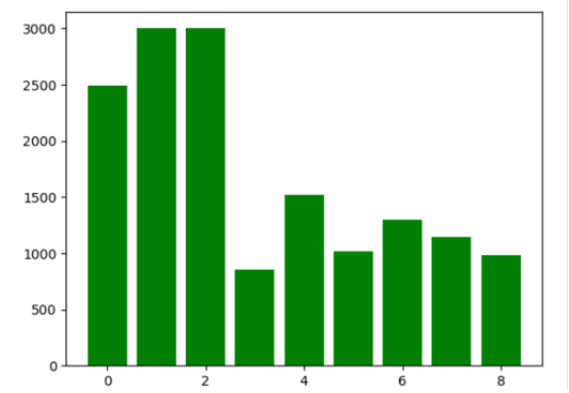


Figure 2.3: Dataset classes distribution

2.3.4 Training and evaluation

The three approaches were compared using a first scratch simple model and the hold-out method.

Among the three approaches, the fully balanced approach reached the best performances' metrics on the test set.

The partially balanced approach reached though very similar metrics with a lower amount of artificial data.

For this reason, the latter has been considered to be the best approach, and was the one we considered in the next chapters.

In this chapter, we can see the classification report and the ROC curve obtained in the test set, and the model used to compare the approaches.

Classification Report

	precision	recall	f1-score	support
0	0.46	0.55	0.50	471
1	0.83	0.85	0.84	946
2	0.79	0.74	0.77	835
3	0.57	0.41	0.48	49
4	0.77	0.81	0.79	207
5	0.83	0.76	0.79	79
6	0.42	0.50	0.46	143
7	0.46	0.27	0.34	129
8	0.35	0.25	0.29	72
accuracy			0.70	2931
macro avg	0.61	0.57	0.58	2931
weighted avg	0.70	0.70	0.70	2931

Figure 2.4: Classification report

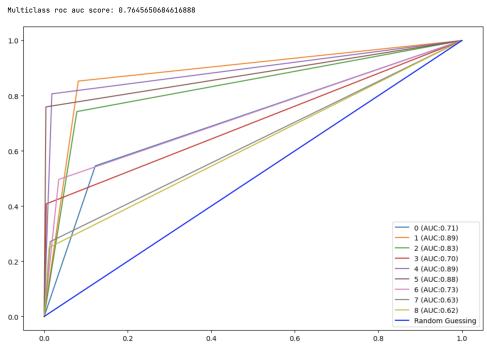


Figure 2.5: ROC Curve

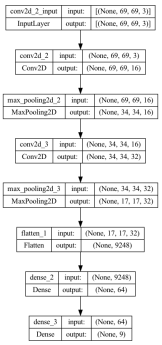


Figure 2.6: Model

2.4 Training callbacks

While training all the neural networks in the following chapters, two different callbacks were used.
The EarlyStopping callback allowed us to interrupt training when accuracy stopped improving on the validation set for at least a certain number of epochs, challenging overfitting.
The ReduceLROnPlateau callback was also used to monitor the model's validation loss and reduce the LR when the validation loss stopped improving, effective strategy to escape local minima.

2.5 Scratch model

Different approaches were subsequently considered in order to enhance the capabilities of our scratch model.

- The network size has been increased, adding other two pairs of convolutional and pooling layers.
- Dropout was added in order to reduce the overfitting.
- Batch normalization layers and an L1_L2 regularizer have been added in order to reduce overfitting even more and speed up network training.

The holdout method was used in order to compare the different approaches.
The following table summarizes the metrics obtained on the test set considering the possible combination of approaches.

Model	Accuracy	Precision	Recall	AUC
Simple CNN	0.70	0.61	0.57	0.76
Bigger CNN	0.76	0.66	0.63	0.80
Bigger CNN with dropout	0.81	0.76	0.73	0.85
Bigger CNN with dropout, regularizer and batch normalization	0.79	0.76	0.72	0.85
Bigger CNN with dropout and batch normalization	0.83	0.79	0.74	0.86
Bigger CNN with dropout and regularizer	0.70	0.59	0.57	0.76
Bigger CNN with regularizer and batch normalization	0.78	0.73	0.71	0.84

During the evaluation of the different combinations, it has been highlighted that dropout and the regularizer show a decrease of performance when paired together. The best scratch model was the one obtained with a bigger CNN, dropout, and batch normalization.

