Galaxy Morphology Classification System

David Thornton

TU Dublin

This dissertation is submitted for the degree of
Bachelor of Science

December 2025

Declaration

Declaration I/We declare that this material, which I/We now submit for assessment, is entirely my/our own work and has not been taken from the work of others, except where otherwise stated. I/We have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the source cited are identified in the assignment references. I/We understand that plagiarism, collusion, and copying are grave and serious offences and accept the penalties that would be imposed should I/we engage in plagiarism, collusion or copying. I acknowledge that copying someone else's assignment, or part of it, is wrong, and that submitting identical work to others constitutes a form of plagiarism. I/We have read and understood the colleges plagiarism policy 3AS08. This material, or any part of it, has not been previously submitted for assessment for an academic purpose at this or any other academic institution. I have not allowed anyone to copy my work with the intention of passing it off as their own work.

David Thornton
December 2025

Acknowledgements

And I would like to acknowledge ...

Abstract

This is where you write your abstract ...

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Galaxies can be classified in many different ways with different techniques. One of the most common ways is to classify them by their morphology, their physical components and structure. Classifying galaxies this way can provide insight into their formation and life cycle, as well as the general composition of the universe. However, there are a massive number of galaxies in our sky, many of which have already been captured as images. The time and resources it would take for these images to be classified by people would be truly immense. Another, better alternative to this is to train a machine learning model to carry out this task. However, machine learning models are not as trustworthy as people and for scientific applications such as research of galaxy systems this is an issue. This project aims to solve these issues using modern techniques.

1.1 Justification and Motivation

In most cases, galaxies can be identified by people with relatively high accuracy. However, there are extremely large datasets of galaxy images, some having hundreds of millions of images, many of which are unlabelled. Categorising these images will aid in researching galaxy morphologies and the composition of the universe. However, there are too many images to be reliably labelled by people, and the number of these images is only increasing. The implementation of machine learning in galaxy classification can aid in resolving this issue. By training machine learning models to classify galaxy images the time it takes to process these massive datasets can be greatly reduced. Additionally, machine learning models can often find associations people cannot, and understand some datasets better than people can, especially in multidimensional data. This may also lead to machine learning models being more accurate in some cases.

1.2 Project Outline

1.2.1 Main Project Aim

The aim of the project is to develop a hierarchical galaxy morphology classification model that classifies galaxies into the basic Hubble types, then refining classifications into subcategories within the Hubble sequence.

1.2.2 Research questions

- How do flat classification models perform in the task of galaxy morphology classification?
- To what level do hierarchical models outperform flat models?
- What feature extraction techniques are suitable for galaxy morphology classification?
- {Optional} How effective is implementing explainability at informing decision making for galaxy classification configurations?

Flat classification models are often used to classify data, even in cases when the data labels follow a hierarchical structure, such as in galaxy classifications following Hubble Sequencing. This is done even though models can be structured with the same hierarchical structures as the data they are classifying. How do hierarchical models compare against flat models when classifying data with a hierarchical structure?

What feature extraction techniques can be performed on galaxy images and which of these feature extraction techniques provides the most information content? One of the most important components of enhancing model accuracy is providing the model with relevant features. However, finding good features is a significant challenge for any model, especially in computer vision systems.

Can testing a model with explainability measures such as SHAP provide the necessary insight needed to ensure a model is given relevant features? If this is the case, there should be a noticeable increase in model accuracy when features with high explainability scores are used. Additionally, further insight into how the model makes its predictions will greatly increase model trustability.

1.2.3 Objectives

- Conduct research on relevant concepts in machine learning and galaxy morphology.

- Research current literature on galaxy classification systems.
- Analyse possible datasets.
- Train baseline model on single dataset.
- Refine feature extraction process.
- Test different models.
- Compare refined models to baseline and other published models.

1.2.4 Contribution and benefits

from proposal

1.3 Background

To understand what this project involves, there are two main concepts that need to be understood, galaxy morphology classification and machine learning.

1.3.1 Galaxy Morphology

There are many ways to classify galaxy types, including luminosity (brightness), mass, spin, structure, and distance from us to name a few Roberts and Haynes (1994). However, not all of these can be discovered from images alone, so, for the purposes of this project, we will focus on the morphology/structure of the galaxy types. Galaxy morphologies are often categorized based on Hubble Sequencing Conselice (2006) into five main categories, each with its own subcategories, as follows.

