

# Towards a Universal Machine Learning Pipeline to Understand Galaxies

**Nuno Ramos Carvalho**

Mestrado em Astronomia e Astrofísica

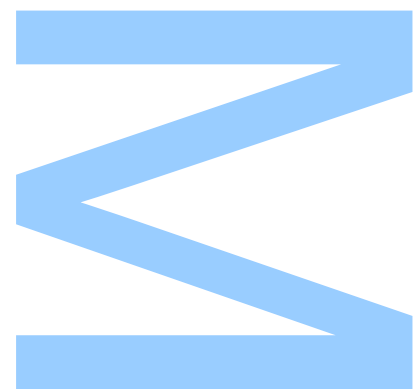
[Departamento de Física e Astronomia](#)

2022

**Orientador**

[Dr. Andrew J. Humphrey](#), Centro de Astrofísica da Universidade do Porto /

IA-Porto



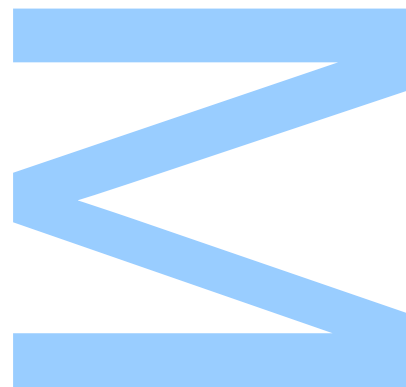




Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, \_\_\_\_ / \_\_\_\_ / \_\_\_\_





Universidade do Porto

Masters Thesis

---

# **Towards a Universal Machine Learning Pipeline to Understand Galaxies**

---

*Author:*

Nuno Ramos Carvalho

*Supervisor:*

Dr. Andrew J. Humphrey

*A thesis submitted in fulfilment of the requirements  
for the degree of MSc. in Astronomy and Astrophysics*

*at the*

Faculdade de Ciências da Universidade do Porto  
Departamento de Física e Astronomia

December 20, 2022



# Declaração de Honra

Eu, Nuno Alexandre Ramos de Carvalho, inscrito(a) no Mestrado em Astronomia e Astrofísica da Faculdade de Ciências da Universidade do Porto declaro, nos termos do disposto na alínea a) do artigo 14.º do Código Ético de Conduta Académica da U.Porto, que o conteúdo da presente dissertação reflete as perspetivas, o trabalho de investigação e as minhas interpretações no momento da sua entrega.

Ao entregar esta dissertação, declaro, ainda, que a mesma é resultado do meu próprio trabalho de investigação e contém contributos que não foram utilizados previamente noutros trabalhos apresentados a esta ou outra instituição.

Mais declaro que todas as referências a outros autores respeitam escrupulosamente as regras da atribuição, encontrando-se devidamente citadas no corpo do texto e identificadas na secção de referências bibliográficas. Não são divulgados na presente dissertação/relatório de estágio/projeto [indicar o aplicável] quaisquer conteúdos cuja reprodução esteja vedada por direitos de autor.

Tenho consciência de que a prática de plágio e auto-plágio constitui um ilícito académico.

Nuno Alexandre Ramos de Carvalho

Porto, 20 de dezembro de 2022





*“There is an infinite number of worlds, some like this world, others unlike it.”*

Epicurus, 300 BCE



# *Acknowledgements*

Dedicated to my ISU family, the class of SSP2022 in particular,  
and to everyone else that dares to look up  
and explore beyond.



UNIVERSIDADE DO PORTO

## *Abstract*

Faculdade de Ciências da Universidade do Porto

Departamento de Física e Astronomia

MSc. in Astronomy and Astrophysics

### **Towards a Universal Machine Learning Pipeline to Understand Galaxies**

by [Nuno Ramos Carvalho](#)

Galaxies are the building blocks of the Universe at large scale. The study of their formation and evolution is fundamental to describe the history of the Universe, and for the understanding of physical processes such as star-formation, the build-up of nuclear supermassive black holes, and quenching processes. While providing valuable insights and knowledge for other disciplines like cosmology for example.

Machine Learning methods are becoming increasingly relevant for many studies in astrophysics and related disciplines. Deep Learning approaches are providing state-of-the-art results for many classification and estimation tasks, enabling the creation of methods for estimating galaxies proprieties in a systematic approach while providing efficient ways to process large volumes of data. With the advent of upcoming missions and surveys new approaches will be required to quickly and consistently process large datasets.

The overarching goal of this work is to explore the composition of Deep Learning models to build pipelines for inferring properties about galaxies exploring data from heterogeneous sources. A total of 31 models, composed by 518 layers, comprised of a total of 70 128 689 trainable parameters, provide the building blocks for creating easy to apply, update and evolve pipelines for systematic galaxy characterization. A new dataset, built from SDSS data, of around *100k* objects is used to train, validate and test the devised models, that achieved a best score on the test sets of 0.00047 on the MSE loss function for predicting the redshift, 7.17838 on the MAE loss function for predicting the stellar mass, an accuracy of 88.456% for predicting the sub-class and of 51.504% for predicting the GZ2 simplified class. All the models, tools and resources are publicly available under open-source licenses, immediately enabling anyone to use them in their own work or research without restrictions or limitations.



UNIVERSIDADE DO PORTO

## *Resumo*

Faculdade de Ciências da Universidade do Porto

Departamento de Física e Astronomia

Mestrado em Astronomia e Astrofísica

### **Rumo a um *Pipeline* Universal de Aprendizagem Máquina para Compreender Galáxias**

por [Nuno Ramos Carvalho](#)

As galáxias são os blocos de construção do Universo em grande escala. O estudo da sua formação e evolução é fundamental para descrever a história do Universo e para a compreensão de processos físicos como a formação de estrelas, a geração de buracos negros super-massivos e processos que levam à redução da taxa de formação estelar. Fornecendo informação e conhecimentos valiosos para outras disciplinas como por exemplo a cosmologia.

Métodos de Aprendizagem Máquina estão a tornar-se cada vez mais relevantes para muitos estudos em astrofísica e disciplinas relacionadas. Abordagens baseadas em *Deep Learning* estão a atingir resultados estado-da-arte em muitas tarefas de classificação e previsão, possibilitando a criação de métodos para estimar propriedades de galáxias de uma forma sistemática e fornecendo métodos eficientes para processar grandes volumes de dados. Com o advento das próximas missões e censos, novas abordagens serão necessárias para lidar de forma rápida e consistente com grandes volumes de novos dados.

O objetivo deste trabalho é explorar a composição de modelos de *Deep Learning* para construir *pipelines* para inferir propriedades de galáxias. Um total de 31 modelos, compostos por 518 camadas, com um total de 70 128 689 parâmetros treináveis, fornecem os blocos necessários para construir *pipelines* expeditos de aplicar e atualizar que exploram dados de fontes heterogêneas para caracterização sistemática de galáxias. Um conjunto de dados construído a partir de dados do SDSS com cerca de 100k objetos é utilizado para treinar e validar os modelos implementados. Os melhores modelos obtiveram os resultados nos dados de teste de: 0.00047 na função de perda MSE para estimar

o *redshift*, 7.17838 na função de perda MAE para estimar a massa estelar, uma precisão de 88.456% para estimar a sub-classe e de 51.504% para estimar a classe simplificada do GZ2. Todos os modelos, ferramentas e recursos desenvolvidos são distribuídos publicamente sob licenças de software aberto, imediatamente disponíveis para qualquer um usar no seu trabalho ou investigação sem restrições ou limitações.



# Contents

|  |             |
|--|-------------|
| <b>Acknowledgements</b>                                    | <b>v</b>    |
| <b>Abstract</b>  | <b>vii</b>  |
| <b>Resumo</b>  | <b>ix</b>   |
| <b>Contents</b>  | <b>xi</b>   |
| <b>List of Figures</b>                                     | <b>xiii</b> |
| <b>List of Tables</b>                                      | <b>xv</b>   |
| <b>List of Acronyms</b>                                    | <b>xix</b>  |
| <b>1 Introduction</b>                                      | <b>1</b>    |
| 1.1 The Problem . . . . .                                  | 2           |
| 1.2 Motivations . . . . .                                  | 3           |
| 1.3 Work Plan . . . . .                                    | 4           |
| 1.4 Main Contributions . . . . .                           | 5           |
| 1.5 Document Outline . . . . .                             | 5           |
| <b>2 Understanding Galaxies</b>                            | <b>7</b>    |
| 2.1 Imaging & Photometry . . . . .                         | 7           |
| 2.2 Morphological Classification . . . . .                 | 8           |
| 2.3 Spectral Analysis . . . . .                            | 10          |
| 2.3.1 Cosmological Redshift . . . . .                      | 10          |
| 2.3.2 Stellar Mass . . . . .                               | 12          |
| 2.4 Starbursts & Active Galaxies . . . . .                 | 12          |
| 2.5 Observational Surveys, Catalogues & Projects . . . . . | 14          |
| 2.5.1 Sloan Digital Sky Survey (SDSS) . . . . .            | 14          |
| 2.5.2 Wide-Field Infrared Survey Explorer (WISE) . . . . . | 15          |
| 2.5.3 Galaxy Zoo 2 (GZ2) . . . . .                         | 15          |
| <b>3 Deep Learning Essentials</b>                          | <b>17</b>   |
| 3.1 Fundamentals of Machine Learning . . . . .             | 18          |
| 3.1.1 Defining the Problem & Major Branches . . . . .      | 19          |
| 3.2 Artificial Neural Networks & Deep Learning . . . . .   | 20          |

|          |  |            |
|----------|--|------------|
| 3.2.1    | Network Architecture & Composition of Layers . . . . .   | 21         |
| 3.2.2    | Loss Functions . . . . .                                 | 24         |
| 3.2.3    | Optimizers . . . . .                                     | 25         |
| 3.2.4    | Training, Validation & Testing . . . . .                 | 25         |
| 3.3      | Advanced Techniques . . . . .                            | 26         |
| 3.3.1    | Hyper-parameters Tuning . . . . .                        | 27         |
| 3.3.2    | Regularization . . . . .                                 | 27         |
| 3.3.3    | Data Generators . . . . .                                | 27         |
| <b>4</b> | <b>Datasets &amp; Models</b>                             | <b>29</b>  |
| 4.1      | The SDSS Galaxy Subset . . . . .                         | 29         |
| 4.2      | Exploratory Analysis . . . . .                           | 31         |
| 4.3      | Features, Inputs & Outputs . . . . .                     | 34         |
| 4.4      | Models Architectures . . . . .                           | 37         |
| 4.4.1    | Predicting Redshift . . . . .                            | 38         |
| 4.4.2    | Predicting Stellar Mass . . . . .                        | 42         |
| 4.4.3    | Predicting Sub-class . . . . .                           | 43         |
| 4.4.4    | Predicting Galaxy Zoo 2 (GZ2) Simplified Class . . . . . | 48         |
| 4.4.5    | Multi Input/Output Models . . . . .                      | 49         |
| 4.5      | Training & Evaluation . . . . .                          | 60         |
| <b>5</b> | <b>Pipelines for Characterization</b>                    | <b>69</b>  |
| 5.1      | Models Composition . . . . .                             | 69         |
| 5.2      | Results & Discussion . . . . .                           | 72         |
| 5.2.1    | Threats to Validity . . . . .                            | 75         |
| 5.3      | Example Application . . . . .                            | 76         |
| <b>6</b> | <b>Conclusion</b>  | <b>83</b>  |
| 6.1      | Final Remarks . . . . .                                  | 83         |
| 6.2      | Future Work . . . . .                                    | 86         |
| <b>A</b> | <b>The <i>astromlp</i> Package</b>                       | <b>89</b>  |
| <b>B</b> | <b>The <i>astromlp-models</i> Repository</b>             | <b>91</b>  |
| <b>C</b> | <b>The <i>astromlp</i> API</b>                           | <b>93</b>  |
| <b>D</b> | <b>Galaxy Zoo 2 Simplified Classes</b>                   | <b>95</b>  |
| <b>E</b> | <b>The Keras API</b>                                     | <b>97</b>  |
| <b>F</b> | <b>SDSS SQL Queries</b>                                  | <b>99</b>  |
| <b>G</b> | <b>Introduction to the Haskell Notation</b>              | <b>101</b> |
|          | <b>Bibliography</b>                                      | <b>105</b> |

# List of Figures

|      |   |    |
|------|---|----|
| 2.1  | Hubble classification diagram, where three major branches of galaxies are illustrated: ellipticals (E) to the left, disk galaxies (S) to the right, and lenticular galaxies (S0) in the fork (Hubble, 1936). . . . .  | 9  |
| 2.2  | The shift towards the red from the wavelength of the spectral feature observed in a laboratory on Earth, where $\lambda_{\text{ref}} = 6563.8 \text{ \AA}$ , to the wavelength of the spectral feature from the distant object, where $\lambda_{\text{obj}} = 7222.0 \text{ \AA}$ . . . . . | 11 |
| 2.3  | Example spectrum for a galaxy classified as starburst from the SDSS database. . . . .   | 13 |
| 2.4  | Example spectrum for a galaxy classified as AGN from the SDSS database. . . . .   | 13 |
| 3.1  | Fundamental relations between the fields of Artificial Intelligence (AI), Machine Learning (ML) and Deep Learning (DL). . . . .   | 18 |
| 3.2  | Artificial neural network example with three fully connected hidden layers. . . . .   | 21 |
| 4.1  | Distribution of the sub-class categorical feature, where the STARFORMING sub-class is the most common with a total of 72213 objects. . . . .  | 32 |
| 4.2  | Distribution of values of the redshift variable, where the majority of values are located in the lower range, below 0.2. . . . .  | 33 |
| 4.3  | Distribution of values for the stellar mass variable, in units of $M_{\odot}$ , where the majority of values are lower than $2.5 \times 10^{10} M_{\odot}$ . . . . .  | 34 |
| 4.4  | Distribution of the GZ2 simplified class categorical feature, where the ScR is the most common class with a total of 14646 objects. . . . .   | 34 |
| 4.5  | Predicting redshift from RGB images (left), FITS data (middle) and spectra data (right) model architectures using convolutional operations. . . . .   | 41 |
| 4.6  | Predicting redshift from selected spectra bands data (left) bands data (middle) and WISE data (right) model architectures using only fully connected layers. . . . .  | 42 |
| 4.7  | Predicting stellar mass from RGB images (left), FITS data (middle) and spectra data (right) model architectures using convolutional operations. . . . .   | 44 |
| 4.8  | Predicting stellar mass from selected spectra bands data (left) bands data (middle) and WISE data (right) model architectures using only fully connected layers. . . . .  | 45 |
| 4.9  | Predicting sub-class from RGB images (left), FITS data (middle) and spectra data (right) model architectures using convolutional operations. . . . .  | 47 |
| 4.10 | Predicting sub-class from selected spectra bands data (left) bands data (middle) and WISE data (right) model architectures using only fully connected layers. . . . .   | 48 |
| 4.11 | Computing GZ2 simplified class from RGB images (left), FITS data (middle) and spectra data (right) model architectures using convolutional operations. . . . .  | 50 |

---

|      |  |    |
|------|--|----|
| 4.12 | Computing GZ2 simplified class from selected spectra bands data (left) bands data (middle) and WISE data (right) model architectures using only fully connected layers. . . . .  | 51 |
| 4.13 | Predicting the redshift from the RGB images, FITS, spectral, bands and WISE data using a combination of networks. . . . .  | 53 |
| 4.14 | Predicting the stellar mass from the RGB images, FITS, spectral, bands and WISE data using a combination of networks. . . . .  | 54 |
| 4.15 | Predicting the sub-class from from the RGB images, FITS, spectral, bands and WISE data using a combination of networks. . . . .  | 55 |
| 4.16 | Predicting the GZ2 simplified class from the RGB images, FITS, spectral, bands and WISE data using a combination of networks. . . . .  | 56 |
| 4.17 | Predicting the redshift and the stellar mass from the RGB images, FITS, spectral, bands and WISE data (left); and, the sub-class and the GZ2 simplified class from the RGB images, FITS, spectral, bands and WISE data (right) using combinations of networks. . . . . | 58 |
| 4.18 | Predicting the redshift, the stellar mass, the sub-class and the GZ2 simplified class from the RGB images, FITS, spectral, bands and WISE data using a combination of networks. . . . .  | 59 |
| 5.1  | Description of a pipeline using a map-reduce approach, each model is applied to the relevant inputs, and the intermediate results are combined to yield the final output. . . . .  | 71 |
| 5.2  | <i>astroMLP for Galaxies</i> application online interface main page. . . . .   | 77 |
| 5.3  | <i>astroMLP for Galaxies</i> application information about the <i>One-2-One Ensemble</i> pipeline. . . . .   | 78 |
| 5.4  | <i>astroMLP for Galaxies</i> application output result of applying the <i>One-2-One Ensemble</i> pipeline. . . . .   | 79 |
| 5.5  | <i>astroMLP for Galaxies</i> application information about the model to infer the redshift from spectral data page. . . . .  | 80 |
| 5.6  | <i>astroMLP for Galaxies</i> application output result visualization of applying the model to infer the redshift form the spectral data. . . . .   | 81 |

# List of Tables

|      |  |    |
|------|--|----|
| 2.1  | Sloan Digital Sky Survey (SDSS) photometric data central wavelengths and corresponding full widths at half maximum for bands u, g, r, i and z. . . . .   | 15 |
| 2.2  | Wide-Field Infrared Survey Explorer (WISE) photometric data central wavelengths and corresponding full widths at half maximum for bands W1, W2, W3 and W4. . . . .   | 15 |
| 4.1  | Summary of the columns information available from the tabular data in the SDSS Galaxy Subset (SDSS GS) dataset. . . . .  | 30 |
| 4.2  | Set of labels for the sub-class variable in the SDSS GS tabular data. . . . .  | 32 |
| 4.3  | Descriptive statistics for the redshift and stellar mass (in units of $M_{\odot}$ ) data available from the SDSS GS dataset. . . . .   | 33 |
| 4.4  | Summary of the data in the dataset used as input features for the devised models. . . . .  | 35 |
| 4.5  | Summary of the data in the dataset used as output for the devised models. . . . .  | 35 |
| 4.6  | Summary of the complete list of models with single input and single output. . . . .  | 39 |
| 4.7  | Summary of the complete list of models with multi inputs and outputs. . . . .  | 39 |
| 4.8  | Summary of functions representing the single input single output models for predicting the redshift. . . . .   | 40 |
| 4.9  | Summary of functions representing the single input single output models for predicting the stellar mass. . . . .   | 43 |
| 4.10 | Summary of functions representing the single input single output models for predicting the sub-class. . . . .  | 46 |
| 4.11 | Summary of functions representing the single input single output models for predicting the GZ2 simplified class. . . . .   | 51 |
| 4.12 | Hyper-parameters exploration for the single input/output models predicting the redshift. . . . .   | 61 |
| 4.13 | Hyper-parameters exploration for the single input/output models predicting the stellar mass. . . . .   | 63 |
| 4.14 | Hyper-parameters exploration for the single input/output models predicting the object sub-class. . . . .   | 64 |
| 4.15 | Hyper-parameters exploration for the single input/output models predicting the GZ2 simplified class. . . . .   | 65 |
| 4.16 | Summary of the complete list of models with single input and output hyper-parameters definition and best score achieved during model training. Type of problem is abbreviated: regression (r) or classification (c), and categorical crossentropy loss function is abbreviated as c.c. . . . . | 66 |









# List of Acronyms

**AGN** Active Galactic Nuclei.

**AI** Artificial Intelligence.

**ANN** Artificial Neural Network.

**BOSS** Baryon Oscillation Spectroscopic Survey.

**CNN** Convolution Neural Network.

**CSV** Comma-Separated Values.

**DL** Deep Learning.

**DNN** Deep Neural Networks.

**FITS** Flexible Image Transport System.

**FWHM** Full Width at Half Maximum.

**GZ2** Galaxy Zoo 2.

**IGM** Intergalactic Medium.

**MAE** Mean Absolute Error.

**MJD** Modified Julian Date.

**ML** Machine Learning.

**MSE** Mean Squared Error.

**NN** Neural Network.

**SDSS** Sloan Digital Sky Survey.

**SDSS GS** SDSS Galaxy Subset.

**SED** Spectral Energy Distribution.

**SFR** Star Formation Rate.

**SMBH** Super Massive Black Hole.

**SQL** Structured Query Language.

**WISE** Wide-Field Infrared Survey Explorer.

# Chapter 1

## Introduction

Galaxies are one of the major building blocks while discussing the current structure and knowledge about the Universe. The study of galaxy formation and evolution is fundamental to describe the history of the Universe and in other fields of study, e.g. cosmology. Galaxies are colossal objects and they span across a great variety of distances from Earth, from the Local Group, which includes the Milky Way, to the far edge of the observable Universe, and probably beyond, and therefore cannot be studied in a laboratory, as often is the case in the context of astrophysics. To understand these mammoths indirect methods are used to measure familiar characteristics from observations, e.g. mass, size, composition.

The process of drawing conclusions from observations is, historically, a human mental activity. The challenge is to focus on the details of interest (features) in the available data and capture the pattern that generalizes well enough to the majority of cases, which later allows to infer properties for new observations. For example, given observations of planets, the Moon and the Sun (data), Brahe created detailed records of their position in the Sky (features), and from these Kepler inferred his laws of motion (patterns), that can be used to predict (infer) the mass of objects in and outside the Solar System (generalize) (Koestler, 1959). The challenge is: how to go from features to patterns that describe a set of well defined rules able to accurately infer new conclusions.

Machine Learning (ML) is a broad field concerned with the study of techniques used to discover patterns in data and build models (rules) that capture these patterns and enable inference given new never seen before observations. For example, given detailed observations of the movement of the Earth around the Sun, it is possible to train a model to predict the next position of the Earth relative to the Sun. With more traditional ML techniques the features of focus are selected and devised by humans, with the advent of Deep

Learning (DL) this has changed, because more data is directly used to train the network, and given the nature of these models, they are able to find the features of interest, i.e. more relevant to the task at hand. Of course, in some cases feature engineering, i.e. manually creating features based on domain knowledge, still enables achieving better results. For example, some years ago the spam/no-spam e-mail filters were trained by providing a set of specific features to an ML algorithm, e.g. number of nouns, number of capitalized words, number of words found in the dictionary, today the entire text is fed to a neural network and given enough data, and enough time, the algorithm can learn the features of interest as part of the model training stage.

In any case, the use of ML techniques requires data to train models. To end up with a model that is able to generalize well an adequate volume of data observations is required, otherwise the model will not be able to capture the underlying pattern. Projects like the Sloan Digital Sky Survey (SDSS) provide large volumes of data, readily available, and for free, which enable the creation and exploration of DL and other families of models.

The overarching goal of this work is to explore the use of combinations of DL models and data to devise computational ready to use tools capable of characterizing galaxies. The main intuition behind this idea is to let the network figure out the features of interest in the data, and the pattern that better generalizes to unseen examples, opposed to manually selecting the features of interest. The panoply of research available in the literature already sustains the intrinsic assumption that the required information is available in the data. Hence, the guiding research question for this work is defined as:

*Can a set of data pipelines, using a combination of deep learning models and heterogeneous data sources, potentially provide additional benefits for characterizing galaxies properties to enhance galaxy understanding?*

The remaining sections of this chapter include: a discussion of the generic problem that this approach is addressing; a brief motivation for studies in this area of research; a summary of the work plan, including the main contributions of this work; and an outline of the remaining chapters of this document.

## **1.1 The Problem**

The study and characterization of galaxies is an ongoing challenge in astrophysics. It is not possible to measure a galaxy length with a ruler, or its weight with a traditional scale.

Most of the measurements done in modern times are performed using indirect methods. These provide ways to extrapolate quantities of interest that are not possible to measure directly. The overarching goal of this work is to provide a clear and integrated mechanism to characterize galaxies to contribute to all the works that explore this data as input for analysis.

There are currently different approaches available in the literature for analysing different properties of interest about galaxies, including tools and models at different levels of maturity, Section 2 introduces some of them. Two major shortcomings recurrently emerge: the lack of integration of different approaches, techniques and models; and, the lack of integration of different sources of data. The simultaneous integration of data that models different aspects of nature has the potential to provide more accurate predictions of measures of interest. Another shortcoming with current approaches discussed in the literature for the study of galaxies is the lack of information and/or resources for reproducing the approach applied to new or updated datasets. Galaxy characterization is often based on statistical analysis and tools, and it is common to recurrently process data, including new data releases, to explore new or updated results. Techniques that are not fully described may be hard, or impossible, to systematically (re-)apply to new datasets.

## 1.2 Motivations

Galaxies provide a unique environment for studying physical processes, e.g. general hydrodynamics, thermodynamics, plasma, nuclear or atomic physics, in conditions that are not possible to replicate in a laboratory on Earth. Star formation and evolution is another topic that can also take advantage of the studies on galaxies.

Galaxies play an important role in the study of the structure and evolution of the Universe. They are abundant and bright, and span over cosmological time and length scales on large numbers, which allows using them as tracers of the Universe evolution, their distribution and properties on large scale can provide insight on cosmological parameters that can be used to test cosmological models ([Mo et al., 2010](#)).

The space between galaxies, typically referred to as the Intergalactic Medium (IGM), although empty of stars it still contains some gas and dust of minute particles providing valuable information about the distribution of mass of the Universe at a cosmological

scale. This medium can be studied by analysing the light from distant galaxies or stars that crosses it and reaches the detectors (Mo et al., 2010).

With the increase of available data, and with the upcoming surveys, the feasibility and necessity of using ML increases. Also, the currently available literature already includes many works discussing synergies between these areas of research, emphasizing the motivations for working in these topics, some recent examples include the works by Cavanagh et al. (2021), Collister & Lahav (2004), Dieleman et al. (2015), Huertas-Company et al. (2018) and Tuccillo et al. (2018).

Besides the scientific motivations, by providing well defined pipelines capable of yielding properties of interest, that are able to process data from different sources while integrating models for different aspects and by developing tools and models as stand-alone, self-contained building blocks, integration with future development workflows and applying the pipelines to new datasets is possible systematically, hence addressing some of the shortcomings highlighted in the previous section.

### 1.3 Work Plan

In a nutshell, the devised work plan for addressing the problem guided by the research question is divided in three main parts:

1. Compile a dataset from observations of objects of interest (galaxies) that includes useful information for characterizing galaxies in heterogeneous formats and from different sources, images, FITS<sup>1</sup> and spectra data;
2. Build and train a set of DL models for inferring properties of interest of galaxies, given the data compiled in the previous step, using different combinations of input data for different properties outputs;
3. Combine the models and data processing capabilities developed in the previous steps to create ready to use, pipelines for systematic galaxy characterization.

---

<sup>1</sup>Flexible Image Transport System (FITS) is the standard format for astronomical images and derived data.

## 1.4 Main Contributions

The overarching goal of this work is to contribute to enhance the characterization and understanding of galaxies, a summary of the main contributions follows:

- Increased awareness for exploring DL techniques in the context of astrophysics applications, the creation of shareable datasets and models, and easily reproducible data processing techniques and workflows;
- A compiled dataset with SDSS observations from around 100k objects classified as galaxies with data from different sources, including RGB images, FITS and spectral data<sup>2</sup>;
- A collection of DL models with single and multi inputs/outputs that can be used stand alone or combined together to infer several properties of galaxies objects<sup>3</sup>;
- A generic Python package and framework for developing pipelines, including helper modules for dealing with SDSS data and a set of pipelines for systematically processing data readily available<sup>4</sup>;
- An online application for exploring the collection of available models and pipelines, allowing to readily infer and process SDSS objects data<sup>5</sup>.

All packages, models, applications and resources are shared publicly and under MIT or CC permissive licenses, that immediately enables anyone to use them in their own work or research without restrictions or limitations.

## 1.5 Document Outline

A brief outline of the remaining chapters of this document follows:

**Chapter 2** introduces some topics concerning the study and analysis of galaxies, and discusses some of the properties that can be explored to enable a deeper understanding of galaxy formation and evolution;

---

<sup>2</sup>Available from: <https://doi.org/10.5281/zenodo.6393487> (last accessed: 2022-10-18).

<sup>3</sup>Available from: <https://github.com/nunorc/astromlp-models> (last accessed: 2022-10-18).

<sup>4</sup>Available from: <https://github.com/nunorc/astromlp> (last accessed: 2022-10-18).

<sup>5</sup>Available from: <https://nunorc.github.io/astromlp-app> (last accessed: 2022-10-18).