Classification Report

	precision	recall	f1-score	support
0	0.64	0.70	0.67	471
1	0.91	0.94	0.92	946
2	0.89	0.87	0.88	835
3	0.82	0.65	0.73	49
4	0.89	0.89	0.89	207
5	0.84	0.87	0.86	79
6	0.73	0.67	0.70	143
7	0.68	0.50	0.58	129
8	0.72	0.57	0.64	72
accuracy			0.83	2931
macro avg	0.79	0.74	0.76	2931
weighted avg	0.83	0.83	0.83	2931

Figure 2.7: Classification report

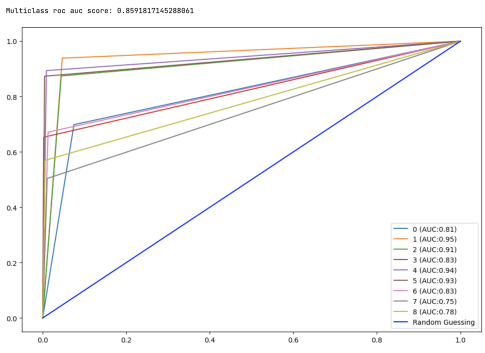


Figure 2.8: ROC Curve

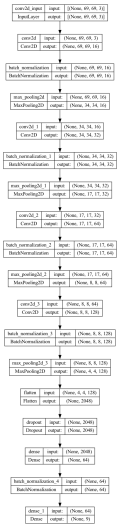


Figure 2.9: Model

2.6 Transfer Learning

Transfer learning is a research problem in machine learning that focuses on using knowledge gained while solving a certain problem and applying it to a different problem.
We've used the VGG-16 pre-trained convolutional neural network to apply transfer learning to our problem.
The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals.

2.6.1 Feature extraction

A first approach involved using feature extraction in order to create a classification model starting from the convolutional base of VGG-16.
The convolutional base was frozen, while two dense layers were added and their weights have been fine-tuned in order to classify galaxies.
The number of resulting trainable parameters with this approach were 131,721 with respect to 14,846,409 total parameters.
This approach, as expected, reached low results, mainly because of the generality of the VGG-16 model.

2.6.2 Fine-tuning

We decided then to fine-tune the VGG-16 model starting from the fifth convolutional block.
The number of resulting trainable parameters with this approach were 7,221,145 with respect to 14,846,409 total parameters.
The model reached very good performances, comparable to the best scratch model found in the previous section.

2.7 K-Fold CV

A K-Fold CV was used in order to compare the best scratch model and the fine-tuned VGG model in order to have a more reliable comparison.
Using 5 folds and comparing the average and standard deviation of accuracy, precision, and recall, the following results were obtained.

Classification Report

	precision	recall	f1-score	support
0	0.64	0.68	0.66	471
1	0.90	0.90	0.90	946
2	0.86	0.83	0.84	835
3	0.80	0.71	0.75	49
4	0.89	0.92	0.91	207
5	0.91	0.86	0.88	79
6	0.69	0.73	0.71	143
7	0.69	0.61	0.65	129
8	0.65	0.72	0.68	72
accuracy			0.81	2931
macro avg	0.78	0.77	0.78	2931
weighted avg	0.82	0.81	0.82	2931