Spiral

Spiral galaxies can be identified by their flat, disk-like structure, made up of arms that spiral out from a central bulge, an area of densely packed stars in the center of the galaxy. These galaxies are typically younger, with young blue stars along their disks. Conselice (2006) This galaxy type can be further broken down into subcategories:

- Bulge type: Spiral galaxies can either have a bar-shaped bulge or a circular bulge
- Bulge size: How big the bulge is by comparison to the arms

Fig. 1.1 Spiral Image Kazmierczak (2024)

- Number of arms: These galaxies can have one to 4 arms or sometimes even more
- Tightness of arms: How tightly wound the arms are.

Baillard *et al.* (2011)

Lenticular

Lenticular galaxies are like spiral galaxies, featuring a central bulge and flat disk shape. However, they do not feature strong/noticeable arms, star formation, or large amounts of dust or gas. These are believed to form from spiral galaxies due to effects such as a lack of gas. Due to the lack of arms featured in this galaxy, their only subcategories (which we are concerned with) are the shape and size of the bulge. Conselice (2006)

Elliptical

Elliptical galaxies can be identified by their lack of a disk structure, often being described as a “smooth” shape. These galaxies are believed to have formed from older galaxies that have collided with each other, resulting in their loss of a defined structure. Most of these galaxies are older, being formed from earlier galaxies, and hence have mostly dimmer red/yellow stars. Conselice (2006) These galaxies can be subcategorised by their shape, including:

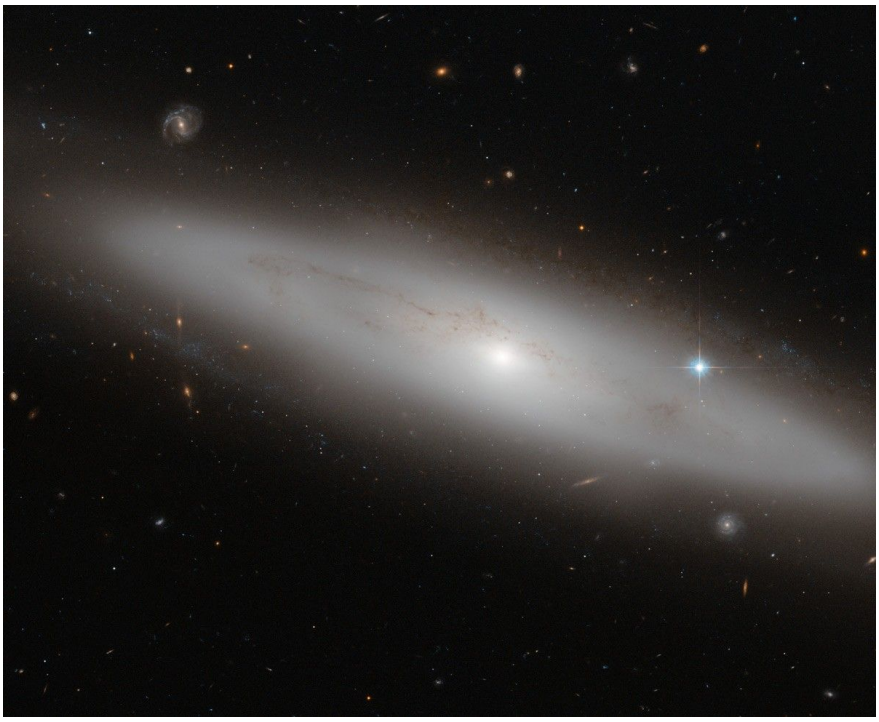

Fig. 1.2 Lenticular Image Kazmierczak (2024)

- Round: Appearing spherical
- In between or ovular: Having an oval shape
- Cigar shaped: Having a longer, narrower shape (like a cigar)

Baillard *et al.* (2011)

Irregular

Irregular galaxies are generally smaller and younger than other galaxy types mentioned previously, in their initial stages of formation. These galaxies do not have any defining shape or structure, and as a result are not often subcategorised Conselice (2006).