**Chapter 3** discusses some ML and DL background concepts used throughout this work, and introduces some artefacts and techniques used in the following chapters;

**Chapter 4** describes and illustrates the devised models to infer properties of interest, and the new dataset created to train, validate and test the models;

**Chapter 5** describes the implementation of pipelines for galaxy characterization using the previously described models and resources and how they can be applied;

**Chapter 6** ends the document with some final remarks and trends for future work.

A couple of additional remarks concerning the content of this document. Mainly during Chapter 4 and 5, to clearly define and discuss models, data inputs and outputs, and pipelines integration the Haskell programming language is used as a specification language. Most of the times the Haskell syntax is followed strictly, i.e. the code can be executed by a compiler, but sometimes in order to increase readability some details are simplified, resulting in non-valid syntax, but hopefully easier to be read by humans. Appendix G provides a brief introduction to the Haskell notation, emphasizing the most common expressions and statements used in this document.



## Chapter 2

# Understanding Galaxies

Galaxies are one of the fundamental building blocks of the current description of the Universe. A galaxy can be described as a gravitationally bound collection of gas, dust, stars, and other exotic matter. They exist in many shapes and forms and are found mostly in pairs or groups, which sometimes are bound together in clusters, groups and clusters may form larger systems, superclusters. Properties like form, size and mass are related with the physical phenomena that drive the processes responsible for galaxies formation and evolution, and understanding these processes enables a deeper understanding of galaxies ([Karttunen et al., 2007](#)). The remaining sections of this chapter discuss some of the properties later addressed using Deep Learning (DL) models in more detail.

### 2.1 Imaging & Photometry

Astronomical imaging encompasses all the tasks (e.g. planning, calibration) of capturing photons using a telescope and a CCD camera or other apparatus; while photometry is a field of study more concerned with measuring the actual amount of electromagnetic radiation captured from astronomical objects ([Gallaway, 2016](#)).

Luminosity and flux are some of the quantities addressed by the field of photometry. Luminosity refers to the total amount of energy radiated by an object per unit time. Flux is the amount of energy from a luminous object that reaches a given area or surface. Luminosity and flux are related mathematically by the following equation:

$$F = \frac{L}{4 \pi r^2}, \quad (2.1)$$

where, the flux  $F$  is given by dividing luminosity  $L$  by the surface of a sphere with radius  $r$ , i.e. the distance to the object, assuming the source is radiating evenly in all directions. For further distances this relation should also include a factor to correct for the spectral distance, usually the K-correction ([Hogg et al., 2002](#)) is used. Typically the flux is measured using some kind of apparatus, e.g. a detector in a telescope, and if the distance to the object is known, its luminosity can be calculated for example, generally only well defined intervals of wavelengths, sometimes referred to as bands, are measured ([Sparke & Gallagher III, 2007](#)).

Morphological classification, cosmological redshift, galaxy activity are some of the properties of interest that are calculated from the heterogeneous fluxes, in different bands, retrieved using imaging and photometry techniques. The remaining sections of this chapter discuss some of these properties in more detail.

## 2.2 Morphological Classification

Morphological classification of galaxies from images, based on their appearance and form, is a characteristic studied to understand galaxy formation and evolution. As a first approach, galaxies can be roughly divided in two major groups: historically named as *early-type* galaxies, with an older stellar population, *redder* in the optical; and, *late-type* galaxies, *bluer* in the optical, with a younger stellar population; but more complex classification schemas are available in the literature ([Sparke & Gallagher, 2000](#)).

The classification system proposed by [Hubble \(1936\)](#), illustrated in Figure 2.1, is the first proposed schema to classify galaxies, it organizes objects by their form and appearance. The underlying interest of classification schemas is to group objects that share common characteristics in order to study their physical properties, some examples of groups follow:

- **Ellipticals:** smooth shaped early-type galaxies, bright in the optical red, and lacking any distinguishable features, very low ongoing star-formation;
- **Disks:** galaxies with a central bulge and a spiral like structure, may have other features like bars and rings, brighter in the optical blue, actively forming stars;
- **Lenticulars:** in-between elliptical and spiral galaxies, with a high bulge to disk ratio and little ongoing star formation;

- **Irregulars:** galaxies without a well defined structure or features, that do not fit in any of the other groups.

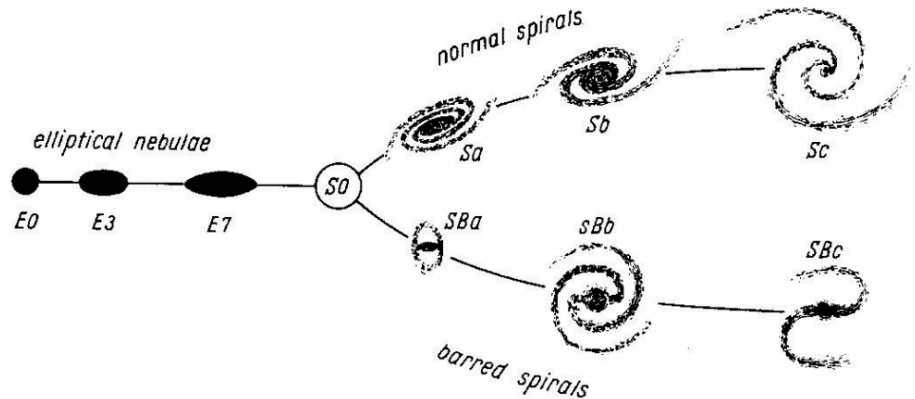


Figure 2.1: Hubble classification diagram, where three major branches of galaxies are illustrated: ellipticals (E) to the left, disk galaxies (S) to the right, and lenticular galaxies (S0) in the fork (Hubble, 1936).

The schema proposed by Hubble was later expanded by [de Vaucouleurs \(1959\)](#), to include specific classes to cope better with the presence and prominence of spiral arms, bars and rings. [Conselice \(2003\)](#) proposes another classification system based on the quantitative measure of three parameters: concentration (C), asymmetry (A), and *clumpiness* (S), referred to as the "CAS" system, able to clearly distinguish galaxies in various phases of evolution.

Independently of the schema used, traditionally, the classification of galaxies is a task performed by humans, by visually inspecting two dimensional images. With recent developments in Machine Learning (ML) techniques, technological improvements of astronomical instruments, the advent of citizen science projects like Galaxy Zoo 2 ([Willett et al., 2013](#)), and the increased availability of labelled data, i.e. observations that has been tagged with one or more labels, e.g. images of galaxies with the corresponding morphological classification label, other methods have been proposed to handle such tasks more automatically and exploring different data. To cite just a few examples related with this work: morphological analysis with unsupervised learning from multi-band photometric imaging by [Martin et al. \(2020\)](#); classification from spectroscopic data by [Ball et al. \(2006\)](#); morphological classification using DL networks by [Cavanagh et al. \(2021\)](#); [Dieleman et al. \(2015\)](#) present a deep neural network model for galaxy morphology classification; and,

[Zhu et al. \(2019\)](#) propose a variant of residual networks for galaxy morphology classification.

## 2.3 Spectral Analysis

Spectroscopy is a field of study concerned with measuring and studying the electromagnetic signatures that result from the interaction of electromagnetic radiation with matter. Several methods and techniques for analysing galaxy spectra details can be used to extract information about the observed object. By comparing the absorption lines and emission lines with a broad continuum emission it is possible to determine for example: the velocity and cosmological redshift from the lines' shift with regard to their reference positions; the relative abundances of elements from the relative strength and width of the lines; or the pressure, density and rotation from the shapes of the lines ([Gallaway, 2016](#)).

### 2.3.1 Cosmological Redshift

Spectral lines of light from moving galaxies, travelling in the form of electromagnetic radiation in a stretching spacetime, get shifted towards the red by an amount proportional to their distances, this phenomena is referred to as the cosmological redshift ([Bunn & Hogg, 2009](#), [Whiting, 2004](#)).

The dimensionless redshift  $z$  can be directly computed from the spectrum, by measuring the shift of some identified line(s) with respect to the expected wavelength as measured in a laboratory using the following relation:

$$1 + z = \frac{\lambda_{\text{obj}}}{\lambda_{\text{ref}}} \quad (2.2)$$

where,  $\lambda_{\text{obj}}$  is the wavelength of the spectral feature from the distant object and  $\lambda_{\text{ref}}$  is the wavelength of the spectral feature observed in a laboratory on Earth ([Karttunen et al., 2007](#)). As an example Figure 2.2 illustrates the spectrum for a galaxy<sup>1</sup>, where the solid vertical orange line identifies the wavelength for the  $H_{\alpha}$  spectral feature from the distant object, where  $\lambda_{\text{obj}} = 7222.0 \text{ \AA}$ , and the dotted vertical orange line identifies the wavelength for the  $H_{\alpha}$  spectral feature observed in a laboratory on Earth, where  $\lambda_{\text{ref}} = 6563.8 \text{ \AA}$ . From the relation in Equation 2.2 is immediate to calculate that for this observation the redshift  $z = 0.1$ .

<sup>1</sup>Object identifier 1237666407379828976, data retrieved from the SDSS database.

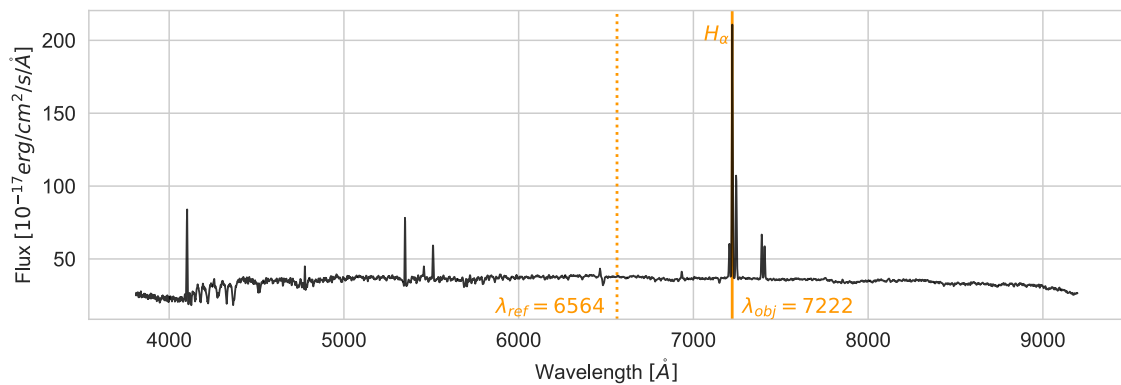


Figure 2.2: The shift towards the red from the wavelength of the spectral feature observed in a laboratory on Earth, where  $\lambda_{\text{ref}} = 6563.8 \text{ \AA}$ , to the wavelength of the spectral feature from the distant object, where  $\lambda_{\text{obj}} = 7222.0 \text{ \AA}$ .

When processing spectra in bulk, for large surveys of data, its not feasible to identify the spectral features of interest against the spectrum continuum manually, a more systematic approach is required. There are two major trends available in the literature:

- Template fitting methods, the first and most common approach, in which a Spectral Energy Distribution (SED) is matched against a set of templates for different galaxy types and redshifts, the best match found determines the redshift, the method of maximum likelihood is predominantly used, trying to find the match that minimizes the  $\chi^2$  when comparing the observed magnitudes with the magnitudes derived from the templates (Bolzonella et al., 2000). Examples that follow this approach include the works by Arnouts et al. (2002), Benítez (2000), Brammer et al. (2008), Coe et al. (2006) or Ilbert et al. (2006);
- ML based approaches, appearing later, where a smaller set of observations with known redshifts is used to train a model that is later used to infer the redshift for observations of interest, for examples that follow this approach refer to the works by Collister & Lahav (2004), Gerdes et al. (2010) or Carrasco Kind & Brunner (2013).

There are advantages and disadvantages for both approaches, and there are also examples of combined approaches, for more details on a comparison between these refer to Sánchez et al. (2014).

### 2.3.2 Stellar Mass

The stellar mass of a galaxy encompasses the sum of the mass of all the stars gravitationally bound to it, including stars still undergoing nuclear fusion, as well as stellar remnants, e.g. white dwarfs, neutron stars. Measuring and studying the distribution of stellar mass at a cosmological scale, helps understanding and testing the evolution of galaxies through time, since typically the mass of a system is its main evolution driver (Courteau et al., 2014).

Although there are several approaches currently available in the literature for estimating the stellar mass, the more popular are computed using a SED fitting approach. A SED conveys information about a galaxy stellar population by describing the distribution of light emitted by all the stars across all wavelengths, which helps to compute the contribution to the overall radiated energy from the stars that are *shining*, this can be observed and also modelled. There are two main ingredients for estimating the stellar mass using this approach: (i) a set of model templates for different stellar populations, and (ii) a fitting method to find the best match between the observed data and the templates. Examples of template sets include the works by Bruzual & Charlot (2003) and of fitting methods the works by Bolzonella et al. (2000)

Typically, the redshift is relevant and required for most approaches and techniques, it can be just another free parameter in the model, i.e. the value is also found during fitting, or it can be provided outside the scope of the model, i.e. just another input variable, this usually depends on if the redshift is already available or not.

## 2.4 Starbursts & Active Galaxies

The Star Formation Rate (SFR) of a galaxy measures the total mass of stars formed, on average, per unit time, usually a year, and is calculated based on flux measures at different wavelengths, e.g. ultraviolet continuum emission from hot stars. When a galaxy experiences an intense SFR, usually somewhere between 10 to 100 times higher than the average SFR for normal galaxies, is referred to as a starburst. The intense star formation is typically concentrated in small regions and can be identified by strong ionized hydrogen emission lines on a blue stellar continuum (Mo et al., 2010). Figure 2.3 illustrates the spectrum for an object from the SDSS database classified as a starburst galaxy<sup>2</sup>.

---

<sup>2</sup>Object identifier 1237668335783116981, data retrieved from the SDSS database.

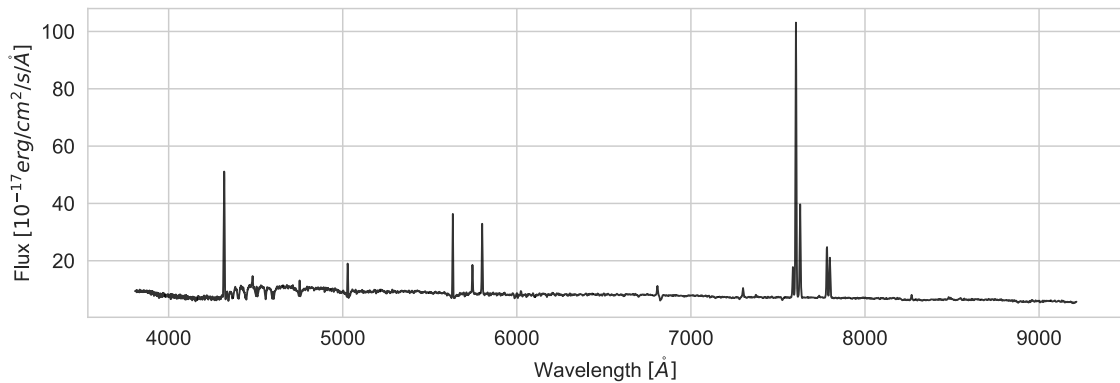


Figure 2.3: Example spectrum for a galaxy classified as starburst from the SDSS database.

An Active Galactic Nuclei (AGN) is a compact region at the centre a galaxy that emits radiation across the entire range of the electromagnetic spectrum, opposed to the considered *normal* galaxies where most of the light comes from stars, gas and dust, thought to be powered by matter accreting onto a Super Massive Black Hole (SMBH). A galaxy hosting an AGN is usually referred to as an active galaxy, examples include Seyfert galaxies, radio galaxies, and quasars. The spectra of many AGN is usually identified by strong emission lines and non-thermal radiation covering the entire electromagnetic spectrum from radio to gamma-ray, including X-rays, ultraviolet and visible radiation. Some spectral features of active galaxies may vary according to the orientation through which they are observed (Mo et al., 2010, Robson, 1996). Figure 2.4 illustrates the spectrum for a galaxy from the SDSS database classified as an AGN<sup>3</sup>, showing characteristic strong emission lines, e.g. [OII], H $\alpha$ , [NII], over a lower, less flat, continuum with shallow absorption lines.

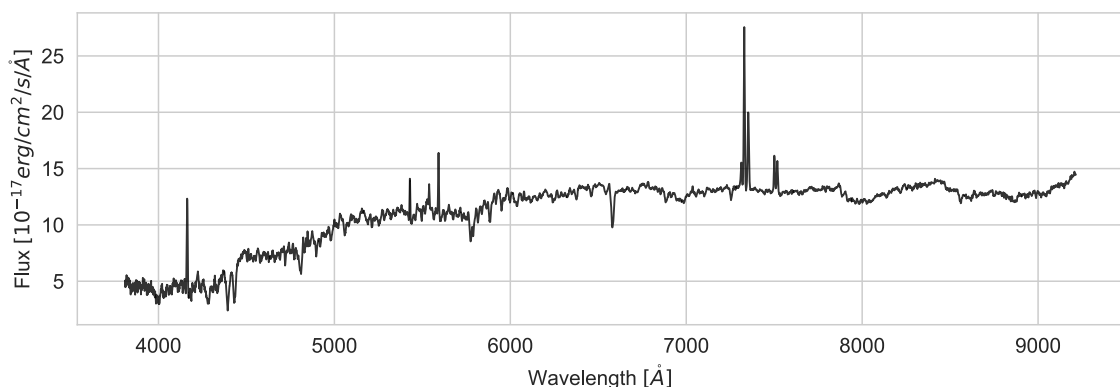


Figure 2.4: Example spectrum for a galaxy classified as AGN from the SDSS database.

<sup>3</sup>Object identifier 1237663548519088286, data retrieved from the SDSS database.

## 2.5 Observational Surveys, Catalogues & Projects

Studying objects like distant stars and galaxies is fundamental to understanding the Universe, but these can not be studied in a laboratory. Indirect measures from observations is one of the sources of information, but these are hard to make accurately and prone to errors. Sometimes the only way to study some phenomena is to perform statistical or ML based analysis, but these require large amounts of data (observations). This section briefly introduces some resources that provide large volumes of data that enable the use of the techniques explored throughout this work.

### 2.5.1 Sloan Digital Sky Survey (SDSS)

The Sloan Digital Sky Survey (SDSS) project is a survey collecting observations for millions of sources primarily at optical wavelengths (Blanton et al., 2017). The project encompasses several instruments, like the 2.5 meters Sloan Foundation Telescope (Gunn et al., 2006), and the multi-object fibre spectrographs capable of obtaining 1000 spectra simultaneously (Smeed et al., 2013), both mounted at Apache Point Observatory in New Mexico, United States of America. The project periodically provides data releases, that provide access to the collected observations, and also to the result of other programs, catalogues, for example, the Baryon Oscillation Spectroscopic Survey (BOSS) (Dawson et al., 2013). At the time of writing this document the last data release made available is DR17 (Abdurro'uf et al., 2022).

Some relevant SDSS data used throughout this work includes: objects classification and redshift resulting from the pipeline discussed by Bolton et al. (2012); photometric data resulting for the observed bands u, g, r, i and z, Table 2.1 summarizes the bands central wavelengths and corresponding full widths at half maximum (Fukugita et al., 1996); and, the stellar mass from the eBOSS Firefly catalogue (Comparat et al., 2017) calculated using the FIREFLY tool (Wilkinson et al., 2017).

For more details and information refer to the SDSS official website<sup>4</sup>, where all the data is made available through different services in a heterogeneous set of formats.

---

<sup>4</sup>Available from: <https://sdss.org> (last accessed: 2022-10-18).



| Band       | u      | g      | r      | i      | z      |
|------------|--------|--------|--------|--------|--------|
| Wavelength | 3500 Å | 4800 Å | 6250 Å | 7700 Å | 9100 Å |
| FWHM       | 600 Å  | 1400 Å | 1400 Å | 1500 Å | 1200 Å |

Table 2.1: SDSS photometric data central wavelengths and corresponding full widths at half maximum for bands u, g, r, i and z.

### 2.5.2 Wide-Field Infrared Survey Explorer (WISE)

The Wide-Field Infrared Survey Explorer (WISE) survey is a mid-infrared all sky survey mission funded by NASA. The 40cm satellite mounted telescope, mapped the whole sky in four infra-red bands, referred to as W1, W2, W3 and W4, Table 2.2 summarizes the bands central wavelengths and corresponding full widths at half maximum (Rodrigo et al., 2012, Wright et al., 2010).

| Band       | W1       | W2       | W3        | W4        |
|------------|----------|----------|-----------|-----------|
| Wavelength | 34 655 Å | 46 443 Å | 132 156 Å | 222 229 Å |
| FWHM       | 6358 Å   | 11 073 Å | 62 758 Å  | 47 397 Å  |

Table 2.2: WISE photometric data central wavelengths and corresponding full widths at half maximum for bands W1, W2, W3 and W4.

For more details and data refer to the WISE website<sup>5</sup>. The data is also available from the SDSS database, including a best match between objects identifiers for both programs.

### 2.5.3 Galaxy Zoo 2 (GZ2)

The Galaxy Zoo 2 (GZ2) project is a crowd-funded project where volunteer citizens manually annotate morphological features of SDSS images of galaxies at a large scale (Willett et al., 2013).

The dataset contains the votes to a set of questions, e.g. “*Is the object a smooth galaxy, a galaxy with features/disk or a star?*”, “*Is there a spiral pattern?*”, from human volunteers. From a statistical analysis of these votes, labels are created for each galaxy conveying morphological information, for example the presence of specific features like bars or spiral features Hart et al. (2016). The volume of data available enables exploring the use of ML techniques for task automation, e.g. galaxy classification, for some examples refer to the works by Zhu et al. (2019) or Polsterer et al. (2012).

<sup>5</sup>Available from: <https://irsa.ipac.caltech.edu/Missions/wise.html> (last accessed: 2022-10-18).

For more details about the project refer to the GZ2 official website<sup>6</sup>, for more details about the data and the complete dataset refer to the data website<sup>7</sup>. Most of the compiled data is also available immediately from the SDSS database.

---

<sup>6</sup>Available from: <https://galaxyzoo.org> (last accessed: 2022-10-18).

<sup>7</sup>Available from: <https://data.galaxyzoo.org> (last accessed: 2022-10-18).

## Chapter 3

# Deep Learning Essentials

*A computer would deserve to be called intelligent if it could deceive a human into believing that it was human.*

– Alan Turing

Artificial Intelligence (AI), in a nutshell, refers to the ability of machines to perform intelligent, human-like behaviour, tasks. This chapter introduces and reviews some fundamental topics in the sub-fields of Machine Learning (ML) and Deep Learning (DL) closely related with this work, starting with some definitions ([Janiesch et al., 2021](#)).

### **Artificial Intelligence**

AI is a broad field of study concerned with automating intellectual tasks, usually performed by humans. This spawns a wide range of applications, for example: a computer program playing chess, a human robot walking, a satellite in orbit taking pictures of the Earth, a mobile phone application providing restaurant suggestions, and so on. In a nutshell we can devise this artefact that displays some human-like behaviour, usually referred to as an agent, as being in an environment and as inputs are received from the environment some action is performed by this agent, given some previous knowledge.

### **Machine Learning**

ML is a sub-field of AI concerned with the study of algorithms and techniques that discover patterns in data. By learning these patterns from real data, a model can capture the

underlying mapping between the inputs and the outputs. A widely accepted more formal definition follows:

“A computer program is said to learn from experience  $E$  with respect to some class of tasks  $T$  and performance measure  $P$ , if its performance at tasks in  $T$ , as measured by  $P$ , improves with experience  $E$ ” (Mitchell, 1997).

More informally, ML enables a computer to learn how to perform a task without being explicitly programmed how to do it.

### Deep Learning

DL, a sub-field of ML, is a broad term encompassing the study of algorithms and techniques exploring models implemented as a (deep) composition of layers where the output of one layer is passed to next layer consecutively building new representations of the data until the output layer computes the final result (Goodfellow et al., 2016).

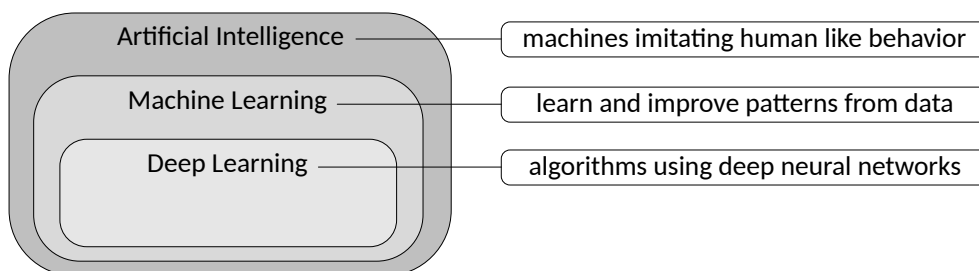


Figure 3.1: Fundamental relations between the fields of AI, ML and DL.

Figure 3.1 illustrates the fundamental relations between AI, ML and DL, and how they fit together. The next sections discuss some aspects from these fields in more detail, focusing on ML and DL.

## 3.1 Fundamentals of Machine Learning

ML introduced a new paradigm for developing applications. In more traditional approaches a computer is given some data and the rules to produce some answers (outputs). When using ML the paradigm changes, the computer is given some data and the answers, and

it produces the rules, i.e. a mapping that describes the pattern that goes from data to answers. In this sense, the computer "learns" the rules from the data, opposed to being explicitly programmed.

This function, or mapping, between inputs and outputs is usually captured in a model and a ML technique or approach is used to build these models. Some examples of common ML techniques include: linear regression, decision trees or neural networks.

### 3.1.1 Defining the Problem & Major Branches

Most of the available literature distinguishes three major branches of ML, in no particular order: supervised learning, unsupervised learning and reinforcement learning, but of course other more detailed subdivisions are possible. Each branch addresses some typical type of task, and usually employs some well know techniques. Hence, usually when addressing a new problem, the first step is to define in which branch (or category) the problem falls into. This usually entails which techniques are more suitable to use. There is no hard line between these branches and some problems may cross different learning approaches but clearly defining the problem, and the context, is a required step for applying ML. A brief description of the major branches follows ([Janiesch et al., 2021](#)).

#### Supervised Learning

In supervised learning the inputs and outputs are known during training, the goal of the ML algorithm is to build a mapping, a function, between the inputs and the outputs, sometimes also referred to as the ground truth, that is later used to predict outputs given new unseen inputs. The most common tasks in supervised learning are regression and classification. In regression problems the output of the model is a continuous variable, e.g. predict the temperature, salary; in classification problems the output variable is discrete, i.e. predict the most probable label, also known as category or class, e.g. spam or no-spam, cat or dog. Common models in this family of problems include linear regression, decision trees and neural networks.

### Unsupervised Learning

In unsupervised learning only the inputs of the problem are available. In this case the goal of the ML becomes finding some underlying structure in the data. A typical example is clustering, e.g. find groups of users with similar tastes, or group books by topic; recommender systems can be another example of unsupervised learning, e.g. recommend friends on social networks, recommend items of interest in a grocery store. Common algorithms in this family of problems include for example k-means and hierarchical clustering.

### Reinforcement Learning

In reinforcement learning the gist is *learn by doing*, in this family of problems the algorithm learns how to perform a task using a reward system, where the algorithm is rewarded when the result of an action, or sequence of actions, has a positive outcome, and penalized otherwise. The main intuition behind this approach is that given a sufficiently large number of iterations eventually the model starts to learn the best actions to take in specific settings as to maximize some notion of cumulative reward, e.g. learning how to play a game. Common techniques in this family of problems include for example Markov Decision Process and Q learning.

## 3.2 Artificial Neural Networks & Deep Learning

An Artificial Neural Network (ANN) is a model used in ML built using a network of nodes, stacked in consecutive layers, connected by edges. Networks get some input data, and outputs a representation of data in a more useful form in the context of the problem at hand. Each node in each layer has its own weights, bias and activation capable of learning a feature of the input it gets. When a significant number of hidden layers exist, i.e. layers that stand between the input and the output layers, we enter the field of DL. The main intuition behind this concept is that each layer learns a more useful representation of the data before passing it on as input to the next layer, by successively transforming the data in more useful representations the network is capable of finding a statistical structure for mapping inputs to outputs (Chollet, 2018). Figure 3.2 illustrates a deep neural network model with three input nodes, one output node, and three fully connected hidden layers in-between.

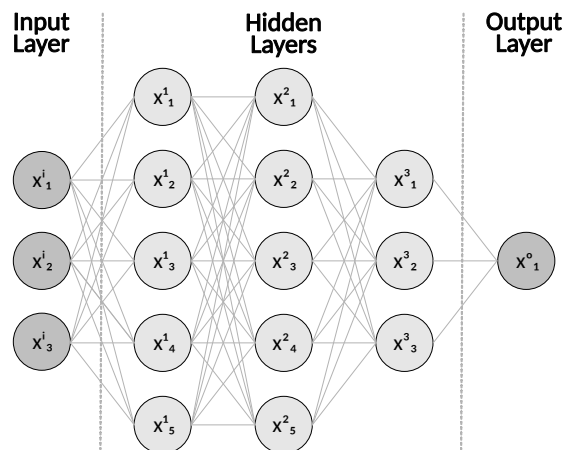


Figure 3.2: Artificial neural network example with three fully connected hidden layers.