Figure 2.10: Classification report

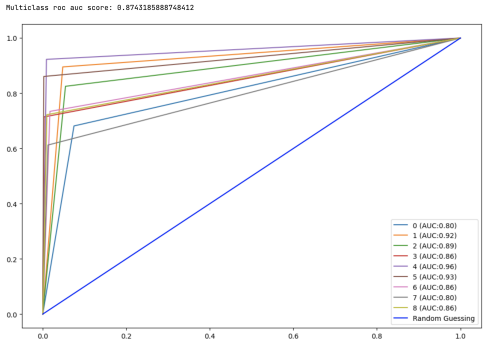


Figure 2.11: ROC Curve

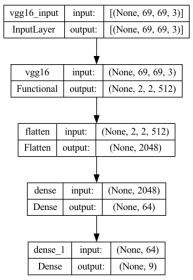


Figure 2.12: Model

K-Fold CV						
Model	Accuracy Avg	Accuracy Std	Precision Avg	Precision Std	Recall Avg	Recall Std
Scratch model	0.80	0.01	0.82	0.02	0.70	0.02
Fine-tuned VGG-16	0.77	0.02	0.79	0.01	0.74	0.02

A Wilcoxon signed rank sum test was considered in order to assess the significance of the comparison, providing p-values of the null hyphotesis all below a confidence level set to 0.05.

2.8 Interpretability

Different visualization methods have been used in order to increase the interpretability of our model.

2.8.1 Class Activation Heatmaps

The class activation heatmaps are able to tell us which part of an image has led the CNN to take its final decision.
These heatmaps have been obtained weighting every channel of the output feature map by the gradient of the maximally activated class with respect to the channel, and multiplying each channel depending on how important it is with respect to the target class.
The channel-wise mean of the resulting feature map has then been calculated in the

heatmap of the class activation, and the heatmap has been normalized between 0 and 1 for visualization purposes.

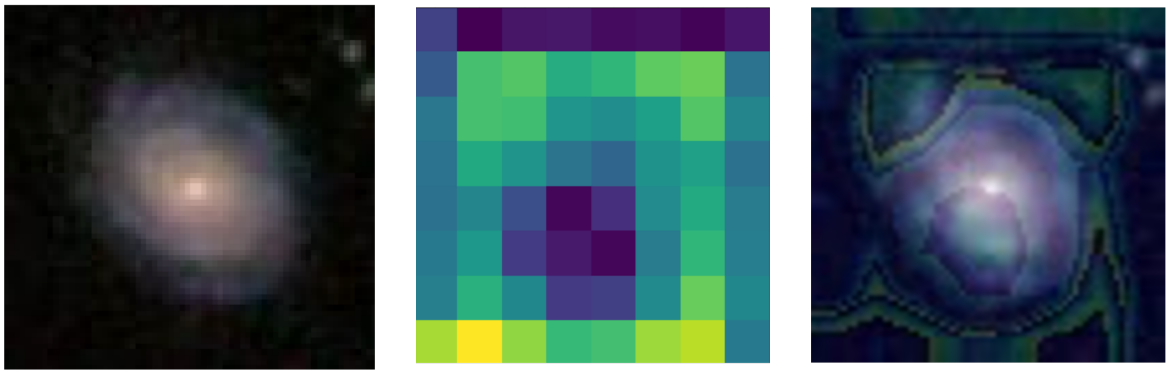
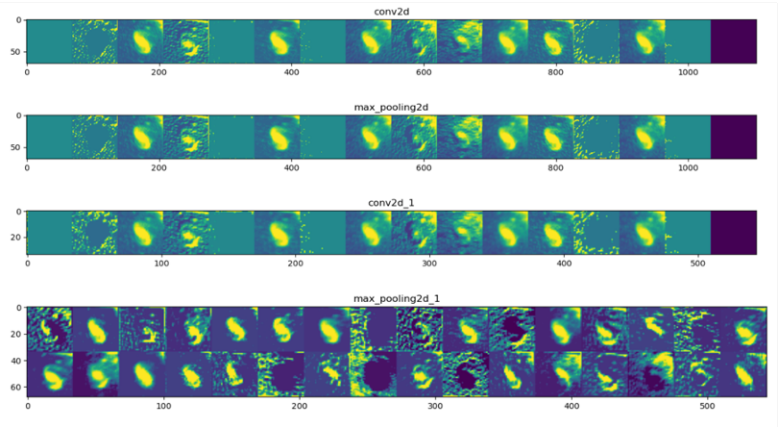


Figure 2.13: Class activation heatmas

2.8.2 Intermediate activations

The feature maps outputs of convolutional and pooling layers of the network given a certain input can be shown, giving a view of how the input is decomposed into the filters learned by the network.