Interacting Galaxies

These are galaxies that are affected by other interstellar objects, most commonly other galaxies, merging/interacting with each other. This is by far the least common galaxy type, since galaxies don't spend a lot of time in this state Conselice (2006).

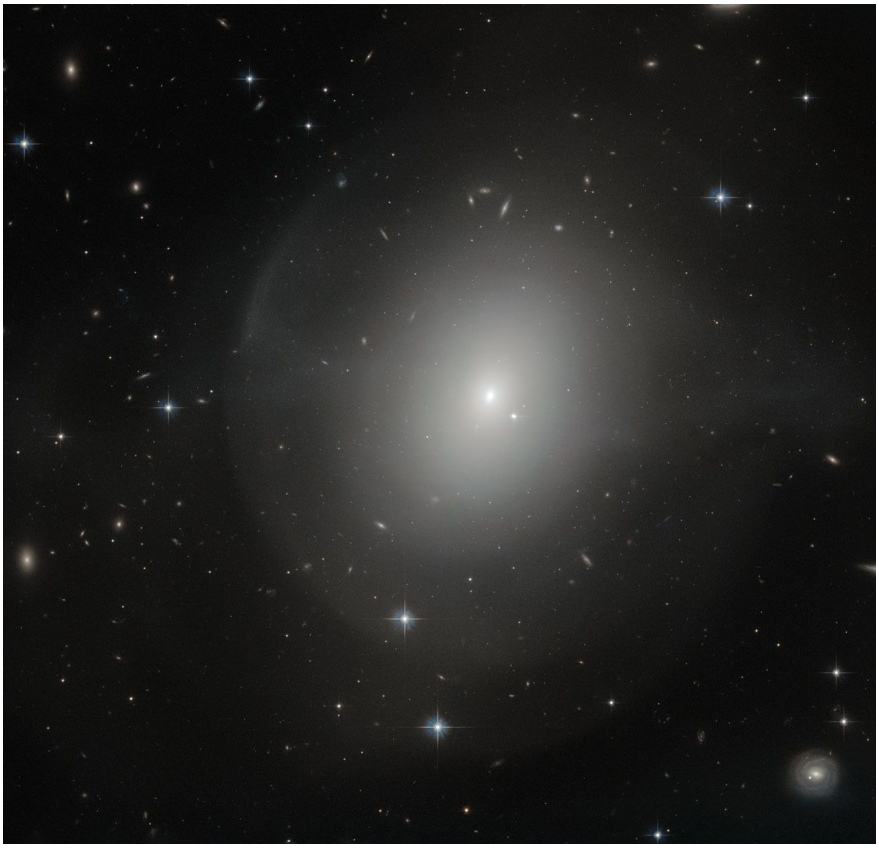

Fig. 1.3 Elliptical Image Kazmierczak (2024)