The gist of building a DL model to perform a specific task in a given context revolves mainly around defining four main components:

1. The model input and output;
2. The network architecture – the combination of layers that define the model;
3. The loss function – to evaluate the model;
4. The optimizer – to update the model.

The input data and model output immediately derive from the problem at hand. For example for an image classification problem, the input is an image of a given size, and the output is a label. The following sections discuss each one of the other components in more detail.

### 3.2.1 Network Architecture & Composition of Layers

In the context of DL a model is a directed, acyclic graph of layers, where the most common instance is a linear stack of layers, mapping multiple input nodes to output nodes. In a nutshell a layer is a function that maps inputs into outputs. Each layer has a number of nodes that describe the shape that the data has before and after being processed. In a broad sense a model is also a function that maps inputs into outputs, and the network architecture of the model is a combination of layers, where the output of one layer is the input to the next, until the final output is computed.

A layer is a construct that is used to define the model architecture. Composition of layers entails connecting a sequence of layers in order, where the output of the previous

layer is the input to the next one. A shape is usually associated with a layer, used to define the dimensions of the layer input and output. Layers inputs and outputs can have  $n$  dimensions and different shapes, layers can also have inherent different structures. Every network model has at least one input layer, this is the first layer in the network and it has as many nodes as required by the input data, usually the shape of the layer is defined by the type of data; and also at least one output layer, that has as many nodes as the desired outputs, for example if a model is predicting temperature and this is defined as a regression problem, the final layer has one node – the temperature. An arbitrary number of layers of different types, referred to as hidden layers, may exist between the input and output layers that are used to build a function for mapping the inputs to the outputs.

### Types of Layers

Some other commonly used layers include ([Chollet, 2018](#)):

- **Fully connected:** a fully connected layer, or sometimes also referred to as a dense layer, is a layer where every node is connected to every other node, this is typically the default type of hidden layer in most libraries, the input shape of the layer is defined by the previous layer, i.e. it is the same, and the output shape of the layer defines the input of the next one. This kind of layer usually implies that all the nodes in the previous data representation may affect all nodes in the current representation, leaving to each nodes' weights the responsibility to define the importance of each edge during the training process.
- **Convolutional layer:** is a specific type of layer that is commonly used to deal with images data, by applying special filters, usually called kernels, to the input a feature map is created that, in a nutshell, summarizes the features of interest found in the input. Note that the kernels used are usually created also during the training process of a network, i.e. they are made of parameters themselves and so the feature maps are specific to the problem and context at hand.
- **Pooling layer:** is commonly used in combination with Convolutional layers to reduce the dimensions of the feature maps, by summarizing the feature map<sup>1</sup> using

---

<sup>1</sup>Feature map refers to the output of a convolutional layer, these include the relevant patches of the previous layer that convey features of interest to the task at hand.



some given mathematical operation, e.g. average, maximum; this also helps finding specific features in different locations of the input.

- **Dropout layer:** its a special layer in the sense that it has no parameters to train; the goal of this layer is to randomly remove nodes from the network. This is used mainly as a regularization technique to prevent over-fitting of the network, or sub-networks, to the training data. The underlying intuition is that by randomly dropping nodes in the network, some edges between some nodes disappear making other weights have to deal with seldom seen cases ([Hinton et al., 2012](#)).
- **Normalization layer:** its another special layer in the sense that there are no parameters to train; the goal of a normalization layer is to normalize the input data, i.e. downscale the values to a more useful scale, usually small numbers around 0 is best for the inner mechanics of neural networks. This is usually done at the beginning of the network but it can also be used with other purposes in the middle of the network.

### Activation Functions

An activation function, required for every layer, defines the output of a node given the node internal representation of the data, and it allows the network to learn more complex (non linear) patterns. The output of the activation function is actually the output of the node that becomes the input on the nodes in the layer. Some common activation functions include ([Nwankpa et al., 2018](#)):

- **Linear:** the linear activation function is proportional to the input, it can be defined as:

$$a(x) = x . \tag{3.1}$$

- **Sigmoid:** accepts arbitrary number of inputs, and outputs a number between 0 and 1, mainly used in binary classification problems:

$$a(x) = \frac{1}{1 + e^{-x}} . \tag{3.2}$$

- **Rectified Linear Unit (ReLU):** is one of the most commonly used activation functions

$$a(x) = \max(0, x) . \tag{3.3}$$

- **Softmax**: is actually a more generalized form of a sigmoid applied to the output layer, typically used for multi-class classification, and it computes the relative probability of belonging to a class or label, each node in the output layer produces a value between 0 and 1, and the sum of all the output nodes is 1.
- **Hyperbolic Tangent (TanH)**: accepts arbitrary number of inputs, and outputs a number between -1 and 1, it can be defined as:

$$a(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} . \quad (3.4)$$

### 3.2.2 Loss Functions

The goal of a loss function, also known as objective function, is to measure a model performance by calculating some measure of difference between the values computed by the model and the actual true values from the data, also known as the ground truth. This usually entails that the lower value comes out of a loss function, the better the model is performing ([Wang et al., 2022](#)). So, the goal of the training process becomes minimizing the loss function output. Some popular loss functions for regression problems, i.e. models that output a real value, include:

- **Mean Squared Error (MSE)**: is a loss function that computes the average squared difference between predictions and true values, the result of the function will get closer to zero as the difference between the predicted values ( $\hat{Y}_i$ ) and the true values ( $Y_i$ ) gets smaller:

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2 . \quad (3.5)$$

- **Mean Absolute Error (MAE)**: is a loss function that computes the mean of absolute difference between predictions and true values:

$$MAE = \frac{1}{n} \sum_{i=1}^n |Y_i - \hat{Y}_i| . \quad (3.6)$$

Some popular loss functions for classification problems include:

- **binary cross-entropy**: is a loss function that computes the mean of squares of errors between labels predictions for two classes problems;
- **categorical cross-entropy**: is a loss function that computes the mean of absolute difference between labels and predictions for multi-class problems.

### 3.2.3 Optimizers

An optimizer is responsible for computing the weights updates for the model on every iteration during model training, usually called an epoch, in order to reduce the model loss. A common parameter to these algorithms is the learning rate, used to scale the model weights updates, by setting an high learning rate the model may converge faster but may also overshoot the function minimum, by setting a lower learning rate the model may converge too slowly. Some popular optimizers include ([Deisenroth et al., 2020](#)):

- **Gradient Descent** (GD): is one of the most common optimization algorithms used, the algorithm consistently moves to the local minimum of a function, proportional to the negative of the gradient at the current point, and scaled by the defined learning rate.
- **Stochastic Gradient Descent** (SGD): is a stochastic approximation of the GD method, by only randomly selecting a batch of data this algorithm is faster.
- **Adaptive Gradient Descent** (AdaGrad): is another variation of GD, where the learning rate depends on the gradient at the current point, leaving to the algorithm the job of updating the learning rate making the model converge faster ([Duchi et al., 2011](#));
- **Root Mean Square Propagation** (RMSProp): is an extension of the SGD, popular in DL, using an adaptive learning rate, scaled by an exponentially decaying average of squared gradients;
- **Adaptive Moment Estimation** (Adam) is another extension of SGD, that computes adaptive learning rates for each parameter, similar to RMSProp ([Kingma & Ba, 2014](#)).

In a nutshell there are pros and cons to each algorithm and there is no generic optimal solution, depending on the specific details of each problem and related data any of these algorithms can outperform the others. Besides the actual algorithm there are still a variable number of hyper-parameters that can be fine-tuned, the learning rate for example.

### 3.2.4 Training, Validation & Testing

Given a model, a loss function and an optimizer the training of the model is the process by which the model weights are calculated given the training data. This process is usually divided in epochs, i.e. every time the model processes all the data an epoch has passed.

Within each epoch, the data is processed in steps, the batch size defines the number of observations, or samples, of the entire input data that are processed on each step. For example, training a model for 10 epochs with a batch size of 32, entails that the entire dataset is processed 10 times, and on each time its analysed in steps, using batches of 32 observations from the input dataset. On each step the following operations are executed:

1. The forward propagation is done, where the model computes the network output for a number (batch size) of samples;
2. The loss function is used to calculate the error of the outputs;
3. The error is propagated backwards in the network during back propagation;
4. Given the loss update the model weights get updated by the optimizer.

This loop repeats for the defined number of epochs. On each epoch the validation set can be used to measure the model performance in samples that are not used during training, so that it has a better measure of the models' ability to generalize. This validation can also be used to fine-tune the model hyper-parameters. In the end, after the training process, the test set can be used to measure the model performance, by using data never seen during training.

## Normalization

Normalization includes a set of data transformations with the goal of setting all the input data to the same comparable scale, so that the contribution of the feature to the model output is dictated by its absolute value. For most DL models the usual desired distribution of values is around 0, between -1 and 1. This usually also entails speeding up the optimization process. Data normalization can be done before applying the model or a normalization layer can be used in the model network, as long as it done before training, some common normalization techniques include: *z-score* or *MinMax* (Yu & Spiliopoulos, 2022).

## 3.3 Advanced Techniques

This section introduces some more advanced techniques used throughout this work.

### 3.3.1 Hyper-parameters Tuning

In the context of DL besides the model definition there are other components that play an important role in building an actual model, and also have a deep impact on the model performance, for example the loss function, the optimizer or the batch size. Fine-tuning these parameters to achieve the best results is not an easy task because there are no rules *set in stone* to which option is always best for a given problem, and the hyper-parameters search space is very large. Hence, a common more pragmatic way to address this issue is by exploring a sub-space of the parameters space available, by using different combinations of hyper-parameters and comparing some models evaluation of performance. This can be done manually, usually using a smaller parameters sub-space but with faster results, or using a more systematic approach, covering a larger sub-space but a slower process.

### 3.3.2 Regularization

Over-fitting is a concept generally used in ML that represents the state of a model that is over trained to the fitting data, and has a high performance on this data but lower performance on data that the model does not see during train, i.e. the model is failing to generalize to unseen inputs. This is a major concern with neural networks.

Regularization is a set of techniques that address the problem of over-fitting by providing techniques that put constraints on the complexity of the model, ensuring the model weights and model complexity is kept low. Common regularization techniques include ([Chollet, 2018](#)): L1 regularization and L2 regularization that add a factor to the loss function based on the sum of the model parameters values, this entails penalizing parameters with higher values, trying to guide the optimization process to lower value parameters, usually less prone to over-fitting; and dropout layers, that randomly drop (up to some degree) some part of the network.

### 3.3.3 Data Generators

Some libraries available for DL require all the data to be loaded into memory for training the models. But, due to computers limited resources and given the huge size of datasets used today, sometimes its not possible to load the entire data to memory. One common approach to deal with this shortcoming is to load data as needed, i.e. on every epoch load the next data batch from disk. To accomplish this and following deep learning *parlance*, an

artefact called *data generator* is used. This enables, among others things, loading data into memory as needed for training, opposed to loading all the data before the training process. For example, when processing thousands or millions of images it enables loading the actual images data (or any other sources of data) to memory immediately before each step for processing, and discarding the images from memory after each step, opposed to loading all the images in the beginning and keeping them for the entire training process which would require a much higher amount of memory, most of the times not available. A positive side effect of this approach is that its possible to also do some custom processing task on the data after loading and before actually passing it to the model.

# Chapter 4

## Datasets & Models

*Divide et impera*<sup>a</sup>

– Philip II of Macedon

---

<sup>a</sup>“Divide and Rule” or “Divide and Conquer”.

The previous chapters introduce some background about galaxies, and some fundamental topics on Machine Learning (ML) and Deep Learning (DL). This chapter discusses the creation of a set of models to infer properties of interest about galaxies, and of a dataset devised to train and explore these models.

### 4.1 The SDSS Galaxy Subset

The SDSS Galaxy Subset (SDSS GS)<sup>1</sup> is a subset of around 100k objects classified as *GALAXY* from the Sloan Digital Sky Survey (SDSS) survey described in Section 2.5.1. The dataset includes tabular data, stored in a CSV<sup>2</sup> format where each row refers to an object and Table 4.1 summarizes the columns information available for each object. All the data in the table is gathered directly from tables in the SDSS database, except for the Galaxy Zoo 2 (GZ2) classification from Willett et al. (2013)<sup>3</sup>. Appendix F describes the SQL<sup>4</sup> queries required to retrieve the data from the SDSS official database. The GZ2

---

<sup>1</sup>Available from: <https://doi.org/10.5281/zenodo.6393487> (last accessed: 2022-10-18).

<sup>2</sup>Comma-Separated Values (CSV) is a common format for storing tabular data in a plain text file.

<sup>3</sup>Available from: <https://data.galaxyzoo.org> (last accessed: 2022-10-18).

<sup>4</sup>Structured Query Language (SQL) is a domain specific language for querying and handling data in a relational database management system.

simplified classification labels are a simplification of the original class set where some details are removed in order to reduce the total number of classes, the complete simplified class set including a brief description is available in Appendix D.

| Column      | Description  |
|-------------|--|
| objid       | unique SDSS object identifier  |
| mjd         | Modified Julian Date (MJD) of observation  |
| plate       | plate identifier   |
| tile        | tile identifier  |
| fiberid     | fibre identifier   |
| run         | run number   |
| rerun       | rerun number   |
| camcol      | camera column  |
| field       | field number   |
| ra          | object right ascension   |
| dec         | object declination   |
| class       | spectroscopic class (only objects with <i>GALAXY</i> class are included)         |
| subclass    | spectroscopic sub-class  |
| modelMag_u  | better of DeV/Exp magnitude fit for band u                                       |
| modelMag_g  | better of DeV/Exp magnitude fit for band g                                       |
| modelMag_r  | better of DeV/Exp magnitude fit for band r                                       |
| modelMag_i  | better of DeV/Exp magnitude fit for band i                                       |
| modelMag_z  | better of DeV/Exp magnitude fit for band z                                       |
| redshift    | final redshift from SDSS data z  |
| stellarmass | stellar mass extracted from the eBOSS Firefly catalogue, in units of $M_{\odot}$ |
| w1mag       | WISE W1 "standard" aperture magnitude  |
| w2mag       | WISE W2 "standard" aperture magnitude  |
| w3mag       | WISE W3 "standard" aperture magnitude  |
| w4mag       | WISE W4 "standard" aperture magnitude  |
| gz2c_f      | Galaxy Zoo 2 classification  |
| gz2c_s      | simplified version of Galaxy Zoo 2 classification                                |

Table 4.1: Summary of the columns information available from the tabular data in the SDSS GS dataset.

Besides the data there are also available four extra files for each object namely:

- A RGB image file for the object in JPEG format, generated using the SkyServer DR16 WebService<sup>5</sup>, with a scale of 0.4 arsec per pixel, and a height and width of

<sup>5</sup>Available from: <https://skyserver.sdss.org/dr16/en/help/docs/api.aspx> (last accessed: 2022-10-19).



150 pixels; the image is generated from the g, r, i bands based on an adaptation of the algorithm described by [Lupton et al. \(2004\)](#).

- The FITS original data, downloaded from the SDSS file server for u, g, r, i, and z bands, a cut is made around the object location. The cut is made using the Image-Cutter library<sup>6</sup>, the cut out size is the default size of 0.4 arc minutes in size on the x and y axis and the data is stored in a *numpy* array.
- The spectra data is also downloaded directly from SDSS data files in CSV format, the *BestFit* value between wavelengths 4000 Å and 9000 Å is usually used throughout this work, but the complete spectra is available.
- There is also another spectra file (*sse/*) that includes only a set of selected bandwidths of interest discussed by [Sánchez Almeida et al. \(2010\)](#).

The objects included in the dataset are sampled at random from the *SpecPhoto* table from the SDSS database. The only constraint on the sampling is that it is stratified on the *subclass* column, i.e. the subset contains approximately the same percentage of samples of each sub-class as the original table.

## 4.2 Exploratory Analysis

This section explores the SDSS GS dataset with a closer look to some of the variables later used in models as inputs and outputs. The dataset has a total of 100077 objects classified as galaxies, defined by the *class* variable. There is a sub-class for each object, the *subclass* column, that splits galaxies in different categories, Table 4.2 summarizes the four available sub-classes for classifying galaxies.

The sub-class information is available for all objects, Figure 4.1 illustrates the distribution of sub-classes for all objects, the STARFORMING sub-class is the most common with a total of 72213 objects, followed by STARBURST with a total of 15481 objects, and with the AGN and BROADLINE sub-classes applying to the remaining objects with a total of 6410 and 5973 respectively. These labels are not mutually exclusive, in the sense that they can be combined together to classify objects (although no multi-labels examples were included in the dataset) and are based on the spectra emission lines. Objects are

---

<sup>6</sup>Available from: <https://github.com/jhoar/ImageCutter> (last accessed: 2022-10-19).

| Label       | Description   |
|-------------|---|
| AGN         | has detectable emission lines that are consistent with being a Seyfert or LINER                     |
| BROADLINE   | has lines detected at the 10-sigma level with $\sigma > 200 \text{ km s}^{-1}$ at the 5-sigma level |
| STARBURST   | galaxy has an intense star-formation rate   |
| STARFORMING | has detectable emission lines that are consistent with star-formation criteria                      |

Table 4.2: Set of labels for the sub-class variable in the SDSS GS tabular data.

classified as AGN when  $\log_{10}([OIII]/H\beta) > 1.2\log_{10}(NII/H\alpha) + 0.22$ , otherwise they are classified as STARBURST if the  $H\alpha$  line width is greater than  $50 \text{ \AA}$ , or STARFORMING otherwise; the BROADLINE label is given when they have line widths in excess of  $200 \text{ km s}^{-1}$  (Baldwin et al., 1981, Bolton et al., 2012).

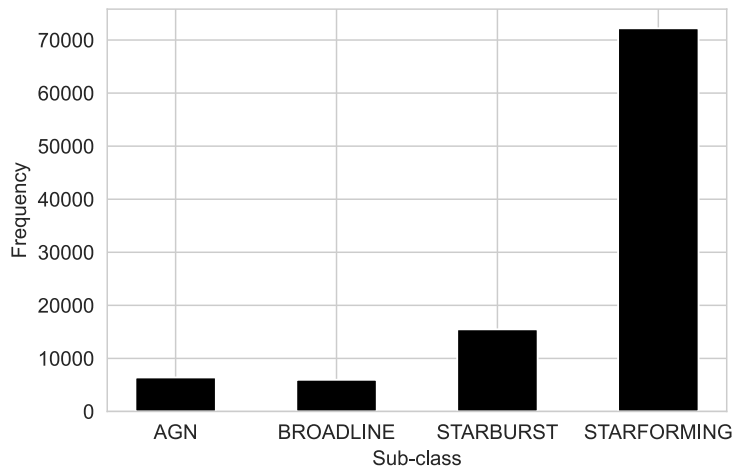


Figure 4.1: Distribution of the sub-class categorical feature, where the STARFORMING sub-class is the most common with a total of 72213 objects.

Concerning the continuous variables redshift and stellar mass Table 4.3 illustrates the statistical summary for the available data. The redshift and stellar mass values are available for all objects. The redshift values distribution has a mean value of 0.087883, with a standard deviation of 0.079735, a minimum value of 0.000042 and a maximum value of 1.584996. The stellar mass values distribution, in units of  $M_{\odot}$ , has a mean value of  $2.53 \times 10^{10}$ , with a standard deviation of  $4.45 \times 10^{11}$ , a minimum value of  $9.05 \times 10^2$ , and a maximum value of  $1.12 \times 10^{14}$ . This interval clearly has values that are not expected for a galaxy stellar mass, from a quick overview of other properties of some of these objects it

seems that the shortcoming is that they are not correctly classified as galaxies, and hence were included in the pool of randomly selected objects from the database.

|              | Count  | Mean                  | Std                   | Min                | Max                   |
|--------------|--------|-----------------------|-----------------------|--------------------|-----------------------|
| redshift     | 100077 | 0.087883              | 0.079735              | 0.000042           | 1.584996              |
| stellar mass | 100077 | $2.53 \times 10^{10}$ | $4.45 \times 10^{11}$ | $9.05 \times 10^2$ | $1.12 \times 10^{14}$ |

Table 4.3: Descriptive statistics for the redshift and stellar mass (in units of  $M_{\odot}$ ) data available from the SDSS GS dataset.

Figure 4.2 illustrates the distribution of values for the redshift variable, from a visual analysis of the figure is clear to see that the values are skewed to the lower range, there are more observations that are closer than far away, the majority of the values are lower than 0.2. Figure 4.3 illustrates the distribution of values for the stellar mass variable, from a visual analysis of the figure is clear to see that the values are again skewed to the lower range, there are more less massive galaxies than galaxies with higher mass, the majority of the values are lower than  $2.5 \times 10^{10} M_{\odot}$ .

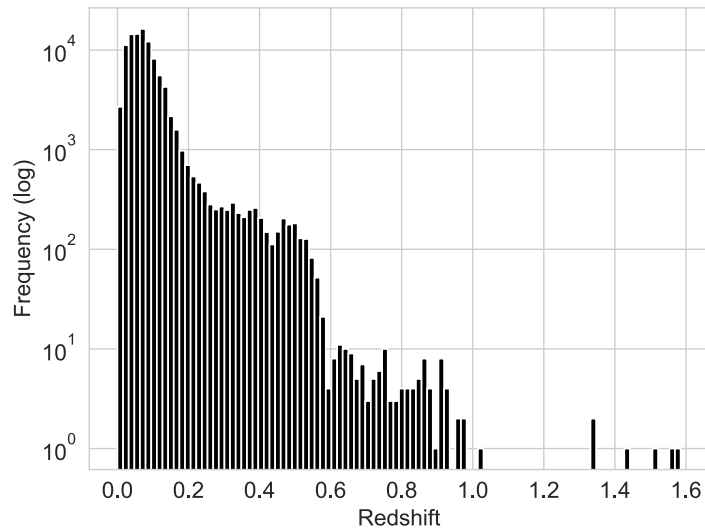


Figure 4.2: Distribution of values of the redshift variable, where the majority of values are located in the lower range, below 0.2.

Figure 4.4 illustrates the distribution of values for the GZ2 simplified class categorical feature, where the most common class is ScR (with features/disks, just noticeable bulge prominence, has spiral structure) with a total of 14646 objects, followed by the class Ei (smooth, in-between) with a total of 8606 objects.

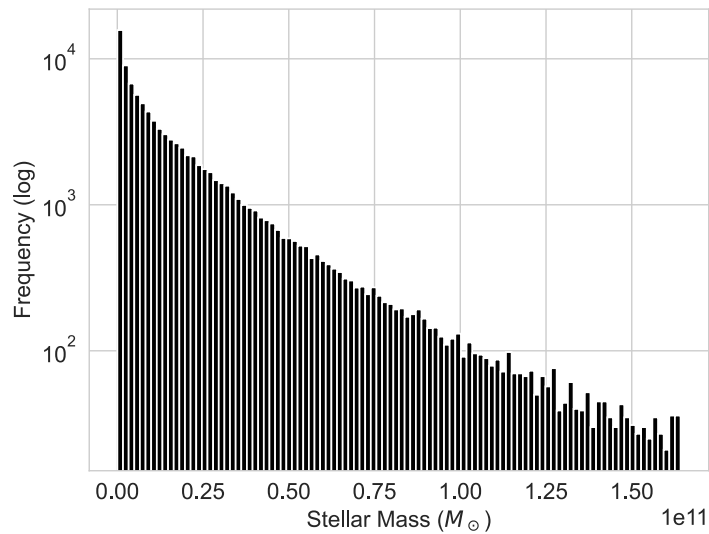


Figure 4.3: Distribution of values for the stellar mass variable, in units of  $M_{\odot}$ , where the majority of values are lower than  $2.5 \times 10^{10} M_{\odot}$ .

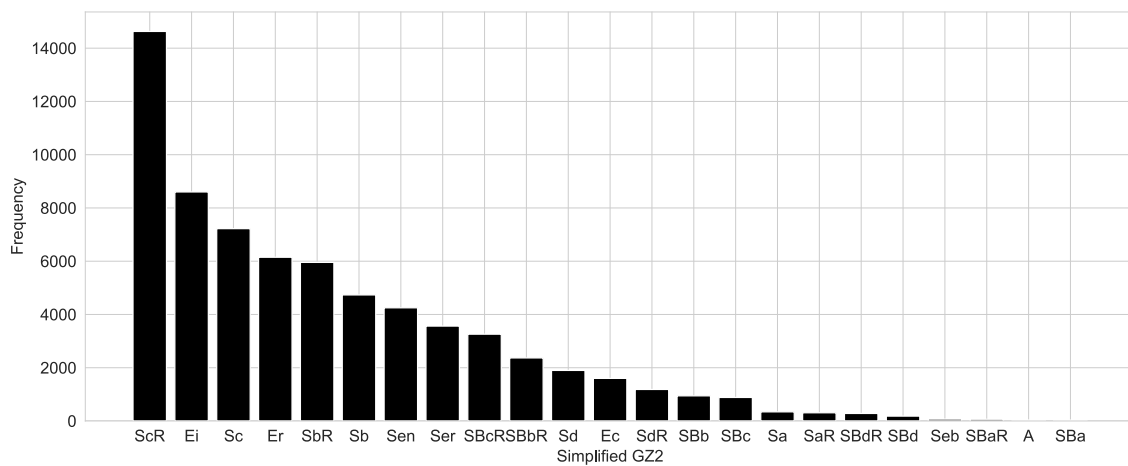


Figure 4.4: Distribution of the GZ2 simplified class categorical feature, where the ScR is the most common class with a total of 14646 objects.

### 4.3 Features, Inputs & Outputs

The data contained in the dataset described in the previous section is used as features for creating single and multi input/output models. Regarding features (inputs) pre-processing:

- The `img` data values are normalized by a factor of 255 when loaded;
- The `fits` data are used as is;
- The `spectra`, `sse1`, `bands` and `wise` data is normalized using a standard scaler;

- The `smass` data is down-scaled by a factor of  $10^9$ , this is used as a model output and in theory the actual range of values is not relevant to the inner model mechanics but just to make the updates smaller numbers and the training and validation metrics easier to follow and visualize.

Table 4.4 summarizes the data from the dataset used as input in the models, and Table 4.5 summarizes the data used as models outputs.

| Input                | Shape                     | Description  |
|----------------------|---------------------------|--|
| <code>img</code>     | $150 \times 150 \times 3$ | RGB image from the object in JPEG format                             |
| <code>fits</code>    | $61 \times 61 \times 5$   | FITS data subset around the object across the u, g, r, i and z bands |
| <code>spectra</code> | $3225 \times 1$           | full best fit spectra data between wavelengths 4000 Å and 9000 Å     |
| <code>ssel</code>    | $1225 \times 1$           | best fit spectra data for selected intervals of wavelengths          |
| <code>bands</code>   | $5 \times 1$              | photometric magnitudes for the u, g, r, i and z bands                |
| <code>wise</code>    | $4 \times 1$              | photometric values for the W1, W2, W3 and W4 WISE magnitudes         |

Table 4.4: Summary of the data in the dataset used as input features for the devised models.

| Output                | Type                  | Description   |
|-----------------------|-----------------------|---|
| <code>redshift</code> | <i>regression</i>     | final redshift from SDSS data ‘z’   |
| <code>subclass</code> | <i>classification</i> | subset of sub-class from SDSS data for the galaxy objects <code>subclass</code> |
| <code>smass</code>    | <i>regression</i>     | stellar mass extracted from the eBOSS Firefly catalogue                         |
| <code>gz2c</code>     | <i>classification</i> | simplified version of the GZ2 classification                                    |

Table 4.5: Summary of the data in the dataset used as output for the devised models.

For the upcoming discussion of the models architecture and in order to be closer to an actual implementation, a more formal definition (specification) of the data types used by the models follows. This allows a more accurate upcoming discussion of models and data flows, using a more formal language, and helps with the actual implementation.