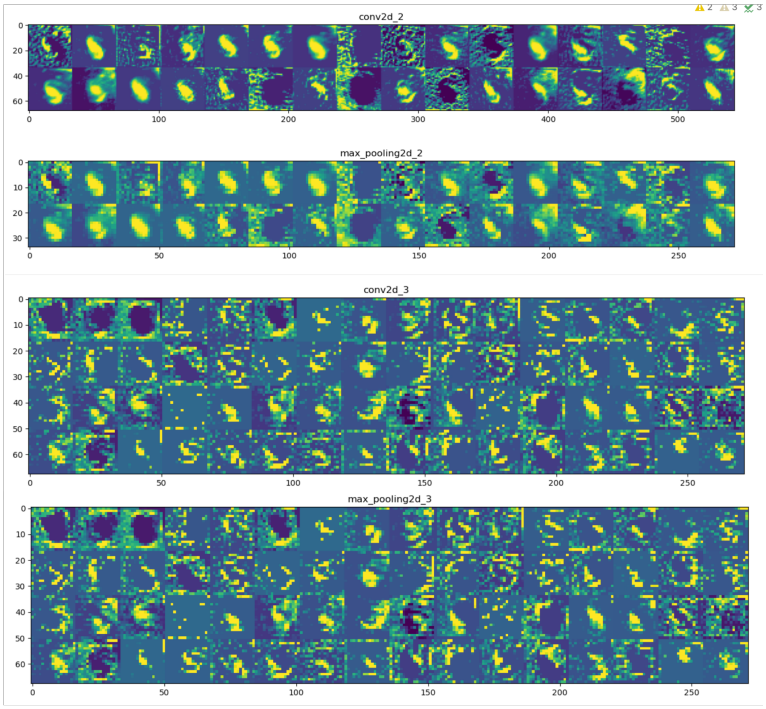


Figure 2.14: Intermediate activations

2.8.3 Convnet filters

Convnet filters display the visual pattern that each filter is meant to respond to, that can be obtained applying gradient ascent in the input space, maximizing the response of a specific filter, and calculating the input images the filter is maximally responsive to. We highlight here the patterns of response of the first 8 channels of the conv2d_3 layer. The same information can be taken from each layer.

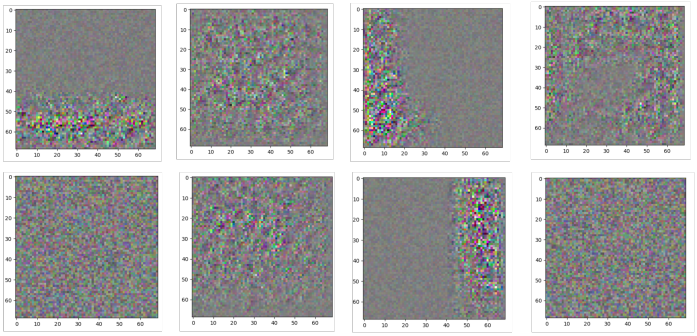


Figure 2.15: Convnet filters

2.9 Further Considerations

2.9.1 Ensemble method

As we can see in the table above, the CNN from scratch has a better precision whilst the fine-tuned VGG-16 has a better recall.
What we tried to do is combine the contribution of the two networks to see if the newly created ensemble method gave better results.
Another reason we decided to implement such ensemble model is because it may be more stable and better in classifying unseen data.
The contribution is weighted and the value of the weights has been found heuristically. The best are 0.6 for the CNN from scratch and 0.4 for the fine-tuned VGG-16.
As we can see from the results shown below, the metrics are slightly more satisfactory. Of course, we have the drawback that the model will be more expensive in terms of computation and memory.