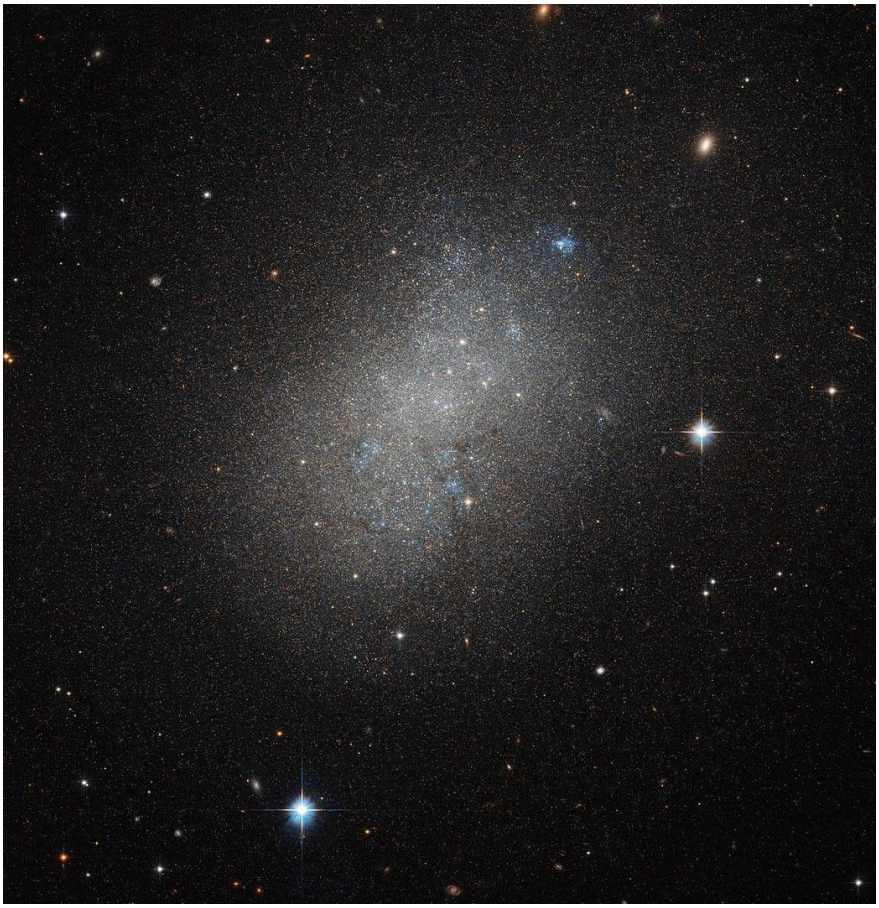

Fig. 1.4 Irregular Image Kazmierczak (2024)

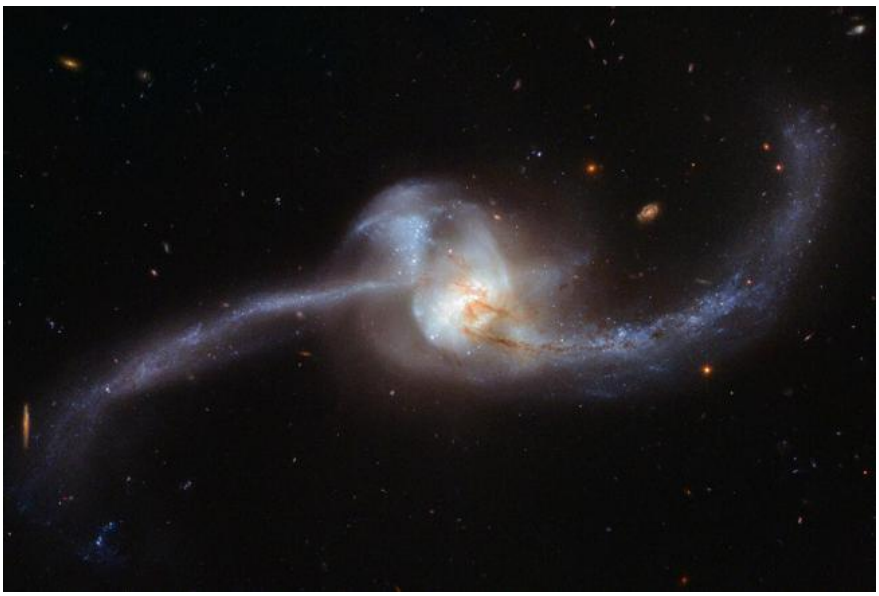

Fig. 1.5 Merging Image Kazmierczak (2024)

A common sixth categorisation is often included for "on-edge galaxies". This is because when a disk-shaped galaxy (spiral or lenticular) is viewed from the edge of the disk, it's very difficult or even impossible to identify whether the galaxy features arms or not.

1.3.2 Machine Learning

Machine learning is a branch of data science that deals with systems that can improve automatically through experience, or training. Machine learning models are trained using data, split into features (what we tell the model) and labels (what we want the model to predict). The model makes its own decisions on how to make predictions, changing its parameters throughout training iterations to become more accurate at predicting the correct label. The model can then be used in practical applications to make predictions on unlabelled data. Jordan and Mitchell (2015)

Computer Vision

Computer vision is a field of machine learning that allows machines to interpret and understand content in images or videos. This is done by training a model to extract and process features or structural components from visual data such as edges, shapes, textures, and patterns. Neural networks (NN) such as convolutional neural networks (CNN) are often used in this process. These features are then used in other layers or models to perform linear regression, clustering or classification tasks, which is the purpose of our model. Szeliski (2022)