**data** `Img = Img { values :: NDArray[height, width, bands] }`

where, the `Img` data type is defined as a three-dimensional array with dimensions `height = 150`, `width = 150` and `bands = 3`, inline with the RGB images described in the SDSS GS dataset that have the same size,  $150 \times 150$  pixels, and 3 colour channels (Red, Green and Blue). Note that the order of dimensions between the height and width is somewhat arbitrary, because all the *images*-like data in the dataset have a ratio of 1, i.e. have the same width

and height, the reason to select this particular ordering is to be consistent with the convention that the frameworks is used for the implementation of the models, discussed later in this chapter, which is: samples, height, width and channels. The data type for FITS data is defined following a similar reasoning:

```
data Fits = Fits { values :: NDArray[height, width, bands] }
```

where, the *Fits* data type is defined as a three-dimensional array with dimensions *height* = 61, *width* = 61, and *bands* = 5, inline with the FITS data available in the described dataset where the cut of the data around the object has a size of 61 × 61 pixels, and the data has 5 bands. The data type for the full spectra data is defined:

```
data Spectra = Spectra { values :: NDArray[height, width] }
```

where, the *Spectra* data type is defined as a two-dimensional array (or a vector) with dimensions *height* = 3225 and *width* = 1. The spectra data available in the dataset, in a tabular format, has more dimensions but since only the *BestFit* flux is considered this is just a vector of values, including the height and width is just a consistency detail. The data type for the spectra selected bands follows exactly the same logic:

```
data Ssel = Ssel { values :: NDArray[height, width] }
```

where, the *Ssel* data type is defined as a two-dimensional array (or a vector) with dimensions *height* = 1225 and *width* = 1. The data type for the bands data is defined as:

```
data Bands = Bands { values :: NDArray[height, width] }
```

where, the *Bands* data type is defined as a two-dimensional array (or a vector) with dimensions *height* = 5 and *width* = 1. The bands data available in the dataset, in a tabular format, corresponds to the values in the specific bands as described in the catalogue. The data type for the WISE data is defined as:

```
data Wise = Wise { values :: NDArray[height, width] }
```

where, the *Wise* data type is defined as a two-dimensional array (or a vector) with dimensions *height* = 4 and *width* = 1. The WISE data available in the dataset, in a tabular format, corresponds to the values in the specific bands as described in the catalogue. Finally we can define a data type to represent an arbitrary input, i.e. an object representing an instance of any of the defined input types:

**type** *Input* = *Img* | *Fits* | *Spectra* | *Ssel* | *Bands* | *Wise*

And the data types for the outputs are also defined for the sake of completeness, the redshift is defined as an alias for the native type *Float*:

**type** *Redshift* = *Float*

and the same for the stellar mass:

**type** *Smass* = *Float*

The data type for the sub-class is defined as an alternative between all the classes available:

**type** *Subclass* = *AGN* | *Broadline* | *Starburst* | *Starforming*

where, each class is simply defined as string with the name of the class. The same for the GZ2 simplified class labels:

**type** *Gz2c* = *A* | *Ec* | *Ei* | *Er* | *SBa* | *SBaR* | *SBb* | *SBbR* | ... | *Ser*

For a complete list of the GZ2 simplified classes and descriptions refer to Appendix D. And again we can define a data type to represent an arbitrary output, i.e. an object representing an instance of any of the defined output types:

**type** *Output* = *Redshift* | *Subclass* | *Smass* | *Gz2c*

Given the aforementioned data type definitions, where all the data types are represented using multi-dimensional arrays, note that all dimensions are consistent and follow the pattern [ *height*, *width*, *bands* ], this is relevant because when instances of these data types, i.e. real data, are feed to neural networks this is the expected semantics of the values. By choosing these definition there is no ambiguity later to what the data represents and everything is readily consistent.

## 4.4 Models Architectures

The generic definition of the specification of a model in the context of this work, is a function that maps a list of inputs to a list of outputs, following the collection of data types defined in the previous section this is captured in the following function signature definition:

$$model :: [Input] \rightarrow [Output]$$

A more specific definition for a model, for example, that infers the redshift from an RGB image can be described as:

$$i2r :: Img \rightarrow Redshift$$

where,  $i2r$  is a function that maps an image, a three dimensional array with an height and width of 150, and depth of 3 values as defined by the *Img* data type, to the redshift a real number (*Float*). The intended inputs and outputs of the model are well defined, and there is no ambiguity.

To explore and experiment with these models an actual implementation is required. In the context of this work models are built using a DL approach, i.e. they are defined as a combination of layers and networks described in Chapter 3 and their inputs and outputs match exactly the data types defined in the previous section. Table 4.6 summarizes the complete list of devised models with a single input and a single output, following the inputs described in Table 4.4 and the outputs described in Table 4.5, including the type of model, regression or classification, and a small description on each model intended goal. Table 4.7 summarizes the complete list of devised models with multiple inputs and outputs, following the inputs described in Table 4.4 and the outputs described in Table 4.5.

All the models are defined as a combination of layers, The next section describes in detail all the DL models architectures devised, implemented using Keras ([Ketkar, 2017](#)), a open-source framework providing an interface for artificial neural networks libraries. Appendix E illustrates the use of the Keras API to create a simple network example.

#### 4.4.1 Predicting Redshift

To predict the redshift of an object a collection of models is available, namely one for each available individual input: image, FITS, spectra, selected spectra, bands and WISE data; and some exploring multi inputs and outputs.

In the case where the single input are images or FITS data the model first layers apply convolutional operations, followed by max polling layers, to capture small details from the data and build up larger representations. After a flatten operation a sequence of fully connected layers using a ReLU activation function connects to the output layer, a single node – the redshift. Figure 4.5 illustrates the network composition used for the models to predict the redshift from the image (left) and FITS data (middle).



| Model | Input   | Output   | Type           | Description   |
|-------|---------|----------|----------------|---|
| i2r   | img     | redshift | regression     | infer redshift from RGB image                         |
| f2r   | fits    | redshift | regression     | infer redshift from FITS data                         |
| s2r   | spectra | redshift | regression     | infer redshift from spectra data                      |
| ss2r  | ssel    | redshift | regression     | infer redshift from selected spectra data             |
| b2r   | bands   | redshift | regression     | infer redshift from bands data                        |
| w2r   | wise    | redshift | regression     | infer redshift from WISE data                         |
| i2sm  | img     | smass    | regression     | infer stellar mass from RGB image                     |
| f2sm  | fits    | smass    | regression     | infer stellar mass from FITS data                     |
| s2sm  | spectra | smass    | regression     | infer stellar mass from spectra data                  |
| ss2sm | ssel    | smass    | regression     | infer stellar mass from selected spectra data         |
| b2sm  | bands   | smass    | regression     | infer stellar mass from bands data                    |
| w2sm  | wise    | smass    | regression     | infer stellar mass from WISE data                     |
| i2s   | img     | subclass | classification | infer sub-class from RGB image                        |
| f2s   | fits    | subclass | classification | infer sub-class from FITS data                        |
| s2s   | spectra | subclass | classification | infer sub-class from spectra data                     |
| ss2s  | ssel    | subclass | classification | infer sub-class from selected spectra data            |
| b2s   | bands   | subclass | classification | infer sub-class from bands data                       |
| w2s   | wise    | subclass | classification | infer sub-class from WISE data                        |
| i2g   | img     | gz2c     | classification | infer GZ2 simplified class from RGB image             |
| f2g   | fits    | gz2c     | classification | infer GZ2 simplified class from FITS data             |
| s2g   | spectra | gz2c     | classification | infer GZ2 simplified class from spectra data          |
| ss2g  | ssel    | gz2c     | classification | infer GZ2 simplified class from selected spectra data |
| b2g   | bands   | gz2c     | classification | infer GZ2 simplified class from bands data            |
| w2g   | wise    | gz2c     | classification | infer GZ2 simplified class from WISE data             |

Table 4.6: Summary of the complete list of models with single input and single output.

| Model         | Inputs                                | Outputs                         |
|---------------|---------------------------------------|---------------------------------|
| fSbW2rSM      | fits, spectra, bands, wise            | redshift, smass                 |
| fSbW2sG       | fits, spectra, bands, wise            | subclass, gz2c                  |
| iFsSSbW2r     | img, fits, spectra, ssel, bands, wise | redshift                        |
| iFsSSbW2sm    | img, fits, spectra, ssel, bands, wise | smass                           |
| iFsSSbW2s     | img, fits, spectra, ssel, bands, wise | subclass                        |
| iFsSSbW2g     | img, fits, spectra, ssel, bands, wise | gz2c                            |
| iFsSSbW2rSMsG | img, fits, spectra, ssel, bands, wise | redshift, smass, subclass, gz2c |

Table 4.7: Summary of the complete list of models with multi inputs and outputs.

In the case where the input is the spectral data there are two models available with slightly two different approaches. When the input is the full spectral data, a vector of numbers, the first layer included in the model is a normalization layer, followed by convolutional operations layers and a max pooling layer, mainly to try to capture details in the spectra that may shift positions in the input vector. The next layers include a flatten operation and a sequence of fully connected layers using a ReLU activation function, ending in a single node – the redshift. When the input is only the spectral data for the list of selected bands since the input is no longer completely sequential, i.e. there are gaps in the data – the bands not selected to be included – there is less point on looking for moving patterns that could appear in different positions of the vector, so there are no convolutional operations. After the normalizing layers a sequence of fully connected layers is used, ending in an output node – the redshift. Figure 4.5 (right) illustrates the network composition used for the model to predict the redshift from the full spectral data, and Figure 4.6 (left) illustrates the model for predicting the redshift from the selected spectral bands.

In the case where the input are the bands and WISE data, the models start with a normalizing layer to scale the values, followed by a sequence of fully connected layers, ending in a single output node – the redshift. Figure 4.6 illustrates the model for predicting the redshift from the bands data (middle), and from the WISE data (right).

Following the aforementioned notation and data types definitions, Table 4.8 summarizes the signatures for the functions that map the individual inputs representing the available single input single output models to predict the redshift.

| Models  |
|---|
| $i2r :: \text{Img} \rightarrow \text{Redshift}$     |
| $f2r :: \text{Fits} \rightarrow \text{Redshift}$    |
| $s2r :: \text{Spectra} \rightarrow \text{Redshift}$ |
| $ss2r :: \text{Ssel} \rightarrow \text{Redshift}$   |
| $b2r :: \text{Bands} \rightarrow \text{Redshift}$   |
| $w2r :: \text{Wise} \rightarrow \text{Redshift}$    |

Table 4.8: Summary of functions representing the single input single output models for predicting the redshift.

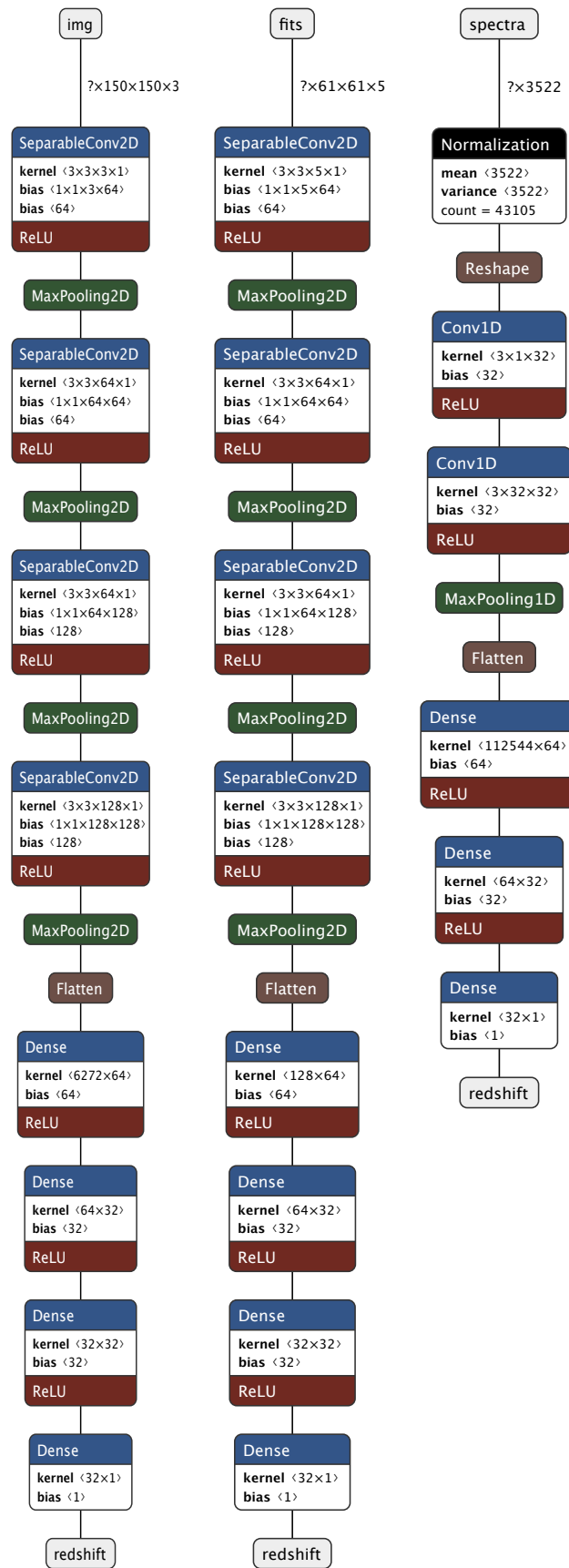


Figure 4.5: Predicting redshift from RGB images (left), FITS data (middle) and spectra data (right) model architectures using convolutional operations.

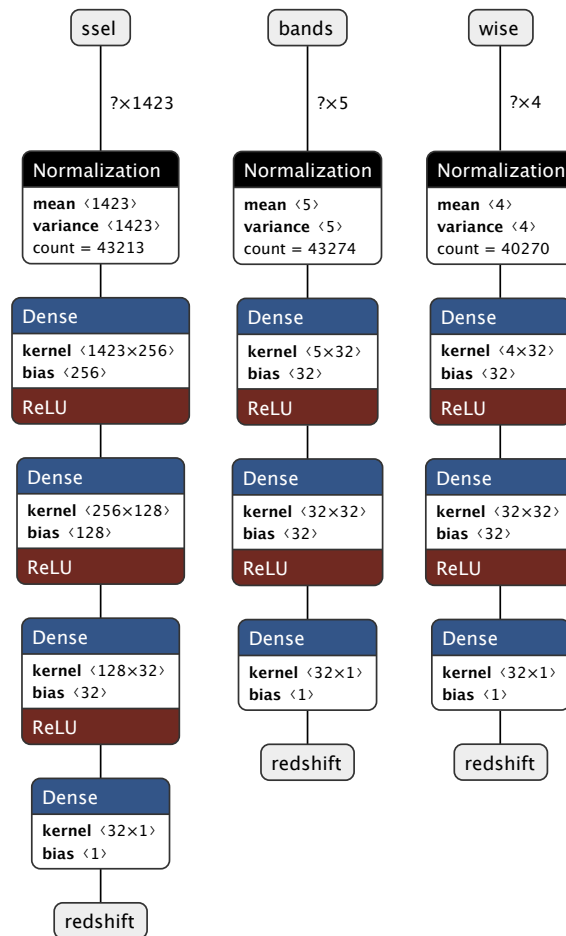


Figure 4.6: Predicting redshift from selected spectra bands data (left) bands data (middle) and WISE data (right) model architectures using only fully connected layers.

#### 4.4.2 Predicting Stellar Mass

To predict the stellar mass of an object another collection of models is available, namely one for each available individual input: image, FITS, spectra, selected spectra, bands and WISE data; and some exploring multi inputs and outputs.

In the case where the single input are images or FITS data the model initial layers apply convolutional operations, followed by max pooling layers, as before to capture small details from the data and build up larger representations. After a flatten operation a sequence of fully connected layers using a ReLU activation function connects to the output layer, a single node – the stellar mass. Figure 4.7 illustrates the network composition used for the models to predict the the stellar mass from the image (left) and FITS data (middle).

In the case where the input is the spectral data there are two models available with slightly two different approaches. Following a similar approach as to predict the redshift, when the input is the full spectral data, a vector of numbers, the first layer included in

the model is a normalization layer, followed by convolutional operations layers and a max pooling layer, mainly to try to capture details in the spectra that may shift positions in the input vector. The next layers include a flatten operation and a sequence of fully connected layers using a ReLU activation function, ending in a single node – the stellar mass. When the input is only the spectral data for the list of selected bands since the input is no longer completely sequential, i.e. there are gaps in the data as discussed before, there is less point in looking for moving patterns that could appear in different positions of the vector, so there are no convolutional operations. After the normalizing layers a sequence of fully connected layers is used, ending in an output node – the stellar mass. Figure 4.7 (right) illustrates the network composition used for the model to predict the stellar mass from the full spectral data, and Figure 4.8 (left) illustrates the model for predicting the stellar mass from the selected spectral bands.

In the case where the input is the bands and WISE data the models start with a normalizing layer to scale the values, followed by a sequence of fully connected layers, ending in a single output node – the stellar mass. Figure 4.8 illustrates the model for predicting the stellar mass from the bands data (middle), and from the WISE data (right).

Following the aforementioned definitions, Table 4.9 summarizes the signatures for the functions that map the individual inputs representing the available single input single output models to predict the stellar mass.

| Models                              |
|-------------------------------------|
| $i2sm :: Img \rightarrow Smass$     |
| $f2sm :: Fits \rightarrow Smass$    |
| $s2sm :: Spectra \rightarrow Smass$ |
| $ss2sm :: Ssel \rightarrow Smass$   |
| $b2sm :: Bands \rightarrow Smass$   |
| $w2sm :: Wise \rightarrow Smass$    |

Table 4.9: Summary of functions representing the single input single output models for predicting the stellar mass.

### 4.4.3 Predicting Sub-class

To predict the sub-class of an object another collection of models is available, namely one for each available individual input: image, FITS, spectra, selected spectra, bands and WISE data; and some exploring multi inputs and outputs.

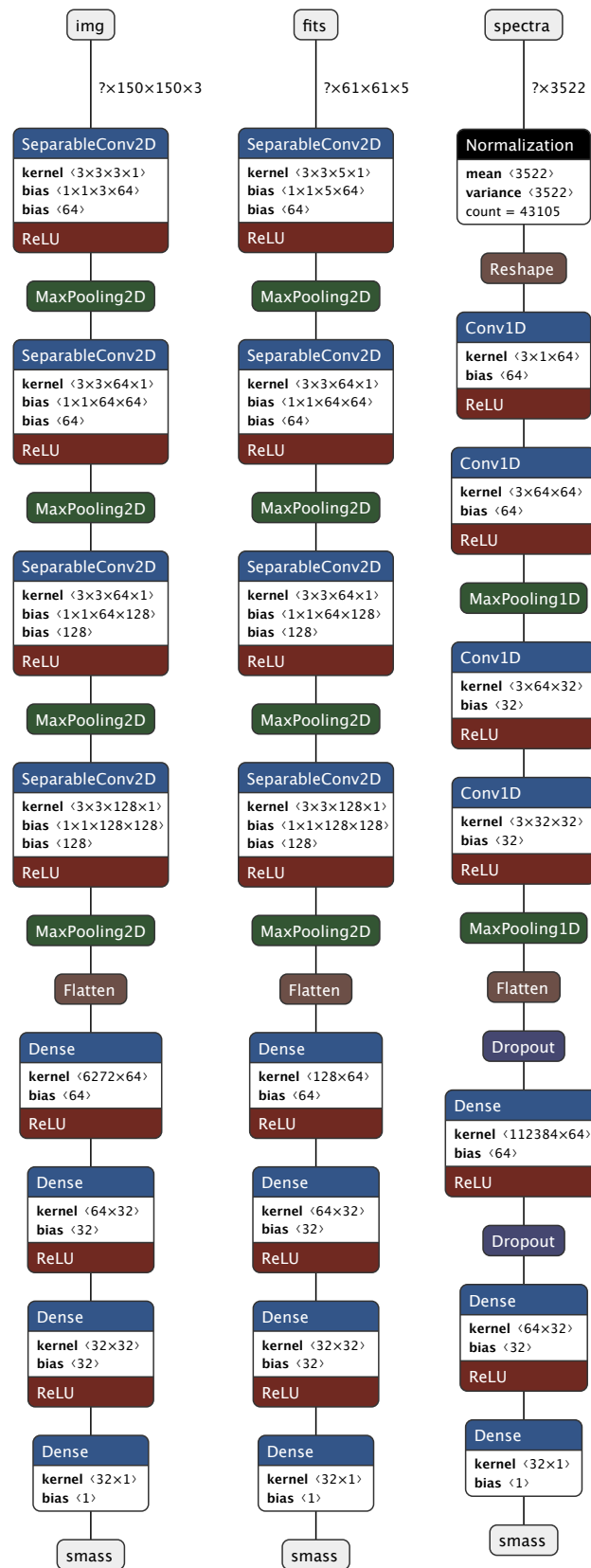


Figure 4.7: Predicting stellar mass from RGB images (left), FITS data (middle) and spectra data (right) model architectures using convolutional operations.

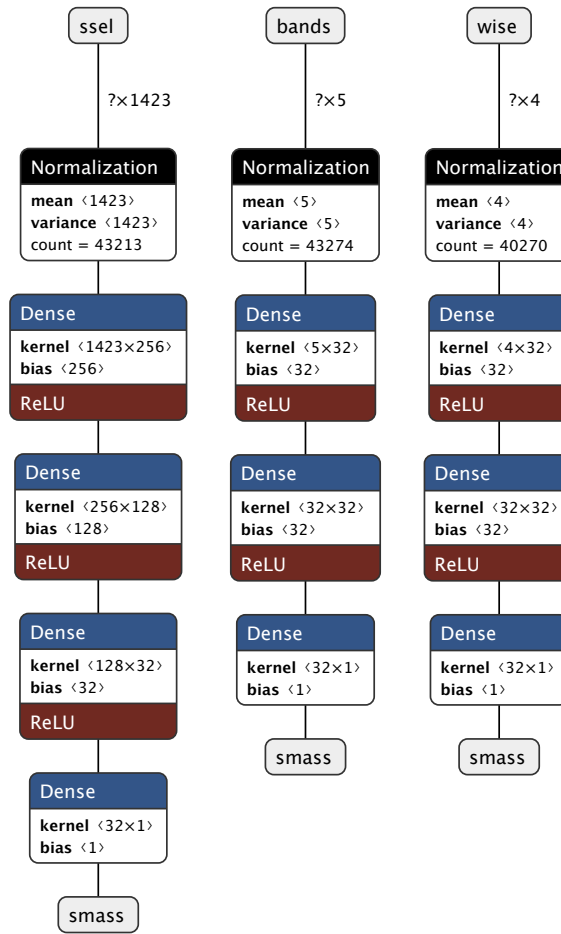


Figure 4.8: Predicting stellar mass from selected spectra bands data (left) bands data (middle) and WISE data (right) model architectures using only fully connected layers.

In the case where the single input are images or FITS data the model initial layers apply convolutional operations, followed by max pooling layers, to capture small details from the data and build up larger representations. After a flatten operation a sequence of fully connected layers using a ReLU activation function connects to the output layer, fully connected layer with the number of nodes equal to the number of classes using a *softmax* activation function. The final layer represents the probability distribution of being classified to one specific class – the sub-class. Figure 4.9 illustrates the network composition used for the models to predict the sub-class from the image (left) and FITS data (middle).

In the case where the input is spectral data, once again there are two models available with slightly two different approaches. Following a similar approach as to predict the redshift and the stellar mass, when the input is the full spectral data, a vector of numbers, the first layer included in the model is a normalization layer, followed by convolutional operations layers and a max pooling layer, mainly to try to capture details in the spectra that

may shift positions in the input vector. The next layers include a flatten operation and a sequence of fully connected layers using a ReLU activation function, ending in a fully connected layer with the number of nodes equal to the number of classes using a *softmax* activation function. As before, the final layer represents the probability distribution of being classified to one specific class – the sub-class. When the input is only the spectral data for the list of selected bands since the input is no longer completely sequential, i.e. there are gaps in the data – the bands not selected to be included – there is no point on looking for moving patterns that could appear in different positions of the vector as before, so there are no convolutional operations. After the normalizing layers a sequence of fully connected layers is used, ending in a fully connected layer with the number of nodes equal to the number of classes using a *softmax* activation function. Figure 4.9 (right) illustrates the network composition used for the model to predict the sub-class from the full spectral data, and Figure 4.10 (left) illustrates the model for predicting the sub-class from the selected spectral bands.

In the case where the input is the bands and WISE data the models start with a normalizing layer to scale the values, followed by a sequence of fully connected layers, ending in a fully connected layer with the number of nodes equal to the number of classes using a *softmax* activation function. As before, the final layer represents the probability distribution of being classified to one specific class – the sub-class. Figure 4.10 illustrates the model for predicting the sub-class from the bands data (middle), and from the WISE data (right).

Following previous definitions, Table 4.10 summarizes the signatures for the functions that map the individual inputs representing the available single input single output models to predict the sub-class.

| Models                                |
|---------------------------------------|
| $i2s :: Img \rightarrow Subclass$     |
| $f2s :: Fits \rightarrow Subclass$    |
| $s2s :: Spectra \rightarrow Subclass$ |
| $ss2s :: Ssel \rightarrow Subclass$   |
| $b2s :: Bands \rightarrow Subclass$   |
| $w2s :: Wise \rightarrow Subclass$    |

Table 4.10: Summary of functions representing the single input single output models for predicting the sub-class.



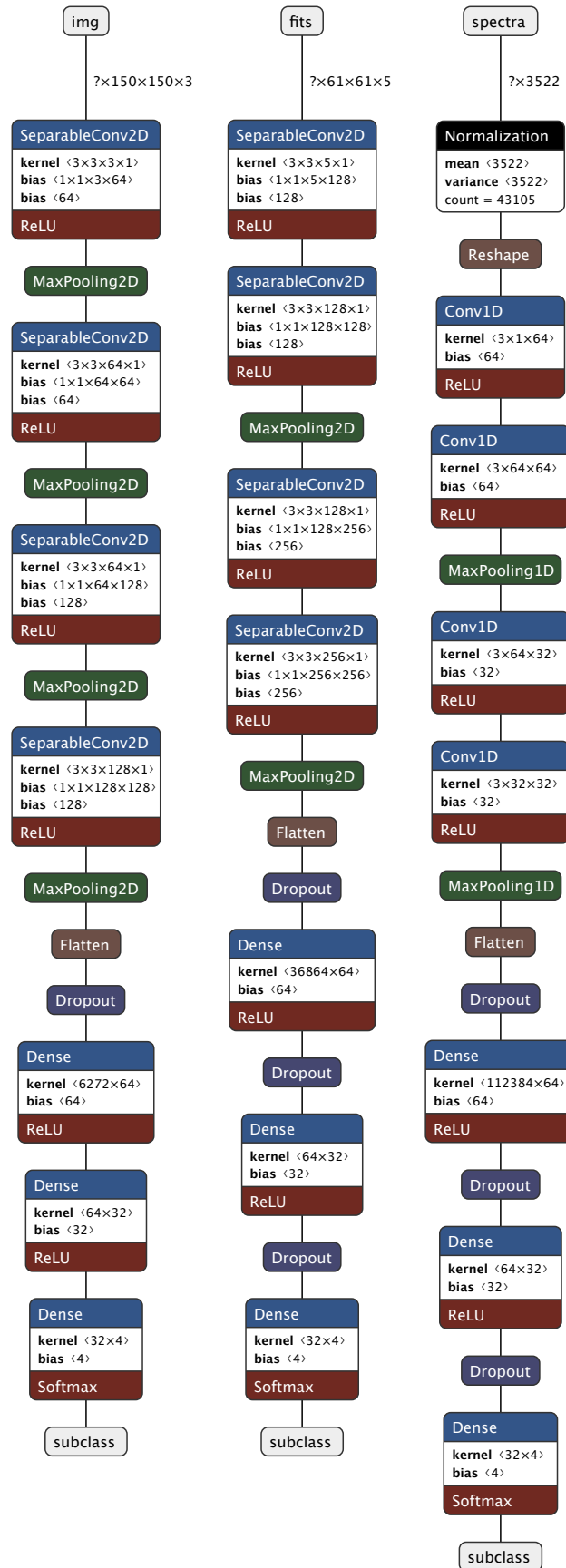


Figure 4.9: Predicting sub-class from RGB images (left), FITS data (middle) and spectra data (right) model architectures using convolutional operations.

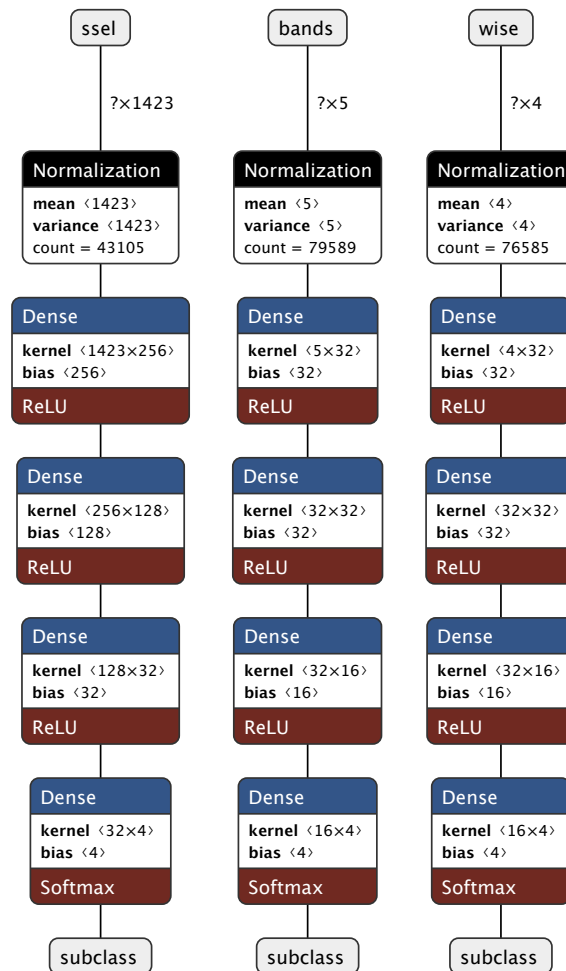


Figure 4.10: Predicting sub-class from selected spectra bands data (left) bands data (middle) and WISE data (right) model architectures using only fully connected layers.