	precision	recall	f1-score	support
0	0.67	0.72	0.70	471
1	0.92	0.94	0.93	946
2	0.89	0.88	0.88	835
3	0.88	0.73	0.80	49
4	0.91	0.91	0.91	287
5	0.98	0.98	0.98	79
6	0.77	0.71	0.74	143
7	0.75	0.61	0.67	129
8	0.77	0.69	0.73	72
accuracy			0.85	2931
macro avg	0.83	0.79	0.81	2931
weighted avg	0.85	0.85	0.85	2931

Figure 2.16: Classification report

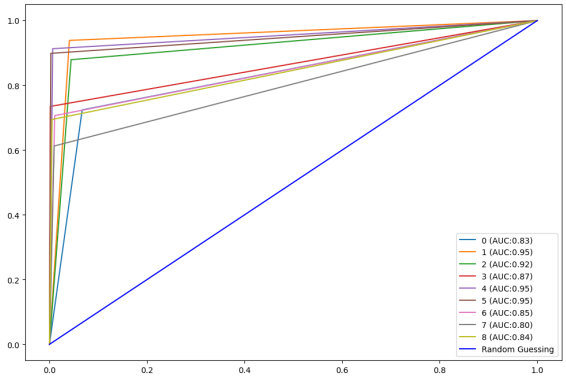


Figure 2.17: ROC Curve

2.9.2 Hyper-parameters optimization

As mentioned in the introduction, CNNs can be embedded in the hardware of most telescopes. Nowadays, telescopes within everyone's reach can be able to see galaxies and proper hardware can be embedded to gather the images collected.
If that's the case, the networks shouldn't be too big or computationally expensive, if anything they should be small and fast.
That's why we tried to perform hyper-parameters optimization on a small network by means of the Bayesan Optimization algorithm.

What we decided to tune are the following parameters, limitating the size of the network to two layers.

- Filter size of a first convolutional layer
- Kernel size of a first convolutional layer
- Filter size of a second convolutional layer
- Kernel size of a second convolutional layer
- Number of units of first dense layer
- Dropout rate

The network obtained got comparable performances to the previous networks but is much smaller in size.

Classification Report

	precision	recall	f1-score	support
0	0.63	0.63	0.63	471
1	0.91	0.93	0.92	946
2	0.91	0.87	0.89	835
3	0.83	0.78	0.80	49
4	0.86	0.92	0.89	207
5	0.88	0.80	0.83	79
6	0.64	0.70	0.67	143
7	0.63	0.57	0.60	129
8	0.54	0.67	0.60	72
accuracy			0.82	2931
macro avg	0.76	0.76	0.76	2931
weighted avg	0.83	0.82	0.82	2931