Machine Learning Explainability

Machine learning explainability is the process of explaining and understanding how a model makes predictions. In some models, this process is more straightforward. In decision tree models for example, we can plainly see how the model makes decisions at each level of the tree. These are called white box models. Black box models, however, do not produce direct explanations of how they make decisions, which can make them less trustworthy. To eliminate this issue, we can implement external methods to help understand how the model makes its decisions, such as by analysing how important the features that it is given are to making a prediction. Linardatos *et al.* (2020)

Hierarchical Model

Hierarchical models often contain sub models which perform different tasks. Hierarchical models start with a core model, which will classify higher level, more general labels. Based on this initial classification, feature data will then be passed into different sub models, which will carry out classifications on subcategories associated with a given category. Serrano-Pérez *et al.* (2024)

1.4 Feasibility and Ethics

Galaxy classification using machine learning has been heavily researched for several decades and has been proven to be a reliable method of classification. There are also many studies on methods for machine learning explainability, in both general applications and computer vision specifically.

There are many large, well structured, public datasets of galaxy images that can be used for the purposes of this project. Open source nonprofit projects like Galaxy Zoo have organised large datasets to be labelled by the public, such as the galaxy zoo 2 dataset, and have also taken extra steps to account for unreliable and biased classifications. Some of these projects were developed specifically with machine learning experimentation in mind, such as the Galaxy10 dataset. Other projects, such as EFIGI, have had professionals in the field of astronomy classify galaxy images, providing datasets with highly reliable labelling.

The author of this project is familiar with the technologies and concepts involved and has a basic understanding of techniques and methodologies commonly used in similar projects. However, further research into common techniques will need to be conducted to complete the project to a suitable standard. For this reason, the author will conduct further research on common techniques and available resources in the form of a literature review and will converse with professionals in the field for further insight and guidance.

There are no considerable ethical concerns associated with this project, as all datasets under consideration are publicly available. Additionally, no personal data will need to be collected or used in the completion of the project. The main ethical concern for the project is to ensure authenticity of work and accrediting the work of others used throughout.

Chapter 2

Literature Review

literature review - latex wont compile with it in so wont include for this

Chapter 3

Methodology

3.1 Modified Crisp - DM

The main methodology chosen for this thesis is a modified version of the Crisp-DM framework. The original Crisp-DM framework was adapted to include a Baseline modelling section as described below. This was added ensure more rigorous testing standards and that time wasn't wasted on ineffective strategies. The Deployment phase of the traditional Crisp-DM framework was also removed as this is an experimental project and as such there is nothing to be deployed. This framework includes the following steps:

3.2 Problem understanding

As this is a research-based project with some elements of experimentation, the process was not as linear as described, often returning to research. for a better understanding of concepts, experimenting with different datasets and data preparation techniques and building several different models over the course of the project. Despite this, most of the research for this project was carried out as part of a literature review in the earlier stages of development, covering core concepts and techniques in the current literature.

The problem understanding was developed in the initial stages of the project and during the creation of a literature review, which covered core concepts including some of the considered data preparation, modelling and evaluation techniques used throughout the project development.

3.3 Data acquisition, understand and consolidation

The first step of this phase was to choose the dataset from the possible options. When choosing a dataset, the following were considered:

- **Data quality** - The source of the data and its reliability.
- **Labelling system** - How the data was labelled and what information the data provides
- **Dimensionality** - How many data points were given, how much information was provided for each?
- **Formatting** - How well the data was structured and formatted?
- **Distributions** - How well was the data distributed? What are the imbalances like?

After these factors were considered for each dataset, a final dataset was chosen. This dataset was then further analysed to gain better insight not just into how it compares to others, but how this dataset needs to be worked with. To gain a better understanding of the chosen dataset, the following were analysed:

- **Formatting** - How the dataset was structured, including file type, how data was divided among files and how the data would need to be processed to be used?
- **Labels and attributes** - What labels and attributes were given? How can these labels and attributes be used?
- **Inconsistencies** - Missing values, outliers, noise and how these issues can be eliminated/handled.
- **Distributions** - Are there any imbalances in the data, are there underrepresented classes, are they sever enough to affect results and if they are, how can these effects be addressed.
- **Dimensionality** - How much information is stored in each data point? How can the amount of data in each data point be reduced without losing information content? Can information be converted to a better format?