#### 4.4.4 Predicting GZ2 Simplified Class

To predict the GZ2 simplified class of an object another collection of models is available, namely one for each available individual input: image, FITS, spectra, selected spectra, bands and WISE data; and some exploring multi inputs and outputs.

In the case where the single input are images or FITS data the model initial layers apply convolutional operations, followed by max polling layers, to capture small details from the data and build up larger representations. After a flatten operation a sequence of fully connected layers using a ReLU activation function connects to the output layer, fully connected layer with the number of nodes equal to the number of classes using a *softmax* activation function. The final layer represents the probability distribution of being classified to one specific class – the GZ2 simplified class. Figure 4.11 illustrates the network composition used for the models to predict the GZ2 simplified class from the image (left)

and FITS data (middle).

In the case where the input is the spectral data there are two models available with slightly two different approaches. Following a similar approach as to predict the sub-class, when the input is the full spectral data, a vector of numbers, the first layer included in the model is a normalization layer, followed by convolutional operations layers and a max pooling layer, mainly to try to capture details in the spectra that may shift positions in the input vector. The next layers include a flatten operation and a sequence of fully connected layers using a ReLU activation function, ending in a fully connected layer with the number of nodes equal to the number of classes using a *softmax* activation function. As before, the final layer represents the probability distribution of being classified to one specific class – the GZ2 simplified class. When the input is only the spectral data for the list of selected bands since the input is no longer completely sequential, again there is less point on looking for moving patterns that could appear in different positions of the vector, so there are no convolutional operations. After the normalizing layers a sequence of fully connected layers is used, ending in a fully connected layer with the number of nodes equal to the number of classes using a *softmax* activation function. Figure 4.11 (right) illustrates the network composition used for the model to predict the GZ2 simplified class from the full spectral data, and Figure 4.12 (left) illustrates the model for predicting the GZ2 simplified class from the selected spectral bands.

In the case where the input is the bands and WISE data the models start with a normalizing layer to scale the values, followed by a sequence of fully connected layers, ending in a fully connected layer with the number of nodes equal to the number of classes using a *softmax* activation function. As before, the final layer represents the probability distribution of being classified to one specific class – the GZ2 simplified class. Figure 4.12 illustrates the model for predicting the GZ2 simplified class from the bands data (middle), and from the WISE data (right).

Following previous definitions, Table 4.11 summarizes the signatures for the functions that map the individual inputs representing the available single input single output models to predict the GZ2 simplified class.

#### 4.4.5 Multi Input/Output Models

The previous sections discuss the single input, single output models, one available for each combination of available input data and outputs. This section discusses first the

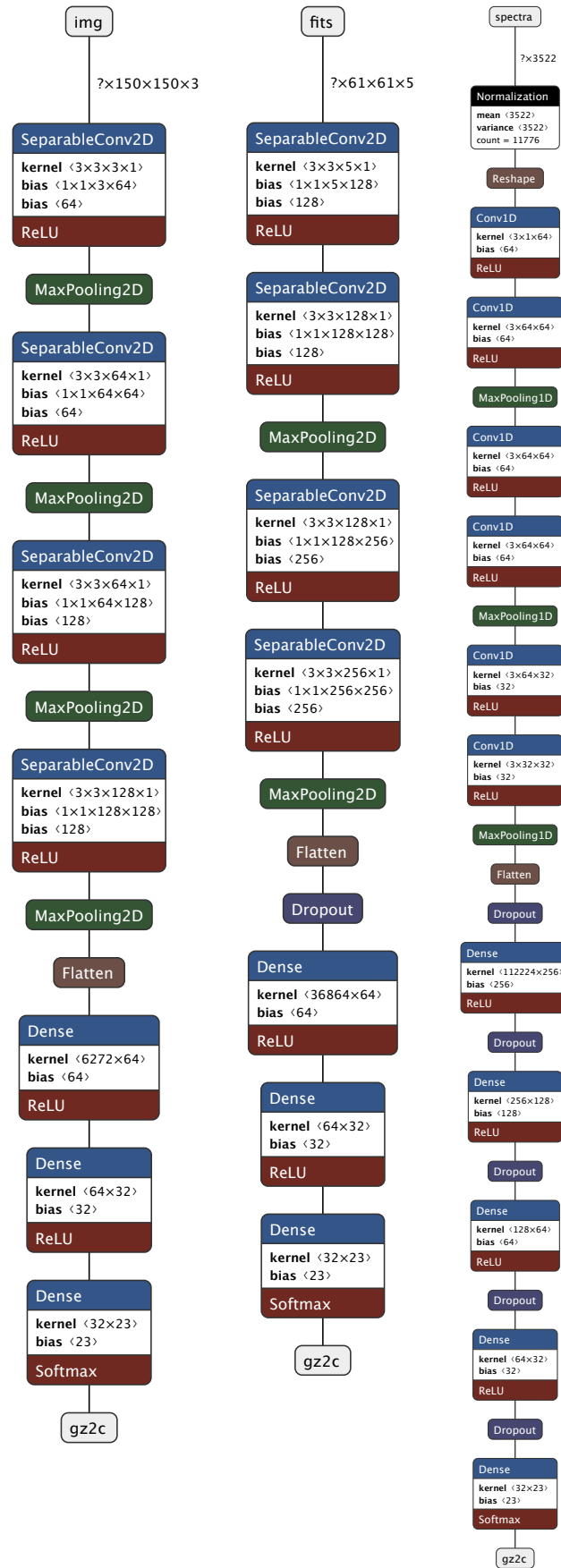


Figure 4.11: Computing GZ2 simplified class from RGB images (left), FITS data (middle) and spectra data (right) model architectures using convolutional operations.

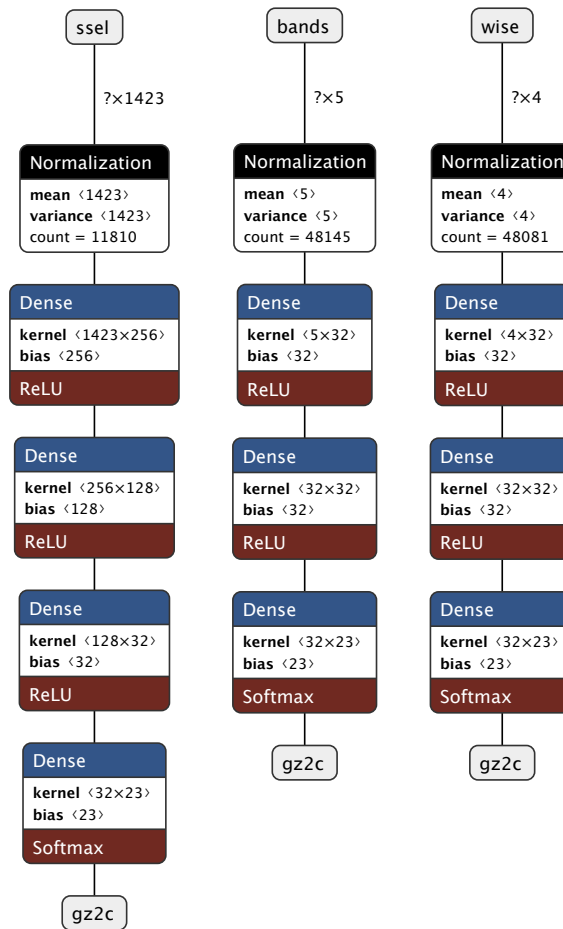


Figure 4.12: Computing GZ2 simplified class from selected spectra bands data (left) bands data (middle) and WISE data (right) model architectures using only fully connected layers.

---

#### Models

---

$i2g :: Img \rightarrow Gz2c$

$f2g :: Fits \rightarrow Gz2c$

$s2g :: Spectra \rightarrow Gz2c$

$ss2g :: Ssel \rightarrow Gz2c$

$b2g :: Bands \rightarrow Gz2c$

$w2g :: Wise \rightarrow Gz2c$

---

Table 4.11: Summary of functions representing the single input single output models for predicting the GZ2 simplified class.

devised models that combine all the available inputs to predict a single output, either for regression or classification. And also some experiments where some combinations of available inputs are used to predict different combinations of outputs. The main intuition for creating these models is that data from different sources can potentially provide more information to predict the intended output. Also the exploration of such models may infer unexpected patterns between inputs and outputs that can give hints on new links between the available measurements and physical phenomena underlying the models outputs characteristics.

The following multi input, single output models, illustrate the networks for the models that take as input all the available data: images, FITS, spectra, bands and WISE, and predict each one of the available outputs individually: redshift, stellar mass, sub-class and GZ2 simplified class, by composing the single input, single output models described in the previous section.

Figure 4.13 illustrates the model for predicting the redshift, where the output of all the model networks for predicting the redshift described in the previous section are combined together using a concatenation layer, this conveys a single vector representing all the input data, followed by a sequence of fully connected layers, ending in a single node output – the redshift. Figure 4.14 illustrates the model for predicting the stellar mass, following a similar approach.

Figure 4.15 illustrates the model for predicting the sub-class, where the output of all the model networks for predicting the sub-class described in the previous section are combined together using a concatenation layer, this conveys a single vector representing all the input data, followed by a sequence of fully connected layers, ending in an output layer with a *softmax* activation – the sub-class. Figure 4.16 illustrates the model for predicting the GZ2 simplified class, following a similar strategy.

Besides multi input with single output models some more complex networks with multi input multi output models are also devised. The intuition behind these models is exploring the relations between the inputs and the outputs at the same time, looking for new synergies. This entails that during the training process, when models weights space is explored, there are several output constrains being captured by the same underlying patterns, at an higher level of abstraction the physical phenomena that drive different output measurements should be the same. Figure 4.17 illustrates the networks for implementing models that use all the available inputs: RGB images, FITS, spectral, bands and WISE data to

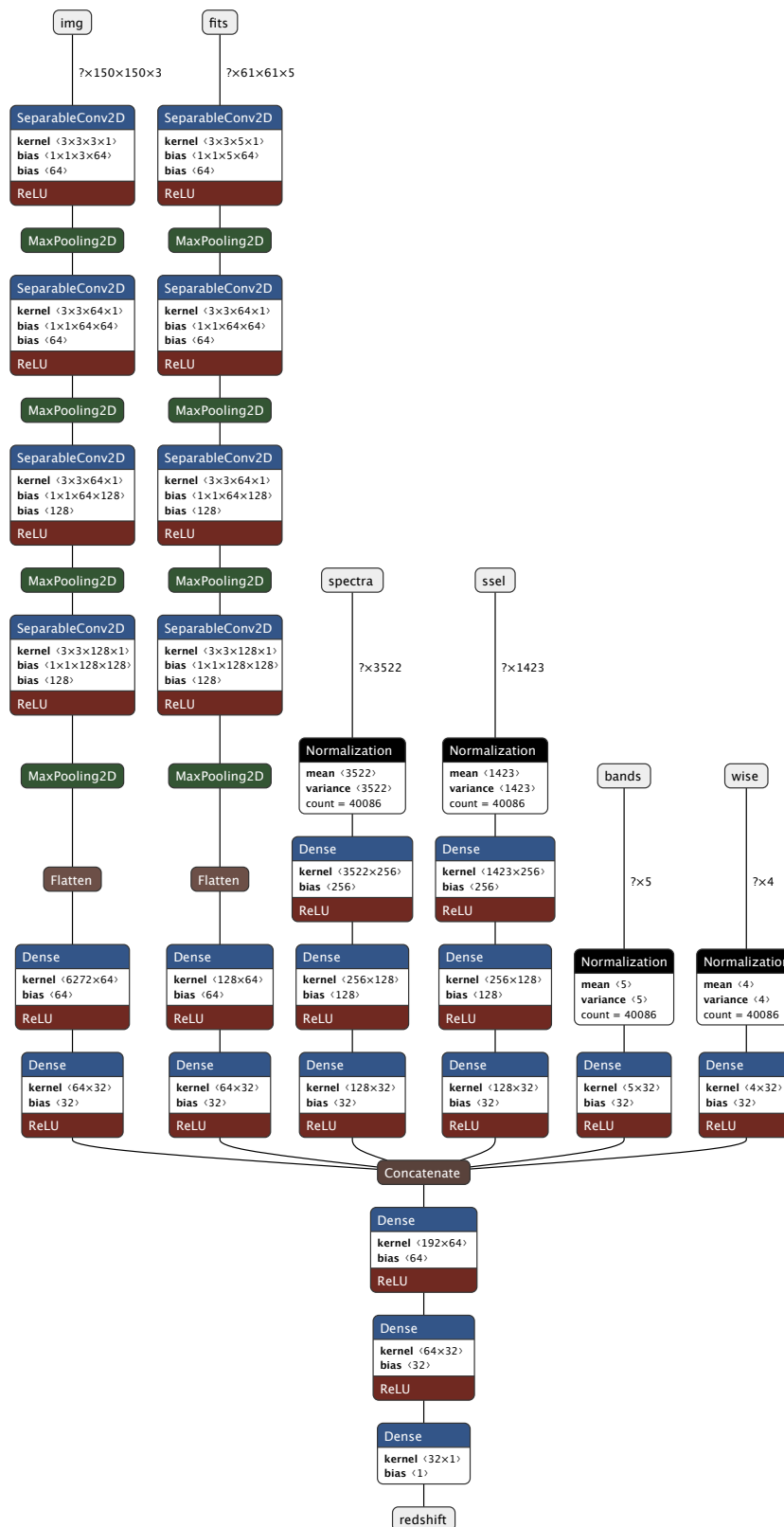


Figure 4.13: Predicting the redshift from the RGB images, FITS, spectral, bands and WISE data using a combination of networks.

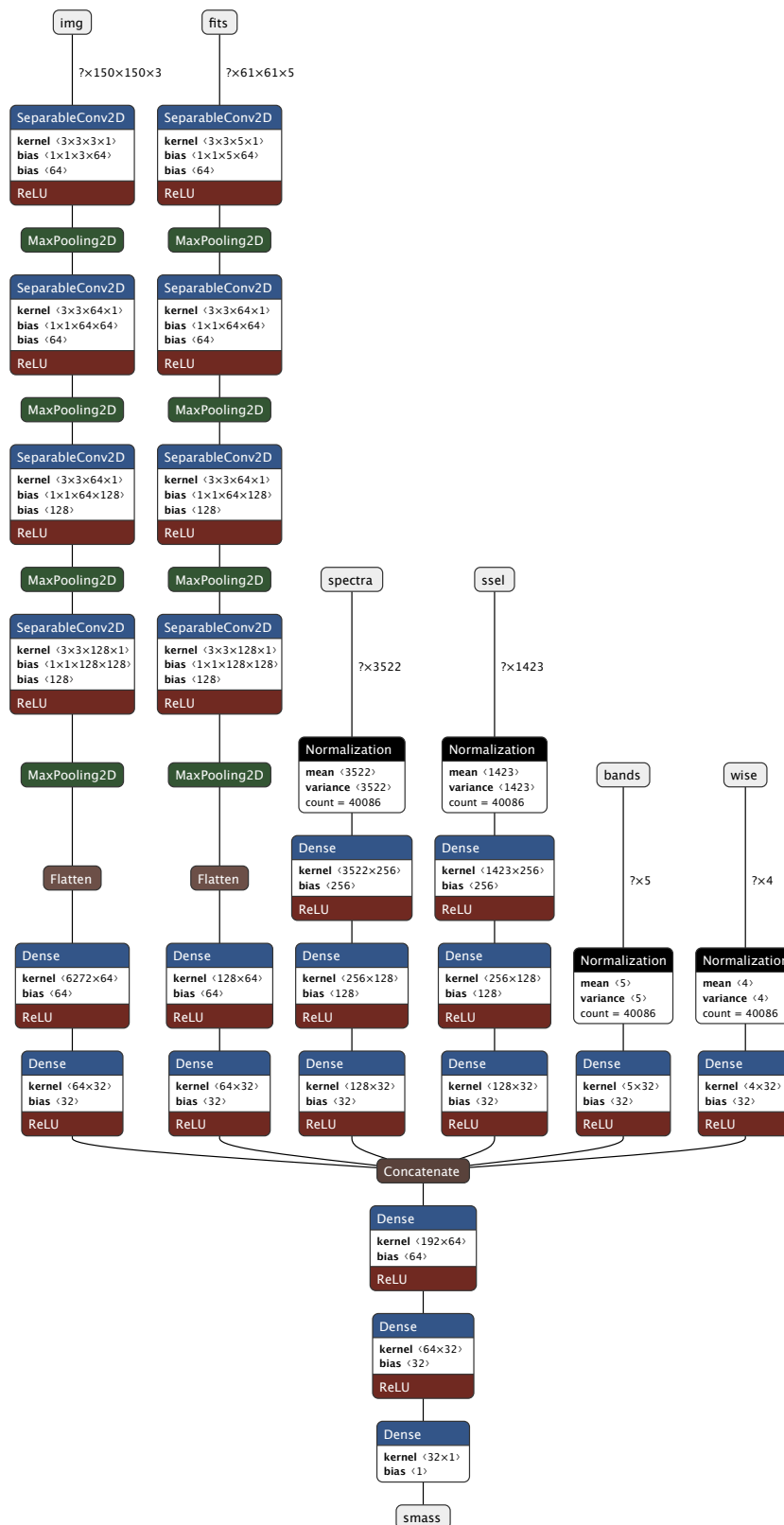


Figure 4.14: Predicting the stellar mass from the RGB images, FITS, spectral, bands and WISE data using a combination of networks.



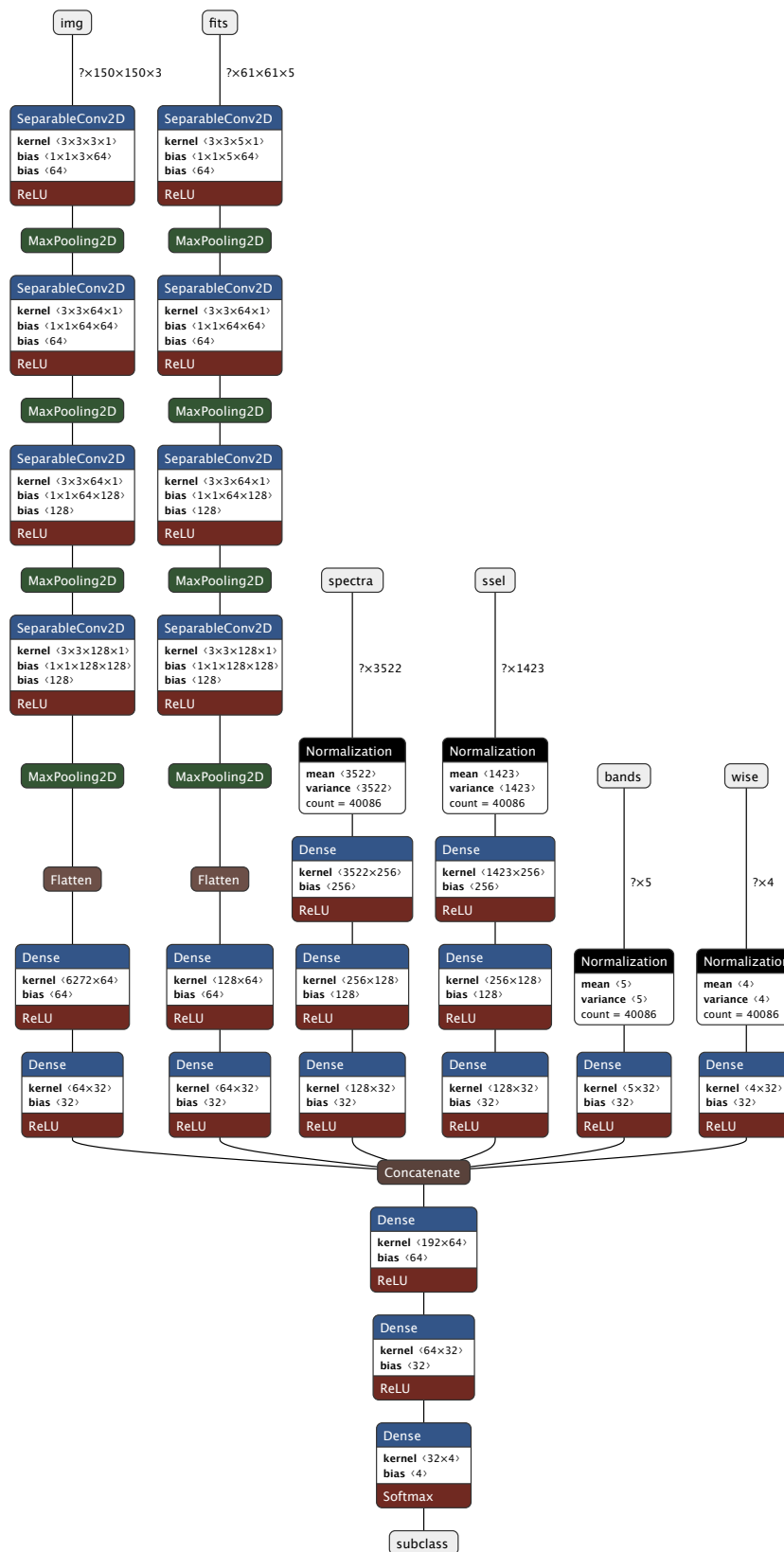


Figure 4.15: Predicting the sub-class from from the RGB images, FITS, spectral, bands and WISE data using a combination of networks.

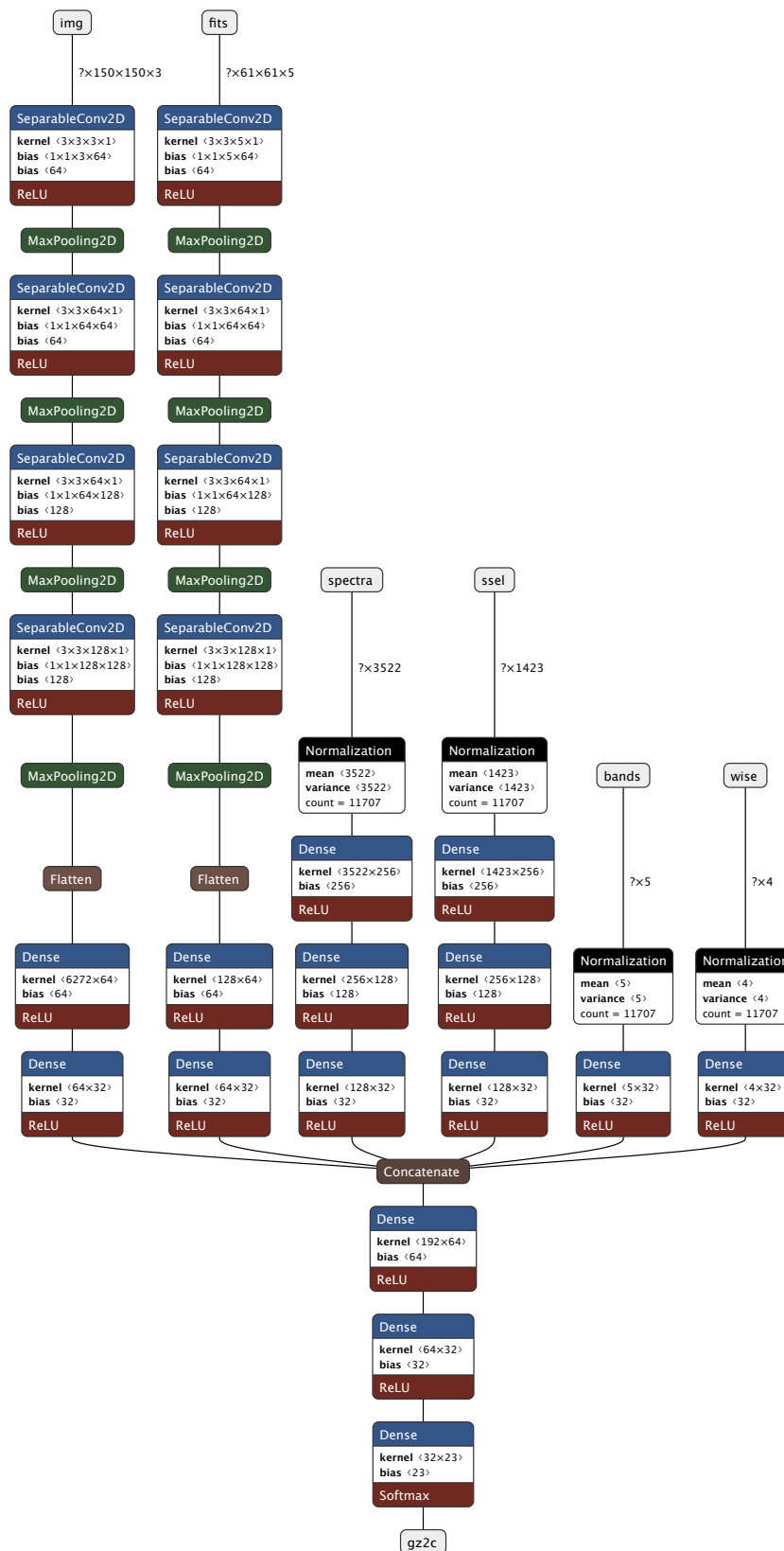


Figure 4.16: Predicting the GZ2 simplified class from the RGB images, FITS, spectral, bands and WISE data using a combination of networks.

predict simultaneously the redshift and the stellar mass (left) and the sub-class and GZ2 simplified class (right).

And finally the model network illustrated in Figure 4.18 uses all the inputs available by combining all the required sub-networks, and yields all the output measurements. The approach used is similar to the ones before, all the input data is combined into a single vector, followed by a sequence of fully connected layers, but this time the output layer is comprised of one node to output the redshift value, another node to output the stellar mass value, a set of nodes using a *softmax* activation layer to output the sub-class, and another set of nodes, also using a *softmax* activation function, to output the GZ2 simplified class. The underlying patterns mapping all the inputs to all the outputs should be captured in a single function.



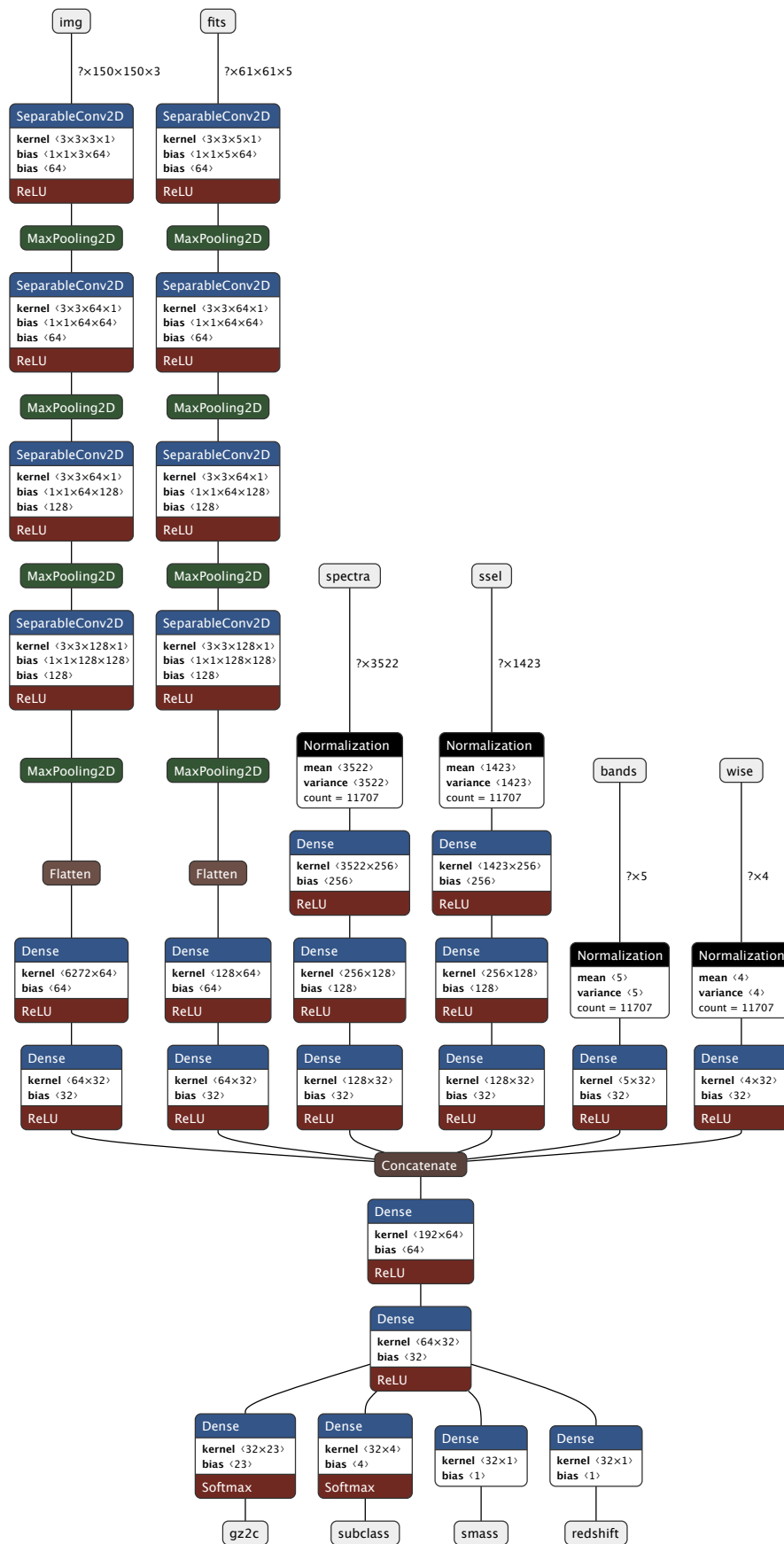


Figure 4.18: Predicting the redshift, the stellar mass, the sub-class and the GZ2 simplified class from the RGB images, FITS, spectral, bands and WISE data using a combination of networks.