Figure 2.18: Hyper-optimized model Classification report

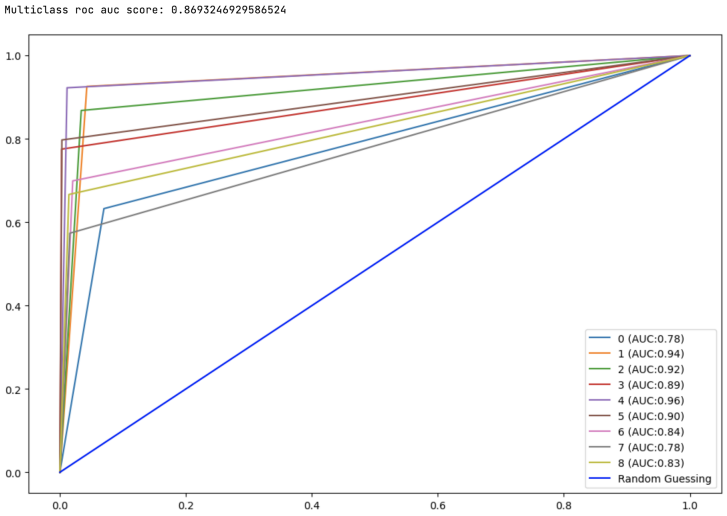


Figure 2.19: Hyper-optimized model ROC Curve

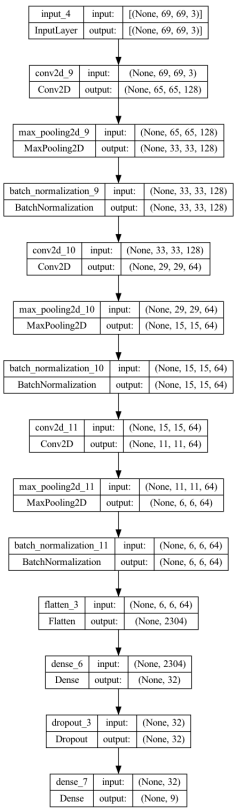


Figure 2.20: Hyper-optimized model

Chapter 3

Object Detection

The TensorFlow Object Detection API has been used in order to develop an object detection model able to detect galaxies.
Due to the presence of a few training samples, transfer learning has been used, fine-tuning different pre-trained algorithms on COCO (Common Objects in a COmmon context) dataset, that can be found here: `https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md`.
The idea of the object detection model is to train a model able to detect galaxies among celestial objects.
This model could be paired with the previous model, creating an end-to-end model composed of a first detection of galaxies and then the subsequent classification.
An object detection model can be used to both localize and classify objects, but due to the absence of a dataset suited for both tasks and also to satisfy the objective of the exam, the localization and classification models has been considered as two separate models, that could be paired to perform an end-to-end task.

3.1 GalayZoo 2 Dataset

We used the GalaxyZoo 2 dataset in order to create our Object Detection model: `https://www.kaggle.com/datasets/jaimetrickz/galaxy-zoo-2images?resource=download&select=gz2_filename_mapping.csv`.
A different dataset was used because this last dataset taken in consideration, provided images of galaxies together with other objects, differently from the previous one, in order to train the network to distinguish the galaxies from other possible celestial elements.

Moreover, this dataset could not be used for classification because of the absence of information regarding the class.
We manually annotated 2000 images, among the ones contained in the dataset, using the labelImg tool, that provides for each image an XML annotation file containing information about the bounding box and the associated class.

3.2 Transfer Learning

Two different pre-trained models were compared in this project.
The considered model are particularly suited for constrained devices, in order to run properly on instrumenting devices that could be embedded in normal telescopes, in order to achieve fast real-time detection.
The first considered model is SSD Mobilenet, an efficient architecture introduced by Google (using depthwise and pointwise convolutions). It can be used for classification purposes, or as a feature extractor for other tasks (i.e. detection).
The second model considered is EfficientDet, a type of object detection model, which utilizes several optimization and backbone tweaks, such as the use of a BiFPN, and a compound scaling method that uniformly scales the resolution,depth and width for all backbones, feature networks and box/class prediction networks at the same time.
The SSD Mobilenet reached higher performances in the considered metrics calculated on the test set.

K-Fold CV		
Model	MAP@0.50IOU	AR@1
SSD Mobilenet	0.96	0.62
EfficientDet	0.94	0.60

Figure 3.1: Galaxy detection on test set images

Bibliography

[1] https://www.epfl.ch/labs/mlo/wp-content/uploads/2021/05/crpmlcourse-paper845.pdf

[2] https://academic.oup.com/mnras/article/493/3/3178/5722127?login=false

[3] https://ieeexplore.ieee.org/abstract/document/9672430?casa_token=Tyb9pO9vRN8AAAAA:3G1kRH8OQxFB-q3sfj96xNQXfN-Sdmd8XlFyScY4bh7pKJbqRT_MGSMUeUknDNfhD9sc_mOrbA

[4] https://www.sciencedirect.com/science/article/pii/S2215098621001701https://ieeexplore.ieee.org/document/9622583

[5] https://www.researchgate.net/publication/327673682_Galaxy_detection_and_identification_using_deep_learning_and_data_augmentation

[6] https://arxiv.org/pdf/2212.07881.pdf