## 4.5 Training & Evaluation

Given the collection of models described in the previous section, the next step is to actually fit the models to real data (train) and perform some kind of evaluation to assess the models performance. As discussed in Section 3 there is a set of hyper-parameters that can be fine-tuned to potentially improve each model. Some of these parameters, besides the architecture of the model, include: the optimizer, the batch size and the loss function. One common way to explore some options for these parameters that yield good results is to test different combinations using the same dataset and compare the performance results. Also note that sometimes the output of the loss functions by itself is not directly comparable due to the mathematical definition of the function, for example the Mean Squared Error (MSE) *squares* the difference between the output and *true* values.

Table 4.12 illustrates the hyper-parameters exploration for the single input single output models discussed in the previous section for predicting the redshift. Each row in the table represents a full training cycle run for the given number of epochs and changing the batch size, the optimizer and the loss function. For each cycle the available data is split in the three sets: training, validation and testing. The validation dataset is used to monitor model evolution during training, and the score in the table indicates the model performance on the test dataset, that is never seen during training. Since these are regression models, i.e. models that output a real value, the two loss functions that are tested include: MSE and Mean Absolute Error (MAE); the selected optimizers to test are *RMSProp* and *Adam* with the default learning rates; and the batch size is tested between 32 and 64. By looking at the score (lower is better in this case) the hyper-parameters that achieve the best performance can be selected. For example, given the results for the `i2r` model, the model achieves the best result using a batch size of 32, a *RMSProp* optimizer and a MSE loss function.

Table 4.13 illustrates the hyper-parameters exploration for the single input single output models discussed in the previous section for predicting the stellar mass. Following the same approach as before, each row in the table represents a full training cycle run for the given number of epochs and changing the batch size, the optimizer and the loss function. For each cycle the available data is split in the three sets: training, validation and testing. The validation dataset is used to monitor model evolution during training, and the score in the table indicates the model performance on the test dataset, that is never seen during training. Since these are regression models, the two loss functions that are tested include: MSE and MAE; the selected optimizers to test are *RMSProp* and *Adam* with the default

| Model | Input   | Output   | Epochs | Batch Size | Optimizer | Loss | Score    |
|-------|---------|----------|--------|------------|-----------|------|----------|
| i2r   | img     | redshift | 10     | 32         | RMSProp   | MSE  | 0.003365 |
| i2r   | img     | redshift | 10     | 32         | RMSProp   | MAE  | 0.030170 |
| i2r   | img     | redshift | 10     | 32         | Adam      | MSE  | 0.016661 |
| i2r   | img     | redshift | 10     | 32         | Adam      | MAE  | 0.071612 |
| i2r   | img     | redshift | 10     | 64         | RMSProp   | MSE  | 0.003508 |
| i2r   | img     | redshift | 10     | 64         | RMSProp   | MAE  | 0.030480 |
| i2r   | img     | redshift | 10     | 64         | Adam      | MSE  | 0.003498 |
| i2r   | img     | redshift | 10     | 64         | Adam      | MAE  | 0.071558 |
| f2r   | fits    | redshift | 10     | 32         | RMSProp   | MSE  | 0.001930 |
| f2r   | fits    | redshift | 10     | 32         | RMSProp   | MAE  | 0.027434 |
| f2r   | fits    | redshift | 10     | 32         | Adam      | MSE  | 0.001917 |
| f2r   | fits    | redshift | 10     | 32         | Adam      | MAE  | 0.024896 |
| f2r   | fits    | redshift | 10     | 64         | RMSProp   | MSE  | 0.002075 |
| f2r   | fits    | redshift | 10     | 64         | RMSProp   | MAE  | 0.028206 |
| f2r   | fits    | redshift | 10     | 64         | Adam      | MSE  | 0.001870 |
| f2r   | fits    | redshift | 10     | 64         | Adam      | MAE  | 0.024997 |
| s2r   | spectra | redshift | 10     | 32         | RMSProp   | MSE  | 0.004649 |
| s2r   | spectra | redshift | 10     | 32         | RMSProp   | MAE  | 0.033334 |
| s2r   | spectra | redshift | 10     | 32         | Adam      | MSE  | 0.006617 |
| s2r   | spectra | redshift | 10     | 32         | Adam      | MAE  | 0.040978 |
| s2r   | spectra | redshift | 10     | 64         | RMSProp   | MSE  | 0.005554 |
| s2r   | spectra | redshift | 10     | 64         | RMSProp   | MAE  | 0.038496 |
| s2r   | spectra | redshift | 10     | 64         | Adam      | MSE  | 0.014275 |
| s2r   | spectra | redshift | 10     | 64         | Adam      | MAE  | 0.044180 |
| ss2r  | ssel    | redshift | 10     | 32         | RMSProp   | MSE  | 0.004343 |
| ss2r  | ssel    | redshift | 10     | 32         | RMSProp   | MAE  | 0.030403 |
| ss2r  | ssel    | redshift | 10     | 32         | Adam      | MSE  | 0.008098 |
| ss2r  | ssel    | redshift | 10     | 32         | Adam      | MAE  | 0.045030 |
| ss2r  | ssel    | redshift | 10     | 64         | RMSProp   | MSE  | 0.005174 |
| ss2r  | ssel    | redshift | 10     | 64         | RMSProp   | MAE  | 0.031783 |
| ss2r  | ssel    | redshift | 10     | 64         | Adam      | MSE  | 0.006342 |
| ss2r  | ssel    | redshift | 10     | 64         | Adam      | MAE  | 0.053812 |
| b2r   | bands   | redshift | 10     | 32         | RMSProp   | MSE  | 0.003765 |
| b2r   | bands   | redshift | 10     | 32         | RMSProp   | MAE  | 0.027912 |
| b2r   | bands   | redshift | 10     | 32         | Adam      | MSE  | 0.004437 |
| b2r   | bands   | redshift | 10     | 32         | Adam      | MAE  | 0.027528 |
| b2r   | bands   | redshift | 10     | 64         | RMSProp   | MSE  | 0.004726 |
| b2r   | bands   | redshift | 10     | 64         | RMSProp   | MAE  | 0.029070 |
| b2r   | bands   | redshift | 10     | 64         | Adam      | MSE  | 0.004593 |
| b2r   | bands   | redshift | 10     | 64         | Adam      | MAE  | 0.027921 |
| w2r   | wise    | redshift | 10     | 32         | RMSProp   | MSE  | 0.011475 |
| w2r   | wise    | redshift | 10     | 32         | RMSProp   | MAE  | 0.055713 |
| w2r   | wise    | redshift | 10     | 32         | Adam      | MSE  | 0.011523 |
| w2r   | wise    | redshift | 10     | 32         | Adam      | MAE  | 0.055016 |
| w2r   | wise    | redshift | 10     | 64         | RMSProp   | MSE  | 0.011657 |
| w2r   | wise    | redshift | 10     | 64         | RMSProp   | MAE  | 0.056263 |
| w2r   | wise    | redshift | 10     | 64         | Adam      | MSE  | 0.011557 |
| w2r   | wise    | redshift | 10     | 64         | Adam      | MAE  | 0.055382 |

Table 4.12: Hyper-parameters exploration for the single input/output models predicting the redshift.

learning rates; and the batch size is tested between 32 and 64. By looking at the score (again, lower is better) the hyper-parameters that are prone to achieve the best results can be selected. For example, given the results for the  $i2_{sm}$  model, the model achieves a good performance using a batch size of 32, a *RMSProp* optimizer and a MAE loss function. Also, remember that some combinations of hyper-parameters are not directly comparable, for example MSE and MAE give outputs in different ranges due to their definition, but the exploration of parameters is always useful to yield an informed decision of a reasonable good place to start.

Table 4.14 illustrates the hyper-parameters exploration for the single input single output models discussed in the previous section for predicting the object sub-class. Each row in the table represents a full training cycle run for the given number of epochs and changing the batch size, the optimizer and the loss function. For each cycle the available data is split in the three sets: training, validation and testing. The validation dataset is used to monitor model evolution during training, and the score in the table indicates the model performance on the test dataset, that is never seen during training. Since these are classification models, i.e. models that output a discrete value, the only loss function considered is *categorical crossentropy*; the selected optimizers to test are *RMSProp* and *Adam* with the default learning rates; and the batch size is tested between 32 and 64. By looking at the score (higher is better in this case) the hyper-parameters that achieve the best result can be selected. For example, given the results for the  $i2_s$  model, the model achieves the best performance using a batch size of 32 and an *Adam* optimizer.

Table 4.15 illustrates the hyper-parameters exploration for the single input single output models discussed in the previous section for predicting the GZ2 simplified class. Each row in the table represents a full training cycle run for the given number of epochs and changing the batch size, the optimizer and the loss function. For each cycle the available data is split in the three sets: training, validation and testing. The validation dataset is used to monitor model evolution during training, and the score in the table indicates the model performance on the test dataset, that is never seen during training. Since these are classification models, as in the previous case, the only loss function considered is *categorical crossentropy*; the selected optimizers to test are *RMSProp* and *Adam* with the default learning rates; and the batch size is tested between 32 and 64. By looking at the score (higher is better in this case) the hyper-parameters that achieve the best result can be selected. For example, given the results for the  $i2_g$  model, the model achieves



| Model | Input   | Output | Epochs | Batch Size | Optimizer | Loss | Score     |
|-------|---------|--------|--------|------------|-----------|------|-----------|
| i2sm  | img     | smass  | 10     | 32         | RMSProp   | MSE  | 40929.859 |
| i2sm  | img     | smass  | 10     | 32         | RMSProp   | MAE  | 26.048    |
| i2sm  | img     | smass  | 10     | 32         | Adam      | MSE  | 40813.363 |
| i2sm  | img     | smass  | 10     | 32         | Adam      | MAE  | 26.695    |
| i2sm  | img     | smass  | 10     | 64         | RMSProp   | MSE  | 40982.949 |
| i2sm  | img     | smass  | 10     | 64         | RMSProp   | MAE  | 26.736    |
| i2sm  | img     | smass  | 10     | 64         | Adam      | MSE  | 42835.566 |
| i2sm  | img     | smass  | 10     | 64         | Adam      | MAE  | 26.238    |
| f2sm  | fits    | smass  | 10     | 32         | RMSProp   | MSE  | 40600.176 |
| f2sm  | fits    | smass  | 10     | 32         | RMSProp   | MAE  | 27.171    |
| f2sm  | fits    | smass  | 10     | 32         | Adam      | MSE  | 40491.633 |
| f2sm  | fits    | smass  | 10     | 32         | Adam      | MAE  | 25.758    |
| f2sm  | fits    | smass  | 10     | 64         | RMSProp   | MSE  | 40679.352 |
| f2sm  | fits    | smass  | 10     | 64         | RMSProp   | MAE  | 26.600    |
| f2sm  | fits    | smass  | 10     | 64         | Adam      | MSE  | 40903.289 |
| f2sm  | fits    | smass  | 10     | 64         | Adam      | MAE  | 25.931    |
| s2sm  | spectra | smass  | 10     | 32         | RMSProp   | MSE  | 2963.335  |
| s2sm  | spectra | smass  | 10     | 32         | RMSProp   | MAE  | 20.871    |
| s2sm  | spectra | smass  | 10     | 32         | Adam      | MSE  | 3432.733  |
| s2sm  | spectra | smass  | 10     | 32         | Adam      | MAE  | 20.129    |
| s2sm  | spectra | smass  | 10     | 64         | RMSProp   | MSE  | 3159.474  |
| s2sm  | spectra | smass  | 10     | 64         | RMSProp   | MAE  | 21.317    |
| s2sm  | spectra | smass  | 10     | 64         | Adam      | MSE  | 3518.646  |
| s2sm  | spectra | smass  | 10     | 64         | Adam      | MAE  | 20.420    |
| ss2sm | ssel    | smass  | 10     | 32         | RMSProp   | MSE  | 40206.445 |
| ss2sm | ssel    | smass  | 10     | 32         | RMSProp   | MAE  | 22.797    |
| ss2sm | ssel    | smass  | 10     | 32         | Adam      | MSE  | 3018.159  |
| ss2sm | ssel    | smass  | 10     | 32         | Adam      | MAE  | 22.728    |
| ss2sm | ssel    | smass  | 10     | 64         | RMSProp   | MSE  | 40271.656 |
| ss2sm | ssel    | smass  | 10     | 64         | RMSProp   | MAE  | 24.099    |
| ss2sm | ssel    | smass  | 10     | 64         | Adam      | MSE  | 40798.273 |
| ss2sm | ssel    | smass  | 10     | 64         | Adam      | MAE  | 22.380    |
| b2sm  | bands   | smass  | 10     | 32         | RMSProp   | MSE  | 3387.860  |
| b2sm  | bands   | smass  | 10     | 32         | RMSProp   | MAE  | 26.523    |
| b2sm  | bands   | smass  | 10     | 32         | Adam      | MSE  | 5747.023  |
| b2sm  | bands   | smass  | 10     | 32         | Adam      | MAE  | 25.930    |
| b2sm  | bands   | smass  | 10     | 64         | RMSProp   | MSE  | 5520.346  |
| b2sm  | bands   | smass  | 10     | 64         | RMSProp   | MAE  | 26.694    |
| b2sm  | bands   | smass  | 10     | 64         | Adam      | MSE  | 5725.097  |
| b2sm  | bands   | smass  | 10     | 64         | Adam      | MAE  | 25.151    |
| w2sm  | wise    | smass  | 10     | 32         | RMSProp   | MSE  | 4323.260  |
| w2sm  | wise    | smass  | 10     | 32         | RMSProp   | MAE  | 30.034    |
| w2sm  | wise    | smass  | 10     | 32         | Adam      | MSE  | 4391.042  |
| w2sm  | wise    | smass  | 10     | 32         | Adam      | MAE  | 29.858    |
| w2sm  | wise    | smass  | 10     | 64         | RMSProp   | MSE  | 4423.225  |
| w2sm  | wise    | smass  | 10     | 64         | RMSProp   | MAE  | 30.605    |
| w2sm  | wise    | smass  | 10     | 64         | Adam      | MSE  | 4433.951  |
| w2sm  | wise    | smass  | 10     | 64         | Adam      | MAE  | 29.800    |

Table 4.13: Hyper-parameters exploration for the single input/output models predicting the stellar mass.

| Model | Input   | Output   | Epochs | Batch Size | Optimizer | Loss                     | Score    |
|-------|---------|----------|--------|------------|-----------|--------------------------|----------|
| i2s   | img     | subclass | 10     | 32         | RMSProp   | categorical crossentropy | 0.771606 |
| i2s   | img     | subclass | 10     | 32         | Adam      | categorical crossentropy | 0.778097 |
| i2s   | img     | subclass | 10     | 64         | RMSProp   | categorical crossentropy | 0.767020 |
| i2s   | img     | subclass | 10     | 64         | Adam      | categorical crossentropy | 0.787760 |
| f2s   | fits    | subclass | 10     | 32         | RMSProp   | categorical crossentropy | 0.771977 |
| f2s   | fits    | subclass | 10     | 32         | Adam      | categorical crossentropy | 0.782177 |
| f2s   | fits    | subclass | 10     | 64         | RMSProp   | categorical crossentropy | 0.766183 |
| f2s   | fits    | subclass | 10     | 64         | Adam      | categorical crossentropy | 0.778739 |
| s2s   | spectra | subclass | 10     | 32         | RMSProp   | categorical crossentropy | 0.765579 |
| s2s   | spectra | subclass | 10     | 32         | Adam      | categorical crossentropy | 0.767897 |
| s2s   | spectra | subclass | 10     | 64         | RMSProp   | categorical crossentropy | 0.766648 |
| s2s   | spectra | subclass | 10     | 64         | Adam      | categorical crossentropy | 0.713914 |
| ss2s  | sse1    | subclass | 10     | 32         | RMSProp   | categorical crossentropy | 0.765764 |
| ss2s  | sse1    | subclass | 10     | 32         | Adam      | categorical crossentropy | 0.768824 |
| ss2s  | sse1    | subclass | 10     | 64         | RMSProp   | categorical crossentropy | 0.761719 |
| ss2s  | sse1    | subclass | 10     | 64         | Adam      | categorical crossentropy | 0.768415 |
| b2s   | bands   | subclass | 10     | 32         | RMSProp   | categorical crossentropy | 0.758995 |
| b2s   | bands   | subclass | 10     | 32         | Adam      | categorical crossentropy | 0.762148 |
| b2s   | bands   | subclass | 10     | 64         | RMSProp   | categorical crossentropy | 0.757720 |
| b2s   | bands   | subclass | 10     | 64         | Adam      | categorical crossentropy | 0.762742 |
| w2s   | wise    | subclass | 10     | 32         | RMSProp   | categorical crossentropy | 0.772348 |
| w2s   | wise    | subclass | 10     | 32         | Adam      | categorical crossentropy | 0.779303 |
| w2s   | wise    | subclass | 10     | 64         | RMSProp   | categorical crossentropy | 0.753348 |
| w2s   | wise    | subclass | 10     | 64         | Adam      | categorical crossentropy | 0.773345 |

Table 4.14: Hyper-parameters exploration for the single input/output models predicting the object sub-class.

the best performance using a batch size of 32 and an *Adam* optimizer.

Table 4.16 illustrates the final hyper-parameters used for the single input, single output models. The selected parameters in some cases were further fine tuned following some manual changes and testing, besides the parameters exploration aforementioned. Also the number of epochs for some cases had to be increased to allow for the models to converge to a better solution. Besides the hyper-parameters the best score for each model is also illustrated. Table 4.17 illustrates the final hyper-parameters used for the multi input/output models. In this case the selected parameters result from some manual tuning, no hyper-parameters explicit exploration was done. Besides the hyper-parameters, the best score for each model on the test set for each training cycle is also illustrated while the test set split is never seen by the model during the training process. The score interpretation depends of the loss function, for MSE and MAE, since these measure the difference between the models output and the real value, the closer the score is to zero the better, in the case of categorical crossentropy, since this measure the accuracy of the model to predict the correct class, the values fall between 0 and 1, the closer the score is

| Model | Input   | Output | Epochs | Batch Size | Optimizer | Loss                     | Score    |
|-------|---------|--------|--------|------------|-----------|--------------------------|----------|
| i2g   | img     | gz2c   | 10     | 32         | RMSProp   | categorical crossentropy | 0.177365 |
| i2g   | img     | gz2c   | 10     | 32         | Adam      | categorical crossentropy | 0.179617 |
| i2g   | img     | gz2c   | 10     | 64         | RMSProp   | categorical crossentropy | 0.178693 |
| i2g   | img     | gz2c   | 10     | 64         | Adam      | categorical crossentropy | 0.179261 |
| f2g   | fits    | gz2c   | 10     | 32         | RMSProp   | categorical crossentropy | 0.278153 |
| f2g   | fits    | gz2c   | 10     | 32         | Adam      | categorical crossentropy | 0.355011 |
| f2g   | fits    | gz2c   | 10     | 64         | RMSProp   | categorical crossentropy | 0.273864 |
| f2g   | fits    | gz2c   | 10     | 64         | Adam      | categorical crossentropy | 0.322159 |
| s2g   | spectra | gz2c   | 10     | 32         | RMSProp   | categorical crossentropy | 0.268863 |
| s2g   | spectra | gz2c   | 10     | 32         | Adam      | categorical crossentropy | 0.184685 |
| s2g   | spectra | gz2c   | 10     | 64         | RMSProp   | categorical crossentropy | 0.267045 |
| s2g   | spectra | gz2c   | 10     | 64         | Adam      | categorical crossentropy | 0.192330 |
| ss2g  | ssel    | gz2c   | 10     | 32         | RMSProp   | categorical crossentropy | 0.261824 |
| ss2g  | ssel    | gz2c   | 10     | 32         | Adam      | categorical crossentropy | 0.182151 |
| ss2g  | ssel    | gz2c   | 10     | 64         | RMSProp   | categorical crossentropy | 0.261932 |
| ss2g  | ssel    | gz2c   | 10     | 64         | Adam      | categorical crossentropy | 0.256818 |
| b2g   | bands   | gz2c   | 10     | 32         | RMSProp   | categorical crossentropy | 0.246340 |
| b2g   | bands   | gz2c   | 10     | 32         | Adam      | categorical crossentropy | 0.245495 |
| b2g   | bands   | gz2c   | 10     | 64         | RMSProp   | categorical crossentropy | 0.245455 |
| b2g   | bands   | gz2c   | 10     | 64         | Adam      | categorical crossentropy | 0.243466 |
| w2g   | wise    | gz2c   | 10     | 32         | RMSProp   | categorical crossentropy | 0.257320 |
| w2g   | wise    | gz2c   | 10     | 32         | Adam      | categorical crossentropy | 0.262387 |
| w2g   | wise    | gz2c   | 10     | 64         | RMSProp   | categorical crossentropy | 0.258239 |
| w2g   | wise    | gz2c   | 10     | 64         | Adam      | categorical crossentropy | 0.259943 |

Table 4.15: Hyper-parameters exploration for the single input/output models predicting the GZ2 simplified class.

to 1 the better – 1 stands for 100% accuracy. The optimizers used during training are RMSProp and Adam, with the default learning rate of 0.001 except for some models, for example concerning the single output models for predicting the redshift, in this case an adaptive learning rate was used, mainly because the numbers can be very small and as soon as the model starts to get near an optima it may start overshoot it, and so the idea is to start taking smaller steps, i.e. using a smaller learning rate. The full parameters set and history for all the training cycles are available from the *astromlp-models* repository in a *MLflow* (Zaharia et al., 2018) database, an open-source framework for managing the ML models life cycle. The next chapter discusses how the models and other resources are composed together to build end-to-end applications.

| Model | Input   | Output   | Type | Epochs | Batch Size | Optimizer | Loss | Score    |
|-------|---------|----------|------|--------|------------|-----------|------|----------|
| i2r   | img     | redshift | r    | 40     | 32         | RMSProp   | MSE  | 0.00127  |
| f2r   | fits    | redshift | r    | 40     | 64         | Adam      | MSE  | 0.00111  |
| s2r   | spectra | redshift | r    | 20     | 32         | RMSProp   | MSE  | 0.00047  |
| ss2r  | ssel    | redshift | r    | 20     | 32         | RMSProp   | MSE  | 0.00158  |
| b2r   | bands   | redshift | r    | 20     | 64         | RMSProp   | MSE  | 0.05263  |
| w2r   | wise    | redshift | r    | 20     | 32         | RMSProp   | MSE  | 0.03815  |
| i2sm  | img     | smass    | r    | 20     | 64         | Adam      | MAE  | 12.79310 |
| f2sm  | fits    | smass    | r    | 20     | 32         | Adam      | MAE  | 22.20509 |
| s2sm  | spectra | smass    | r    | 20     | 32         | Adam      | MAE  | 17.62649 |
| ss2sm | ssel    | smass    | r    | 20     | 64         | Adam      | MAE  | 14.04072 |
| b2sm  | bands   | smass    | r    | 20     | 64         | Adam      | MAE  | 15.14031 |
| w2sm  | wise    | smass    | r    | 20     | 64         | Adam      | MAE  | 16.90694 |
| i2s   | img     | subclass | c    | 20     | 64         | Adam      | c.c. | 0.81237  |
| f2s   | fits    | subclass | c    | 20     | 32         | Adam      | c.c. | 0.83594  |
| s2s   | spectra | subclass | c    | 20     | 32         | Adam      | c.c. | 0.88456  |
| ss2s  | ssel    | subclass | c    | 20     | 32         | Adam      | c.c. | 0.87131  |
| b2s   | bands   | subclass | c    | 20     | 32         | Adam      | c.c. | 0.81856  |
| w2s   | wise    | subclass | c    | 20     | 32         | Adam      | c.c. | 0.80254  |
| i2g   | img     | gz2c     | c    | 20     | 32         | Adam      | c.c. | 0.48929  |
| f2g   | fits    | gz2c     | c    | 20     | 32         | Adam      | c.c. | 0.51504  |
| s2g   | spectra | gz2c     | c    | 20     | 32         | RMSProp   | c.c. | 0.26308  |
| ss2g  | ssel    | gz2c     | c    | 20     | 32         | RMSProp   | c.c. | 0.27861  |
| b2g   | bands   | gz2c     | c    | 20     | 32         | RMSProp   | c.c. | 0.25681  |
| w2g   | wise    | gz2c     | c    | 20     | 32         | Adam      | c.c. | 0.26090  |

Table 4.16: Summary of the complete list of models with single input and output hyper-parameters definition and best score achieved during model training. Type of problem is abbreviated: regression (r) or classification (c), and categorical crossentropy loss function is abbreviated as c.c.

| Model         | Input   | Output                                | Type | Epochs | Batch Size | Optimizer | Loss                       | Score                                    |
|---------------|---|---------------------------------------|------|--------|------------|-----------|----------------------------|--|
| iFsSSbW2r     | img<br>fits<br>spectra<br>ssel<br>bands<br>wise | redshift                              | r    | 10     | 32         | Adam      | MSE                        | 0.00101                                  |
| iFsSSbW2sm    | img<br>fits<br>spectra<br>ssel<br>bands<br>wise | smass                                 | r    | 10     | 32         | Adam      | MAE                        | 8.93731                                  |
| iFsSSbW2s     | img<br>fits<br>spectra<br>ssel<br>bands<br>wise | subclass                              | c    | 10     | 32         | Adam      | c.c.                       | 0.88171                                  |
| iFsSSbW2g     | img<br>fits<br>spectra<br>ssel<br>bands<br>wise | gz2c                                  | c    | 10     | 32         | Adam      | c.c.                       | 0.49667                                  |
| fSbW2rSM      | fits<br>spectra<br>bands<br>wise                | redshift<br>smass                     | r    | 10     | 32         | Adam      | MSE<br>MAE                 | 0.00170<br>9.41751                       |
| fSbW2sG       | fits<br>spectra<br>bands<br>wise                | subclass<br>gz2c                      | c    | 10     | 32         | Adam      | c.c.<br>c.c.               | 0.88372<br>0.48080                       |
| iFsSSbW2rSMsG | img<br>fits<br>spectra<br>ssel<br>bands<br>wise | redshift<br>smass<br>subclass<br>gz2c | r/c  | 10     | 32         | Adam      | MSE<br>MAE<br>c.c.<br>c.c. | 0.00029<br>7.17838<br>0.87236<br>0.44916 |

Table 4.17: Summary of the complete list of multi input/output models hyper-parameters definition and best score. Type of problem is abbreviated: regression (r) or classification (c), and categorical crossentropy loss function is abbreviated as c.c.



## Chapter 5

# Pipelines for Characterization

*The White Rabbit put on his spectacles. "Where shall I begin, please your Majesty?" he asked. "Begin at the beginning," the King said gravely, "and go on till you come to the end: then stop."*

*– Alice's Adventures in Wonderland*

The previous chapter discusses the creation of a dataset and a collection of Deep Learning (DL) models for studying and inferring galaxies properties. This chapter discusses how these models can be combined together, and with the help of other tools, provide end-to-end pipelines for galaxy characterization.

### 5.1 Models Composition

Following the model generic function behaviour, a similar one can be used to describe a pipeline, for example:

$$\text{pipeline} :: \text{ObjId} \rightarrow \text{Result}$$

where, a pipeline is a function that maps an object unique identifier to a result. The combination of models used to infer properties is abstracted by the pipeline used, that is also why the input is not data any more, but an object (or more specifically an object identifier). The real input data is arbitrary, i.e. it depends on the model(s) used during processing, hence it is inferred by the pipeline implementation in real time. This is also why there was this particular concern to formally define models inputs and outputs, so that it is possible to automatically prepare the real data in the correct arrangement to use as inputs.

The core of a pipeline is the set of models that are used, this automatically entails the inputs that need to be computed for a given object and also entails the outputs (properties) the pipeline is able to infer. More than one model may be inferring the same property, by default a pipeline uses a map-reduce approach (Dean & Ghemawat, 2008), after collecting all the required data, processing is handled to the models for computing the required intermediate results (the map step), and then the intermediate results are reduced to yield the final pipeline output (the reduce step). The entire process is captured in the following three steps summary:

1. Gather **data**: compile all the inputs required for the models

$$data :: ObjId \rightarrow [Input]$$

2. The **map** step: for each model in the pipeline apply the model to the data, this step yields a list of outputs for each property

$$map :: [Model] \rightarrow [Input] \rightarrow [Output]$$

3. The **reduce** step: combine all intermediate outputs into a single final result

$$reduce :: [Output] \rightarrow Result$$

Using a simple function composition it is possible to define the pipeline function as

$$pipeline\ models\ objid = reduce(map(models, data(objid)))$$

where, *objid* is a unique object identifier, the *data* function maps this identifier to a set of inputs, which the *map* function maps to set of outputs using the defined set of models, and finally the *reduce* function builds the final result. This is a high level description, an actual implementation is still required, but it allows for discussing the topic and clearly define the mappings inputs and outputs. Figure 5.1 illustrates the same concept using a more visual construct.

The *data* function needs to output a list of inputs that are consistent with the data types defined in the previous chapter, making sure the arrays have the correct dimensions. The *map* function has the inputs constrained by the data types definitions, and the outputs are also enforced by the models outputs, that already follow the same data types. The only



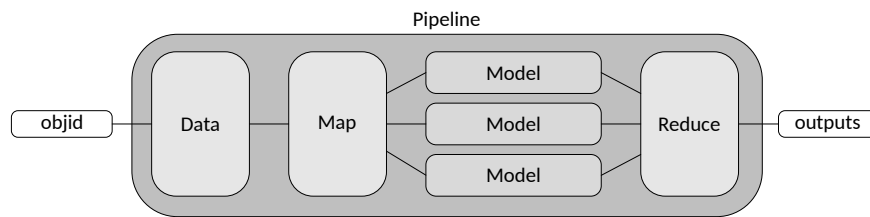


Figure 5.1: Description of a pipeline using a map-reduce approach, each model is applied to the relevant inputs, and the intermediate results are combined to yield the final output.

detail missing is the *reduce* function, i.e. how to actually combine the result of different models or other techniques – a quick side note is that although the illustrated pipelines are only using the devised models any arbitrary tools can contribute to the process as long as their inputs and outputs are well defined, and there is a defined reduce operation able to combine the output with other results. To define the *reduce* function it is possible to take advantage of the models output formal definition. For example, the regression models output a single real value, a *Float*, a straightforward way to combine different outputs from different models to reduce them to a single (final) output is to apply the mean operation for example. In case of classification models, the output is a label, the reduce operation can be for example to select the most voted label between all the models, as this is a strategy already used with models ensembles. And of course devise new reducers that are able to explore more complex ways to combine different results. These reducers can be systematically applied by default in the pipeline because the outputs of the models are well defined.

### Available Pipelines

To illustrate the aforementioned discussion three pipelines are readily available from the *astromlp* library: `One2One`, `CherryPicked` and `Universal`. The first one applies all the available models with a single input and a single output to infer all the galaxy properties, while the second applies a collection of hand picked models to infer the same properties. An `Universal` pipeline is also available for characterizing galaxies, where the models with the best performance are used to estimate all the outputs. This is a baseline to use the library, and the long term goal is to keep updating the combination of models for estimating outputs whenever better options are available. Table 5.1 summarizes the models outputs and corresponding models, or models ensembles, responsible for computing each one,

recalling that the input for the pipeline is an object identifier, it is the responsibility of the framework to infer and gather all the required data automatically.

| Pipeline     | Input | Output               | Models/Ensemble                     |
|--------------|-------|----------------------|-------------------------------------|
| One2One      | objid | redshift             | i2r, f2r, s2r, ss2r, b2r, w2r       |
|              |       | stellar mass         | i2sm, f2sm, s2sm, ss2sm, b2sm, w2sm |
|              |       | sub-class            | i2s, f2s, s2s, ss2s, b2s, w2s       |
|              |       | GZ2 simplified class | i2g, f2g, s2g, ss2g, b2g, w2g       |
| CherryPicked | objid | redshift             | f2r, s2r, ss2r, iFsSSbW2r           |
|              |       | stellar mass         | f2sm, b2sm, w2sm, iFsSSbW2sm        |
|              |       | sub-class            | iFsSSbW2s                           |
|              |       | GZ2 simplified class | i2g, f2g, iFsSSbW2g                 |
| Universal    | objid | redshift             | s2r                                 |
|              |       | stellar mass         | i2sm                                |
|              |       | sub-class            | ss2s                                |
|              |       | GZ2 simplified class | f2g                                 |

Table 5.1: Summary of the available pipelines, including their outputs, and corresponding models or models ensembles to estimate each output.

## 5.2 Results & Discussion

The pipelines implementation takes advantage of all the tools and resources developed throughout this work, but at their core are the models. The collection of models is responsible for actually predicting values of interest, all the others resources act as requirements for the models to fulfil their goal. So the accuracy of properties predictions computed by the pipelines depend mostly on the accuracy of the models used. Table 4.16 and 4.17 summarize the models performance score of the corresponding metrics on the testing slice of the dataset, a typical way to try to assess models accuracy and ability to generalize to new unseen data.

Concerning the redshift estimation, the final score for all single input single output models on the test set is very close to 0 in the order of  $1 \times 10^{-3}$  for the Mean Squared Error (MSE) loss function. The best score, 0.00047, is achieved by the model with the full spectral data as the input, which makes sense, since the spectra is theoretically the best source for redshift information from the available data. In this case it seems that the devised models were able to capture the underlying mapping between the input data and the redshift. Concerning the multi input single output model for predicting the redshift, the performance score on the test set is 0.00101 for the MSE loss function, very close to the single input results, which makes sense since all the input data is there. Concerning the multi input

multi output models that include the redshift prediction, the performance score on the test set is 0.00170 on the MSE loss function for the model with only the regression outputs, and 0.00029 for the model that estimates all the outputs, also on the MSE loss function. Interestingly this result is better than the single input single output model. The function captured by the network illustrated in Figure 4.18 was able to capture the mapping between the inputs and several outputs, hinting that there is an underlying similar structure between the input data and the estimated parameters, but a more in-depth analysis is required to better understand which input data is contributing to explain the model characteristics outputs.

Concerning the stellar mass estimation, the final score for all the single input single output models, on the test set, measuring using the Mean Absolute Error (MAE) loss function is very low when compared to the range of values for the stellar mass, average of  $2.53 \times 10^{10} M_{\odot}$ . Interestingly the best score, 12.79310 (the stellar mass prediction are always downscaled by a factor of  $10^9$ ) is achieved by the model with the RGB images input, but since the stellar mass is related with the *shining* objects, there is a probable relation between these and the image itself. All the models seem to be able to capture the underlying mapping between the inputs and the stellar mass successfully. Concerning the multi input single output model for predicting the stellar mass, the performance score on the test set is 8.93731, a bit better than the best single input model, hinting that there is some combination of inputs that actually provide better predictive information than all the inputs by themselves. Concerning the multi input multi output models that include the stellar mass prediction, the performance score on the test set is 9.41751 on the MAE loss function for the model with only the regression outputs, and 7.17838 for the model that estimates all the outputs, also on the MAE loss function. Again, it's interesting to verify that for the given data this is the model that achieves the best performance for estimating the stellar mass. Hinting, as before, that there is some interesting synergy mechanism in the data, that is not present in the individual inputs.

Concerning the prediction of the object sub-class, the final scores for all the single input single output models measured using the categorical crossentropy function, that conveys the models accuracy to predict the correct class, is above 80%. The best score, 88.456%, is achieved by the model that has the spectral data as input, which makes sense, since the spectra provides the most relevant information to determine the sub-class. All the models achieve a good result, and seem to be able to capture the underlying mapping between all the available inputs and the final sub-class. Concerning the multi input single output model

for predicting the sub-class, the accuracy score on the test set is 88.171%, very close to the best score for the single input models. Concerning the multi input multi output models that include the sub-class prediction, the accuracy score on the test set is 88.372% for the model with only the classification outputs, and 87.236% for the model that estimates all the properties. Very close accuracy values, that show that all these models were able to successfully capture a mapping between the inputs and outputs.

Concerning the prediction of the object Galaxy Zoo 2 (GZ2) simplified class, the final score for all the single input single output models, measured by the categorical cross-entropy function, that conveys the models accuracy to predict the correct class out of the 23 classes described in Appendix D, is divided between values around 20% and values around 50%, being the best accuracy score, 51.504% achieved by the model with the spectral data single input. There is a clear gap in the accuracy of the models that have as input the RGB image and FITS data, closely related, around 50%, when compared to the models that have as input the spectral, bands and WISE data, around 20%. Given that the manual annotation done of the classes was done on the images maybe this can hint that there is a bias in the dataset labels but a more in-depth analysis of the data and the models is required to try to explain these results. Concerning the multi input single outputs model for predicting the GZ2 simplified class, the accuracy score on the test set is 49.667%. Concerning the multi input multi output models that include the GZ2 simplified class prediction, the accuracy score on the test set is 48.080% for the model with only the classification outputs, and 44.916% for the model that estimates all the properties. To double check that there was no fundamental problem with the model implementation or the dataset, an automatically generated model was created, using the Google Cloud service Vertex AI<sup>1</sup>, to predict the GZ2 simplified class from the RGB image. The generated model achieved an accuracy of around 50%, inline with the results from the devised models, hinting that the models are performing as expected.

In summary, concerning the models performance, for the given test sets, randomly extracted from the SDSS Galaxy Subset (SDSS GS) dataset and never seen by the models during training or validation: `iFsSSbW2rSMsG`, a multi input multi output model showed the best performance (for the corresponding metrics) for estimating the redshift and the stellar mass; `s2s`, a single input single output model showed the best accuracy for predicting the sub-class; and, `f2g` a single input single output model showed the best accuracy for

---

<sup>1</sup>Available from: <https://cloud.google.com/vertex-ai> (last accessed: 2022-10-26).

predicting the GZ2 simplified class. But all these are by a small margin, some of the other models have shown very similar performances with different combinations of inputs and outputs. A more in-depth analysis on the relevance on the input for explaining the output is required to have a better understanding on which features really carry the predictive power for the intended properties.

### 5.2.1 Threats to Validity

The previous section discusses the models performance results, this section discusses some possible shortcomings with the devised processes for preparing the data, defining and training the models that may have an impact on the final results.

The first short coming with the illustrated and discussed results is the specific architecture of each model, although some manual endeavour was done to come up with a model able to capture the mappings between inputs and outputs, the model space hypotheses is very big, so it is possible that there are more complex models (or more simple) that are able to better capture the underlying function between inputs and outputs. Although some hyper-parameters tuning was made, again a better combination may exist, like for example the optimizers used and their corresponding parameters, e.g. the learning rate. Also, the hyper-parameters exploration could be expanded to use a larger volume of data, and with an increased number of epochs, because there are some hints that some of the training process may have not converged satisfactorily, for example achieving better performance for predicting the redshift using the image as input, as opposed to the spectra, which was the result of the final training results. The training time, i.e. the number of epochs, can also have an impact on the models final results, training more time may potentially provide a better model. However the numbers of epochs used to train the discussed models was always limited by time and hardware resources.

Another shortcoming with the training and validation processes and results is the dataset itself, the data may already contain some kind of bias in the target variables which makes harder or easier for the models to converge. This is particularly true for predicting the GZ2 simplified class where the current results show that given the available data and models it was not possible to capture a general mapping between the available inputs. Another example is the range of values for the stellar mass of galaxies illustrated in Section 4.2, which contains values that are clearly not expected for the stellar mass of a galaxy. The main shortcoming here seems to be that these objects are not galaxies, i.e. incorrectly

classified as galaxies in the SDSS database, but were chosen as part of the random selection process while creating the dataset. Although these act as outliers and in an optimal situation should not be included for training, they should not impact the performance of the models given they are in a small number. Another possible shortcoming with the devised dataset is some of the options made, for example the size of image cut for each individual galaxy, this can have a possible impact on the data, cutting off relevant information or adding too much noise to the image. Also the selected data inputs may not be the more informative ones for the target estimations. A more in-depth analysis of the correlation between input features could help understand the impact of this shortcoming on the final results.

From a more scientific point of view, a shortcoming with these models is that they do not have any physics constraints on the mappings they produce between the inputs and outputs. In practice the models may devise functions that may not be compatible with the current knowledge of the physical phenomena that drive the studied features, i.e. the created functions are not aware (informed) of the current knowledge on physics concerning the studied topics.

### 5.3 Example Application

To illustrate the practical use of the pipelines discussed in the previous sections, a concrete application was devised that explores their use in a possible real world setting. The application is divided in two main parts: (1) a restful API that uses HTTP requests to process data and characterize objects; and, (2) a web-based online tool providing an interface to query the API and visualize the results. The restful API is an application program interface that uses the HTTP protocol in this case, to submit requests and access results, usually used for communicating data between computer programs (Fielding, 2000). Both the application<sup>2</sup> and the API<sup>3</sup> source code are made available freely under open source licenses.

Figure 5.2 illustrates the *astroMLP for Galaxies* application online interface. From the main page is obvious to see the currently available pipelines for characterizing galaxies, i.e. infer the redshift, stellar mass, sub-class and GZ2 simplified class for SDSS objects.

---

<sup>2</sup>Available from: <https://nunorc.github.io/astromlp-app> (last accessed: 2022-10-18).

<sup>3</sup>Available from: <https://github.com/nunorc/astromlp> (last accessed: 2022-10-18).

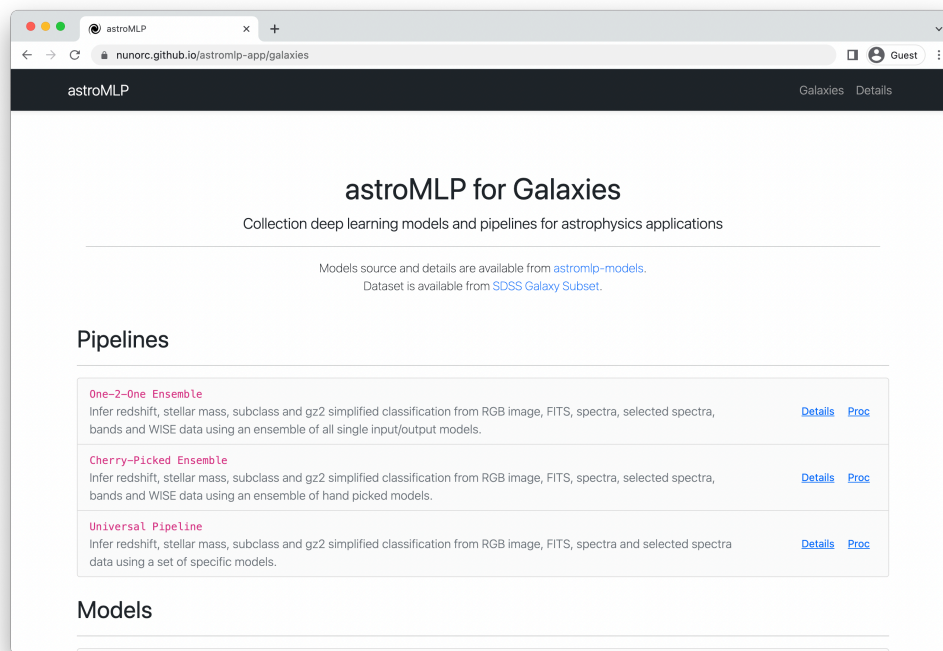


Figure 5.2: *astroMLP for Galaxies* application online interface main page.

Figure 5.3 illustrates the page to view more information about a pipeline, *One-2-One Ensemble* in the example, describing the models ensembles that the pipeline is using to compute the final properties. Figure 5.4 illustrates the visualization of the output of executing the pipeline for a random object.

Besides the pipelines, both the API and the visual interface provide access to the models individually, as hinted in Figure 5.2. This enables to visualize the result of applying each model individually and also enables other applications and tools to simply contact the API to request the results of applying a single model to use the results in more complex workflows, or tools outside the system. Figure 5.5 illustrates the webpage for viewing the information about a specific model, including the model network architecture, predicting redshift from the spectra data in the illustrated example. Figure 5.6 illustrates the output visualization of applying the model to predict the redshift from spectral data for a random object.

The online application allows the exploration of all the available pipelines and models using a visual interface, without requiring the installation of any software or any data download. And the API allows other applications to programmatically apply pipelines and models to use the results in their own workflows or to build new tools. The overarching goal of the application and API is to provide access to the models without requiring any

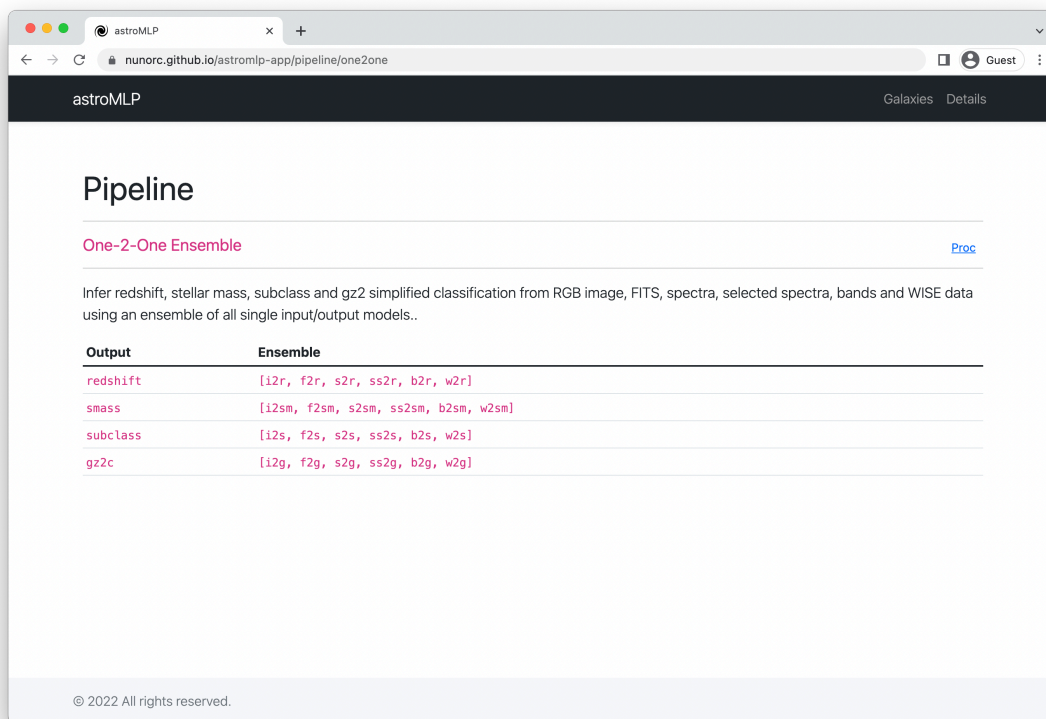
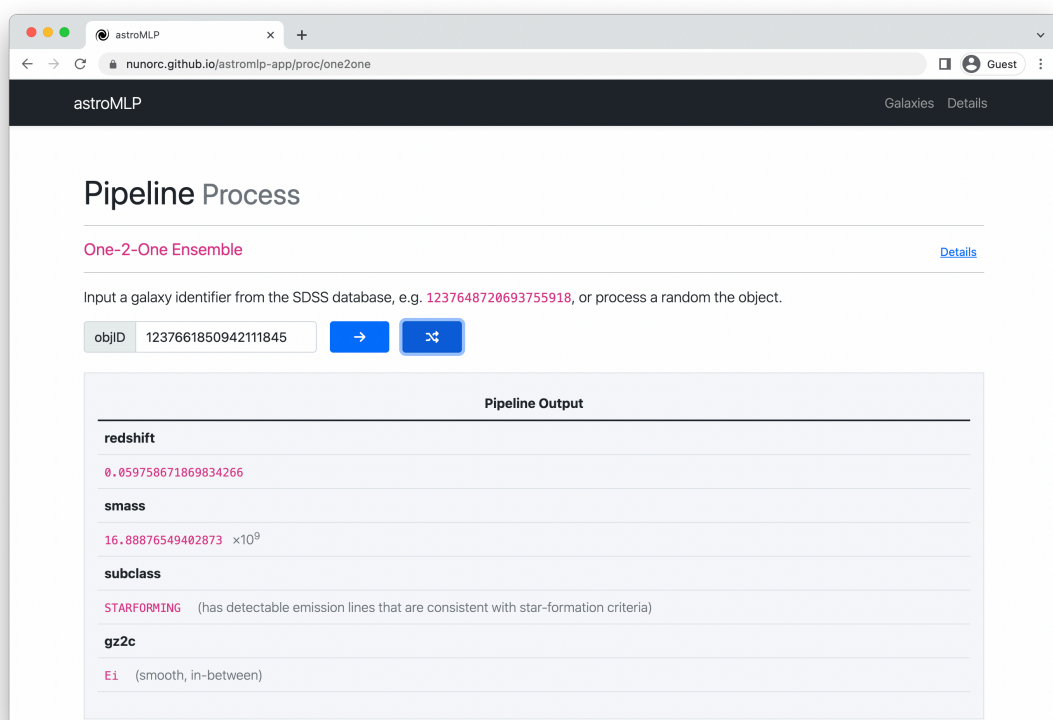


Figure 5.3: *astroMLP for Galaxies* application information about the *One-2-One Ensemble* pipeline.

software download or installation, although all the tools are also available as packages to include in other applications development or workflows.





The screenshot shows a web browser window with the URL `nunorc.github.io/astromlp-app/proc/one2one`. The page title is "astroMLP" and it includes a "Galaxies Details" link. The main heading is "Pipeline Process". Below this, it identifies the "One-2-One Ensemble" pipeline with a "Details" link. A text prompt asks for a galaxy identifier from the SDSS database, with an example ID: 1237648720693755918. An input field contains the objID "123766185094211845" and is accompanied by a blue arrow button and a blue refresh button. The "Pipeline Output" section is a table with the following data:

| Pipeline Output |  |
|-----------------|--|
| <b>redshift</b> |  |
|                 | 0.059758671869834266   |
| <b>smass</b>    |  |
|                 | 16.88876549402873 $\times 10^9$  |
| <b>subclass</b> |  |
|                 | STARFORMING (has detectable emission lines that are consistent with star-formation criteria) |
| <b>gz2c</b>     |  |
|                 | E1 (smooth, in-between)  |

Figure 5.4: *astroMLP for Galaxies* application output result of applying the *One-2-One Ensemble* pipeline.

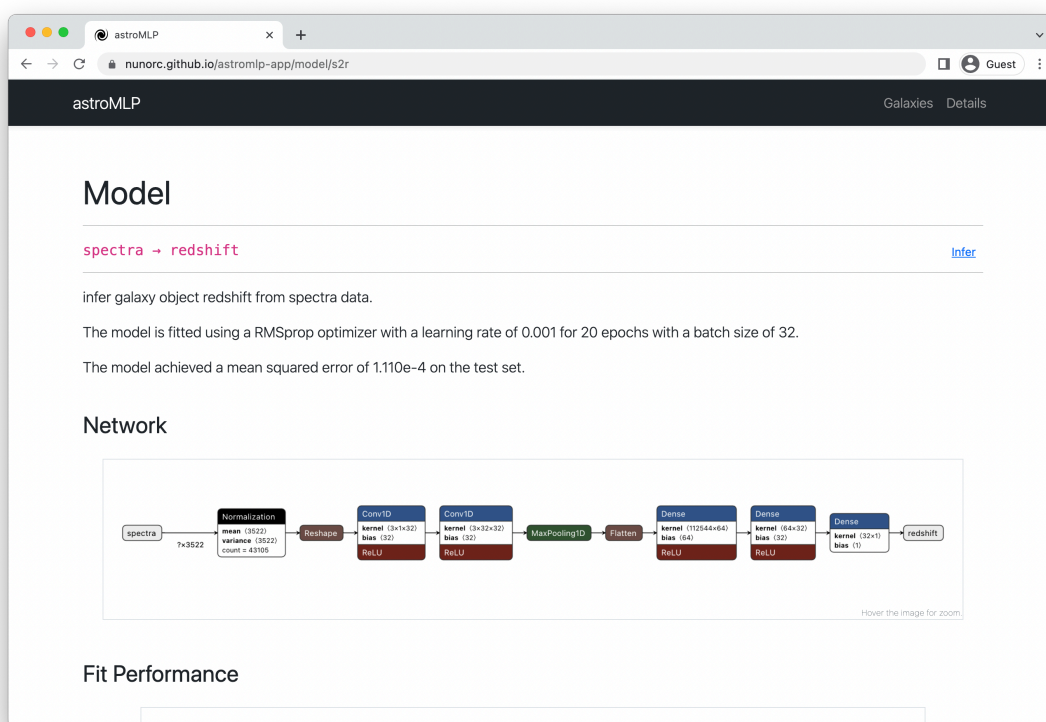


Figure 5.5: *astroMLP for Galaxies* application information about the model to infer the redshift from spectral data page.

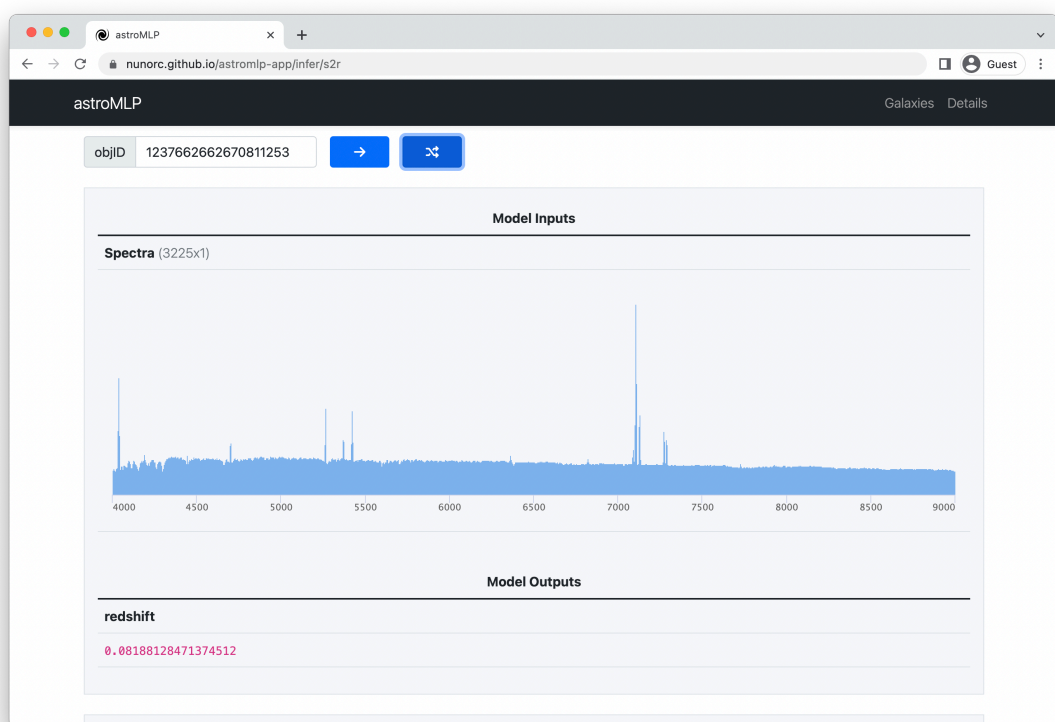


Figure 5.6: *astroMLP for Galaxies* application output result visualization of applying the model to infer the redshift from the spectral data.



## Chapter 6

# Conclusion

*A story has no beginning or end: arbitrarily one chooses that moment of experience from which to look back or from which to look ahead.*

– Graham Greene

This dissertation describes the work done in the field of astronomy and astrophysics, to enhance the understanding and characterization of galaxies using pipelines of Deep Learning (DL) models and heterogeneous sources of data. This chapter discusses some final thoughts and reflections, tries to address the initial proposed research question and discusses some trends for future work.

### 6.1 Final Remarks

Overall the work followed the plan proposed in Section 1.3, during the first part of the work plan, the goal was to build a dataset of heterogeneous data that would include the required information, from a theoretical point of view, that could be used to estimate the intended properties. The SDSS Galaxy Subset (SDSS GS) including around 100k objects was built with data from the SDSS database, this includes RGB images, FITS, spectral, bands and WISE data, and also the relevant information available for the objects, e.g. redshift, class and stellar mass. Section 4.1 describes in more detail the content of the dataset. The major challenge was to relate data with the intended output, i.e. which data is more prone to include the necessary predictive power to build useful models. Some data also required making decisions, for example the size of the image with the object, this can have an impact on models trained using the dataset as discussed in Section 5.2.1,

and other data still needs some further in depth analysis and updates and/or cleaning like for example the objects with a stellar mass clearly outside the expected range of values, most probably due to the objects misclassification. Another step that could help future versions of the dataset and the devised models is a feature importance analysis for each intended combination of inputs and outputs, since maybe not all the used features convey the required predictive power.

During the second part of the work plan the goal was to build, train and test a collection of models for estimating the variables of interest for the objects, namely redshift, stellar mass, sub-class and Galaxy Zoo 2 (GZ2) simplified class. A total of 31 DL models were build, trained, validated and tested, using a total of 518 layers, containing a total of 70 128 689 trainable parameters, using different combinations of single and multi inputs and outputs. Section 4.4 describes in detail all the devised models architectures using combinations of different layers. The major challenge was the definition of the models' network architectures able to capture the mappings between the inputs and outputs, the selection of the loss functions and optimizers and the fine-tuning of the hyper-parameters. Assuming that the available data includes the informative information to accurately infer the value of a given property, the problem during this stage becomes how to define a network that is able to accurately map the data (inputs) to the intended properties (outputs). The manual hyper-parameters exploration described in Section 4.4, although time consuming helped with fine-tuning the parameters, and the networks itself, to enable some of the good results. All the model architectures may be improved, either to make some of the models more simple or more complex, this can help with the models performance, to better generalize to new data, or simply make predictions run faster. All the models were trained, validated and tested using the SDSS GS dataset, which entails that any shortcoming identified with the dataset may have an impact on the models performance.

During the last part of the work plan, the goal was to devise ways to combine all the models and resources developed in previous stages to create and deploy ready to use pipelines for systematic galaxy characterization. Section 5.1 discusses the approach for combining models in a more formal way, enabling the description of model ensembles and the implementation of pipelines using a map-reduce approach, the framework adopted to organize the tools and models application. The formal definition using the Haskell notation, although seldom discussed proved to be a valuable description during the actual

implementation of the resulting packages, enabling the creation of resources easy to integrate and update. The pipelines are immediately available to use from the *astromlp* Python package, and also via a restful API illustrated in Appendix C that enables the estimation of galaxy properties without any software installation or downloads. Although none of the examples illustrates the use of tools from outside the scope of this work, using the devised approach it is possible to easily include the contributions of other libraries and techniques for the pipeline outputs. This is a step forward for a “universal” pipeline for galaxy characterization, although probably there are still some iterations to go, specifically to include more techniques and approaches for estimating individual outputs, and to add more properties of interest to the pipeline final output. Another way to explore the models and pipelines currently available is to visit the online application described in Section 5.3.

Concerning the initially proposed research question, and given the results illustrated in Chapter 4 and discussed in Chapter 5 it seems safe to say that, for the analysed data, the pipelines are able to correctly infer a collection of properties of galaxies with an adequate success rate, maybe except for the GZ2 simplified class. The devised single input single output models achieved a best score on the test sets of 0.00047 on the Mean Squared Error (MSE) loss function for predicting the redshift, 12.79310 on the Mean Absolute Error (MAE) loss function for predicting the stellar mass, an accuracy of 88.456% for predicting the sub-class and an accuracy of 51.504% for predicting the GZ2 simplified class. The multi input and/or multi output models were also able to successfully infer mappings between the inputs and outputs, achieving a best score of 0.00029 on the MSE loss function for predicting the redshift, 7.17838 on the MAE loss function for predicting the stellar mass, 87.236% accuracy for predicting the sub-class and 49.667% accuracy for predicting the GZ2 simplified class. In some cases outperforming the single input single output models, hinting to some underlying synergy between the combination of input data and estimated properties. An in-depth analysis of the measurement of the input features relevance to explain the models output can help with a better understanding the synergy between the features that enable the models achieved performances. The `Universal` pipeline available from the *astromlp* package is a first step to coming up with a generic approach for immediately and consistently process large volumes of data to characterize galaxies populations. Another clear advantage of the use of the pipelines and their implementation approach is how easy they can be used from other tools and included in other workflows, either as a library or via HTTP requests using the available API. Given the huge amount of data

currently available to process, and with the planned missions and surveys this will always increase, reliable, consistent and systematic approaches for estimating galaxies properties enable more complete and in-depth studies to enhance our current understanding of galaxies.

During the development of the required tools, packages and models implementation, there was always a particular care to make sure that adding data, and models, and changing hyper-parameters and another relevant options would be not only possible but also easy. Every little bit of code and information to reproduce the results discussed in this work are made available from the list of shared repositories and resources. Fitting and evaluating models may give slightly different results, given the stochastic nature of some algorithms, initialization, evaluation procedures, etc., or differences in numerical precision. Another relevant detail is that the collection of resources derived from this work (software, datasets, etc.) is shared publicly under permissive licenses, specifically using the MIT or CC license, so that anyone can use and/or continue its' development, or use it in their own work without restrictions or limitations. In a nutshell whenever possible, open source best practices were followed.

## 6.2 Future Work

Some topics and trends for future work include:

- Given the results discussion in Chapter 5 and 6 the first and upmost important future work includes a feature importance analysis to understand which features help better explain the results ([Arrieta et al., 2020](#)), in particular for the multi input multi output models, to better understand the data predictive power for the properties of interest, there are some hints that synergies between inputs can convey important information.
- Expand the dataset by increasing not only the number of objects but also by adding more data from other sources. And also perform some statistical analysis to measure the effective power of the features, and when considering new features to add, to provide enough information to infer the intended outputs, some of the features may not have the required predictive power, i.e. measure feature importance and correlations.



- Add new models, not only exploring the discussed inputs and outputs but taking other advantage of the available data, and also from new data from other sources. Also continue the work of fine-tuning the currently available models hyper-parameters and also models architectures to improve the models performance and increase their ability to better generalize to unseen data;
- Increase the level of complexity of the pipelines strategy, for example allow the output of one model to be used as inputs to another. And also include external tools and techniques to contribute to the results workflow, this may entail updates to the pipeline implementation or new approaches to frame the pipelines, besides map-reduce.
- Increase the performance of the pipelines implementation time-wise, i.e. make the estimation of the properties run faster, by enabling *vectorization* and *parallelization* in some on the calculations that are done, for example for applying model predictions.
- As discussed in Section 5.2.1 the models do not have any constrains based on the current knowledge in the physics of the studied processes, a more recent trend of physics informed models, e.g. [Aliakbari et al. \(2021\)](#), [Mao et al. \(2020\)](#), could help training models aware of some physical known constrains.
- Compare the devised models to models generated using Automated Machine Learning methods ([Hutter et al., 2019](#)), in order not only to validate the proposed model network architectures, but also to use the validation metrics as a baseline for new models.



## Appendix A

# The *astromlp* Package

This section quickly introduces and illustrates the `astromlp` package<sup>1</sup> developed throughout this work. This package, written in Python, provides a framework for building pipelines composed of DL models for astrophysics applications. For installation instructions and more detailed documentation refer to the package website.

Start by importing the `One2One` class from the `astromlp.galaxies` package, that implements one of the readily available pipelines for galaxies characterization:

```
from astromlp.galaxies import One2One
```

Next, create an instance of the `One2One` pipeline, the location for the models store directory may be provided:

```
pipeline = One2One(model_store='./astromlp-models/model_store')
```

The collection of models is available from the `astromlp-models` repository illustrated in Appendix B. The galaxies pipelines are based on SDSS data, so the input to the pipeline is a SDSS object identifier (`objid`), for example to process the object 1237648720693755918 using the selected pipeline run:

```
result = pipeline.process(1237648720693755918)
```

The result object is an instance of `PipelineResult`, a generic class, the outputs of the pipeline processing follow:

```
PipelineResult(redshift=0.0869317390024662, smass=23.44926865895589,  
subclass='STARFORMING', gz2c='ScR')
```

New ensembles of models can easily be created with the map-reduce approach by using the `MapReducPipeline` class, creating a new instance and providing the list of outputs and

<sup>1</sup>Available from: <https://nunorc.github.io/astromlp> (last accessed: 18-10-2022).

corresponding models. For example, to create a new pipeline that computes the redshift using the `i2r` and `f2r` models:

```
from astromlp.galaxies import MapReducePipeline
pipeline = MapReducePipeline({ 'redshift': ['i2r', 'f2r'] })
```

Other modules inside the package may be useful and used in other applications, for example the `Helper` module to work with the dataset derived from SDSS data. We start by importing the module and creating an `helper` instance, optionally giving as argument the path to the dataset:

```
from astromlp.sdss.helper import Helper
helper = Helper(ds='../sdss-gs')
```

It is possible now to query and load data from the dataset using methods. For example, to query the dataset for the identifiers of the objects that have image and FITS data available:

```
ids = helper.ids_list(has_img=True, has_fits=True)
```

To retrieve more information about a specific object the `get_obj` method can be used

```
>>> helper.get_obj(ids[0])
{'objid': 1237674649924469005, 'mjd': 51612, 'plate': 280, 'tile': 108, 'fiberid': 187,
 'run': 6793, 'rerun': 301, 'camcol': 3, 'field': 65, 'ra': 170.36352, 'dec':
 -0.29880776, 'class': 'GALAXY', 'subclass': 'AGN', 'modelMag_u': 19.5784, '
 modelMag_g': 17.81271, 'modelMag_r': 16.87834, 'modelMag_i': 16.42507, 'modelMag_z':
 16.06842, 'redshift': 0.1004738, 'stellarmass': 51983577675.799904, 'w1mag':
 13.423, 'w2mag': 12.911, 'w3mag': 9.871, 'w4mag': 7.613, 'gz2c_f': 'Sb', 'gz2c_s': '
Sb'}
```

For example to retrieve the FITS data for a list of objects the `load_fits` method can be used:

```
>>> data = helper.load_fits(ids)
>>> data.shape
(100054, 61, 61, 5)
```

## Appendix B

# The *astromlp-models* Repository

This section quickly introduces the `astromlp-models` repository<sup>1</sup> developed throughout this work. This repository provides a collection of deep learning models for astrophysics applications. For usage instructions and more detailed documentation refer to the repository website.

The models available in this repository are implemented using Keras, illustrated in Appendix E. To fit the models available in this repository the `astromlp` Python package, illustrated in Appendix A, that includes all the helper classes is also required. To start using the models start by cloning the repository:

```
$ git clone https://github.com/nunorc/astromlp-models.git
```

The models are readily available from the `model_store` directory, and can be directly used from Keras, for example to load the `i2r` model:

```
import tensorflow as tf
model = tf.keras.models.load_model('model_store/i2r')
```

To fit a model using `mlflow`, for example to fit the `i2r` model using the current `python`, i.e. do not create a new environment using `conda`, you can run from the repository directory, and also include this run in the `i2r` experiment:

```
$ mlflow run i2r --experiment-name i2r --no-conda
```

It is also possible to change the parameters to fit the model, namely the number of epochs, the batch size, the loss function and optimizer to use, for example:

```
$ mlflow run i2r -P epochs=10 -P batch_size=32 -P loss=mse -P optimizer=adam --
  experiment-name i2r --no-conda
```

---

<sup>1</sup>Available from: <https://github.com/nunorc/astromlp-models> (last accessed: 07-09-2022).

It is also possible to change the location of the dataset by setting the `ds` parameter:

```
$ mlflow run i2r -P ds=/tmp/sdss-gs --experiment-name i2r --no-conda
```

To view the data concerning the fitting of the available models we can use `mlflow` user interface, including all the parameters and metrics:

```
$ mlflow ui --backend-store-uri sqlite:///mlruns.db
```

To view the logs data concerning the training process using `tensorboard` use:

```
$ tensorboard --logdir i2r/logs/
```

For more information and up to date models to the public repository.

## Appendix C

# The *astromlp* API

This section briefly describes and illustrates the API provided in the *astromlp* package, an expeditious way to use the models with requiring any package installation or data download. The result is returned in JSON format, suitable for machines interoperability. For example to infer the redshift for object 1237648720693755918 using the *i2r* model a query is issued via HTTP to the `/infer/<model>/<objid>` endpoint, the following example illustrates the relevant snippet of the output, which actually includes more information about the object and full input data:

```
$ curl https://astromlp-api.nrc.pt/infer/i2r/1237648720693755918
(...)
"output": [0.07196992635726929], "x": ["img"], "y": ["redshift"]}
```

Besides querying models it is also possible to query a pipeline using the `/proc` endpoint, for example:

```
$ curl https://astromlp-api.nrc.pt/proc/one2one/1237648720693755918
(...)
"output": {"redshift": 0.08778927847743034, "smass": 23.494845072428387, "subclass": "STARFORMING", "gz2c": "ScR"}}
```

For more examples on how to use the API and to fully explore the returned results visit the *astroMLP for Galaxies*<sup>1</sup> application. Although in these examples the API is running on a remote server in the `astromlp-api.nrc.pt` domain, the API source is provided in the *astromlp* package, which means anyone can run their own instance of the API in their local or remote infrastructure.

---

<sup>1</sup>Available from: <https://nunorc.github.io/astromlp-app> (last accessed: 2022-10-18).





## Appendix D

# Galaxy Zoo 2 Simplified Classes

| Label | Description  |
|-------|--|
| A     | artefact, star   |
| Ec    | smooth, cigar-shaped   |
| Ei    | smooth, in-between   |
| Er    | smooth, completely round   |
| SBa   | with features/disks, has bar, dominant bulge prominence                              |
| SBaR  | with features/disks, has bar, dominant bulge prominence, has spiral structure        |
| SBb   | with features/disks, has bar, obvious bulge prominence                               |
| SBbR  | with features/disks, has bar, obvious bulge prominence, has spiral structure         |
| SBc   | with features/disks, has bar, just noticeable bulge prominence                       |
| SBcR  | with features/disks, has bar, just noticeable bulge prominence, has spiral structure |
| SBd   | with features/disks, has bar, no bulge prominence                                    |
| SBdR  | with features/disks, has bar, no bulge prominence, has spiral structure              |
| Sa    | with features/disks, dominant bulge prominence                                       |
| SaR   | with features/disks, dominant bulge prominence, has spiral structure                 |
| Sb    | with features/disks, obvious bulge prominence  |
| SbR   | with features/disks, obvious bulge prominence, has spiral structure                  |
| Sc    | with features/disks, just noticeable bulge prominence                                |
| ScR   | with features/disks, just noticeable bulge prominence, has spiral structure          |
| Sd    | with features/disks, no bulge prominence   |
| SdR   | with features/disks, no bulge prominence, has spiral structure                       |
| Seb   | with features/disks, edge-on, boxy bulge   |
| Sen   | with features/disks, edge-on, no bulge   |
| Ser   | with features/disks, edge-on, round bulge  |



## Appendix E

# The Keras API

This section briefly introduces the `keras` package<sup>1</sup>. Keras is an open-source framework for building and training DL models. All the models devised throughout this work are implemented using this framework inheriting all the advantages of Keras. To quickly illustrate how to use it, let's start by loading some modules:

```
from keras import models, layers, optimizers
```

and create a simple sequential model with two fully connected layers and a single node in the output layer:

```
model = models.Sequential()  
model.add(layers.Dense(64, activation='relu', input_shape=(128,)))  
model.add(layers.Dense(32, activation='relu'))  
model.add(layers.Dense(1))
```

Next, we compile the model by setting the loss function, the optimizer and any metrics that we want to compute:

```
model.compile(optimizer='adam', loss='mse', metrics=['mse'])
```

Finally, we can fit the model to some data tensor  $X$  with a target variable  $y$ , using a batch size of 32 for a total of 100 epochs using the `fit` method:

```
model.fit(X, y, batch_size=32, epochs=100)
```

This is the gist of prototyping models with Keras, for more information refer to the package official documentation.

---

<sup>1</sup>Available from: <https://keras.io> (last accessed: 24-10-2022).



## Appendix F

# SDSS SQL Queries

Query for retrieving all the information for all the objects in the SpecPhoto table:

```
SELECT objID,mjd,plate,tile,fiberID,run,rerun,camcol,field,ra,dec,class,subClass,
       modelMag_u
AS u,modelMag_g AS g,modelMag_r AS r,modelMag_i AS i,modelMag_z AS z, z as redshift
INTO mydb.SpecPhoto
FROM SpecPhoto
WHERE class='GALAXY' AND subClass is not null AND zwarning=0
```

Query for retrieving WISE data:

```
SELECT s.objID, s.class, s.subClass, w.w1mag, w.w2mag, w.w3mag, w.w4mag
INTO mydb.WISE
FROM SpecPhoto s
JOIN WISE_xmatch x ON x.sdss_objid = s.objID
JOIN WISE_allsky w ON x.wise_cntr = w.cntr
WHERE s.class='GALAXY' AND s.subClass is not null AND s.zwarning=0
```



## Appendix G

# Introduction to the Haskell Notation

Haskell is a purely functional and strongly typed programming language. A very brief and summarized introduction to the language notation follows. More information about the language, and detailed resources about its notation are available in the Haskell official website<sup>1</sup>. Some of the illustrated examples use the Glasgow Haskell Compiler<sup>2</sup> interactive environment (ghci) to calculate expressions.

### Functions Definitions and Signatures

A function in Haskell is clearly defined, and has a clear signature, that defines the type of inputs that the function takes, and the type of results the function computes. For example, the signature for a function called *square*, that given an integer computes its square, can be as follows:

$$\text{square} :: \text{Int} \rightarrow \text{Int}$$

This reads as: the *square* function takes an integer, and its result is an integer. Functions can have arbitrary numbers of arguments, for example a function for computing the maximum of four numbers can have the following signature:

$$\text{max} :: \text{Int} \rightarrow \text{Int} \rightarrow \text{Int} \rightarrow \text{Int} \rightarrow \text{Int}$$

To generalize this function to take as input an arbitrary list of numbers, the signature could be changed to take as input a list of integers, and the result is the maximum integer:

---

<sup>1</sup>Available from: <http://haskell.org> (last accessed: 2022-10-26).

<sup>2</sup>Available from: <http://www.haskell.org/ghc/> (last accessed: 2022-10-26).

```
max :: [Int] → Int
```

Most of the times, function signatures can be omitted, in which case the compiler will derive them. But still they are very useful, especially for humans reading the code, to explicitly state which are the arguments to the function and the final result.

The function body follows the signature, for example the complete definition for the *square* function could be as follows:

```
square  :: Int → Int  
square n = n * n
```

The integer value that the function takes as argument is referred using *n*. The result of the function its simply to compute the square of *n*. Intermediate values can be computed inside the function definition. For example, to define a function that computes the cube ( $n^3$ ) of an integer value, using the *square* function:

```
cube  :: Int → Int  
cube n = let  
        sq = square n  
    in  
        sq * n
```

Where, the *let* keyword is used to start a section of intermediate calculations, and the *in* keyword defines the final expression that calculates the function result.

The \$ sign is used in function definitions to avoid the use of parenthesis, anything appearing after takes precedence. The following expressions are equivalent:

```
ghci> square (3 + 2)  
25  
ghci> square $ 3 + 2  
25
```

## Declaring New Data Types

New data types can be defined in Haskell using the *data* keyword. For example, the following statement defines a new data type called *Point2D*, which is composed of two integers (*x* and *y*, the point coordinates):



```
data Point2D = Point2D { x :: Int, y :: Int }
```

New members of this structure are created using the constructor (a function) that is automatically available, and passing the corresponding values in order:

```
ghci> Point2D 4 6
Point2D {x = 4, y = 6}
```

The names of the fields in the data type definition are used as *accessors* to the values. For example, to get the x coordinate of a point *p*:

```
ghci> let p = Point2D 4 6
ghci> x p
4
```

Sometimes, no new data structure is actually required, but to be more clear on values semantics, an alias can be given to other types. For example, in the previous *Point2D* data type definition, the *Int* value is the type for coordinates, to be more explicit an alias for the *Int* type named *Coordinate* can be created using the *type* keyword:

```
type Coordinate = Int
```

Now, the *Coordinate* type can be used, instead of *Int*. Updating the previous example, the data type definition can be written as:

```
data Point2D = Point2D { x :: Coordinate, y :: Coordinate }
```

A new data type can also be defined using possible alternatives. For example the boolean data type has two alternatives: true or false, different alternatives are grouped together using a vertical bar:

```
data Bool = True | False
```



# Bibliography

- Abdurro'uf, Accetta, K., Aerts, C., et al. 2022, ApJS, 259, 35
- Aliakbari, M., Mahmoudi, M., Vadasz, P., & Arzani, A. 2021, in APS Division of Fluid Dynamics Meeting Abstracts, APS Meeting Abstracts, H06.001
- Arnouts, S., Moscardini, L., Vanzella, E., et al. 2002, MNRAS, 329, 355
- Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., et al. 2020, Information fusion, 58, 82
- Baldwin, J. A., Phillips, M. M., & Terlevich, R. 1981, PASP, 93, 5
- Ball, N. M., Brunner, R. J., Myers, A. D., & Tchong, D. 2006, ApJ, 650, 497
- Benítez, N. 2000, ApJ, 536, 571
- Blanton, M. R., Bershady, M. A., Abolfathi, B., et al. 2017, AJ, 154, 28
- Bolton, A. S., Schlegel, D. J., Aubourg, É., et al. 2012, AJ, 144, 144
- Bolzonella, M., Miralles, J. M., & Pelló, R. 2000, A&A, 363, 476
- Brammer, G. B., van Dokkum, P. G., & Coppi, P. 2008, ApJ, 686, 1503
- Bruzual, G. & Charlot, S. 2003, MNRAS, 344, 1000
- Bunn, E. F. & Hogg, D. W. 2009, American Journal of Physics, 77, 688
- Carrasco Kind, M. & Brunner, R. J. 2013, MNRAS, 432, 1483
- Cavanagh, M. K., Bekki, K., & Groves, B. A. 2021, MNRAS, 506, 659
- Chollet, F. 2018, Deep learning with Python (Shelter Island, New York: Manning Publications Co), oCLC: ocn982650571
- Coe, D., Benítez, N., Sánchez, S. F., et al. 2006, AJ, 132, 926

- Collister, A. A. & Lahav, O. 2004, *PASP*, 116, 345
- Comparat, J., Maraston, C., Goddard, D., et al. 2017, arXiv e-prints, arXiv:1711.06575
- Conselice, C. J. 2003, *ApJS*, 147, 1
- Courteau, S., Cappellari, M., de Jong, R. S., et al. 2014, *Reviews of Modern Physics*, 86, 47
- Dawson, K. S., Schlegel, D. J., Ahn, C. P., et al. 2013, *AJ*, 145, 10
- de Vaucouleurs, G. 1959, *Handbuch der Physik*, 53, 275
- Dean, J. & Ghemawat, S. 2008, *Communications of the ACM*, 51, 107
- Deisenroth, M. P., Faisal, A. A., & Ong, C. S. 2020, *Mathematics for machine learning* (Cambridge ; New York, NY: Cambridge University Press)
- Dieleman, S., Willett, K. W., & Dambre, J. 2015, *MNRAS*, 450, 1441
- Duchi, J., Hazan, E., & Singer, Y. 2011, *Journal of machine learning research*, 12
- Fielding, R. T. 2000, *Architectural styles and the design of network-based software architectures* (Doctoral dissertation, University of California, Irvine)
- Fukugita, M., Ichikawa, T., Gunn, J. E., et al. 1996, *AJ*, 111, 1748
- Gallaway, M. 2016, *An Introduction to Observational Astrophysics* (Springer)
- Gerdes, D. W., Sypniewski, A. J., McKay, T. A., et al. 2010, *ApJ*, 715, 823
- Goodfellow, I., Bengio, Y., & Courville, A. 2016, *Deep Learning* (MIT Press)
- Gunn, J. E., Siegmund, W. A., Mannery, E. J., et al. 2006, *AJ*, 131, 2332
- Hart, R. E., Bamford, S. P., Willett, K. W., et al. 2016, *MNRAS*, 461, 3663
- Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. 2012, arXiv e-prints
- Hogg, D. W., Baldry, I. K., Blanton, M. R., & Eisenstein, D. J. 2002, arXiv e-prints, astro
- Hubble, E. P. 1936, *Realm of the Nebulae* (Yale University Press)
- Huertas-Company, M., Primack, J. R., Dekel, A., et al. 2018, *ApJ*, 858, 114

- Hutter, F., Kotthoff, L., & Vanschoren, J. 2019, Automated machine learning: methods, systems, challenges (Springer Nature)
- Ilbert, O., Arnouts, S., McCracken, H. J., et al. 2006, *A&A*, 457, 841
- Janiesch, C., Zschech, P., & Heinrich, K. 2021, *Electronic Markets*, 31, 685
- Karttunen, H., Kröger, P., Oja, H., Poutanen, M., & Donner, K. J. 2007, *Fundamental astronomy* (Springer)
- Ketkar, N. 2017, in *Deep learning with Python* (Springer), 97–111
- Kingma, D. P. & Ba, J. 2014, arXiv e-prints, arXiv:1412.6980
- Koestler, A. 1959, *The sleepwalkers: a history of man's changing vision of the Universe* (London, England ; New York, N.Y., USA: Arkana)
- Lupton, R., Blanton, M. R., Fekete, G., et al. 2004, *PASP*, 116, 133
- Mao, Z., Jagtap, A. D., & Karniadakis, G. E. 2020, *Computer Methods in Applied Mechanics and Engineering*, 360, 112789
- Martin, G., Kaviraj, S., Hocking, A., Read, S. C., & Geach, J. E. 2020, *MNRAS*, 491, 1408
- Mitchell, T. 1997, *Machine learning* (McGraw-hill New York)
- Mo, H., van den Bosch, F. C., & White, S. 2010, *Galaxy Formation and Evolution*
- Nwankpa, C., Ijomah, W., Gachagan, A., & Marshall, S. 2018, arXiv preprint arXiv:1811.03378
- Polsterer, K. L., Gieseke, F., & Kramer, O. 2012, in *Astronomical Society of the Pacific Conference Series*, Vol. 461, *Astronomical Data Analysis Software and Systems XXI*, ed. P. Ballester, D. Egret, & N. P. F. Lorente, 561
- Robson, I. 1996, *Active galactic nuclei*
- Rodrigo, C., Solano, E., & Bayo, A. 2012, *SVO Filter Profile Service Version 1.0*, IVOA Working Draft 15 October 2012
- Sánchez, C., Carrasco Kind, M., Lin, H., et al. 2014, *MNRAS*, 445, 1482

- Sánchez Almeida, J., Aguerri, J. A. L., Muñoz-Tuñón, C., & de Vicente, A. 2010, in ADA 6 - Sixth Conference on Astronomical Data Analysis, ed. J. L. Starck, M. Saber Naceur, & R. Murtagh, 3
- Smee, S. A., Gunn, J. E., Uomoto, A., et al. 2013, *AJ*, 146, 32
- Sparke, L. S. & Gallagher, John S., I. 2000, *Galaxies in the universe : an introduction*
- Sparke, L. S. & Gallagher III, J. S. 2007, *Galaxies in the universe: an introduction* (Cambridge University Press)
- Tuccillo, D., Huertas-Company, M., Decencière, E., et al. 2018, *MNRAS*, 475, 894
- Wang, Q., Ma, Y., Zhao, K., & Tian, Y. 2022, *Annals of Data Science*, 9, 187
- Whiting, A. B. 2004, *The Observatory*, 124, 174
- Wilkinson, D. M., Maraston, C., Goddard, D., Thomas, D., & Parikh, T. 2017, *MNRAS*, 472, 4297
- Willett, K. W., Lintott, C. J., Bamford, S. P., et al. 2013, *MNRAS*, 435, 2835
- Wright, E. L., Eisenhardt, P. R. M., Mainzer, A. K., et al. 2010, *AJ*, 140, 1868
- Yu, J. & Spiliopoulos, K. 2022, arXiv e-prints, arXiv:2209.01018
- Zaharia, M., Chen, A., Davidson, A., et al. 2018, *IEEE Data Eng. Bull.*, 41, 39
- Zhu, X.-P., Dai, J.-M., Bian, C.-J., et al. 2019, *Ap&SS*, 364, 55