Once a good understanding of the dataset was achieved, data filtering and creation were considered. Data filtering was less of a priority during data preparation since the chosen dataset had already selected high quality data from what was available (the SDSS dataset). Data creation was tested more, generating new data to account for underrepresented classes

in the dataset to alleviate the underfitting they were experiencing. Only the most severely affected classes had new data created for them, using methods that preserved as much of the information content from the sources as possible while still generating unique data.

3.4 Feature extraction

Different feature extraction techniques were tested throughout the development of the thesis. However, before feature extraction was carried out, the data would first have to be pre-processed.

Preprocessing involves converting the data into a usable format. This included combining data form different files and converting it into formats appropriate for model training and dealing with any null values or outliers in the dataset, of which there were none in the chosen dataset.

Several feature extraction techniques were tested to attempt to represent individual characteristics of the image data. Several feature extraction algorithms were implemented, both manual and automatic, the latter of which is explained in the Models Used subsection.

Within manual feature extraction methods, there were algorithms for single value features exaction and extraction methods to enhance certain aspects in images with a lower dimensionality. Single value feature exaction methods produced specific metrics, targeting a certain property. These features were often used to describe attributes of the images used by domain experts to describe a specific property. Other feature extraction methods produced structural representations of the images, such as specific textures or features with less dimensions than the original image. Although single value features are often more explainable, these low dimensionality representations of images features have more data and hence, often capture more information.

3.5 Baseline modelling

The baseline models severed as a comparison to later models and data preparation techniques. These models both had minimal data preparation, only what was needed for the data to be usable to the model i.e. converting the data to an appropriate format. The baseline models also featured few algorithms, only basic versions of algorithms which would be used in most future models, such as the CNN for feature extraction, and simple single dimensional heads for classifications. These models also featured standard hyperparameter with little tuning.

In these models, performance was not a concern in the same way it would be in later models. Instead of trying to maximize performance in these models, the performance metrics

were taken as they were, to be a point of comparison for later iterations. These later models whose performance was less than the baseline were not fine-tuned to maximize performance as better models were. These models also featured no explainability techniques.

The baseline models also served as a basic pipeline which later models would be built around. This meant faster implementation of new methods and less repetition of basic work.

3.6 Data preparation

Once sufficient feature extraction techniques had been implemented, the must be prepared before modelling. This data preparation phase includes dimensionality reduction and feature selection.

Dimensionality reduction is an important step when dealing with image data and hence was an important step in data preparation. Several dimensionality reduction techniques were attempted and assessed by the amount of information they retained compared to how much smaller they were compared to the original data.

Feature selection was also an important factor, since experimentation included the creation of several features. During feature selection, these features would be assessed for their overall information content when classifying galaxies, with only relevant features being chosen when modelling.

3.7 Modelling

Once appropriate data perpetration techniques had been implemented, this data was used to train models with different configurations and algorithms. Different models, configurations and algorithms were tested and compared to evaluate their performance along with the implementation, or lack thereof, of model explainability, although the majority of the explainability found in the model originated from features used and separate explainability techniques implemented after modelling/training. When suitable models and algorithms were found (ones with comparable or better performance to the baseline) they underwent several iterations with different data preparation techniques and hyperparameter tuning to maximize model performance before training and evaluating different models.

Several types of models were implemented for different purposes throughout experimentation. NN models, such as CNNs were used both for classification and automatic feature extraction from image data since they perform well when at recognising complex patterns in image data. Several models which either used automatic feature extraction from CNNs, using two stage training, or other pre extracted features were also implemented. These

models included more basic NN architectures, Support Vector Classifiers (SVC) and decision tree-based classifiers, such as a basic decision tree, XGBoost, or Random forest models. As previously mentioned, these classifiers were also implemented in hierarchical architectures using several models, generally of the same type.

3.8 Evaluation

After training each model with different data preparation techniques, the models were evaluated based on their performance. Performance was measured using predefined metrics to ensure consistency, accuracy and relevant results. These results will be compared with other model and data preparation techniques to assess what data preparation model combinations performed best. The best performing models will be further assessed at the end of the project development.

3.9 Discussion of results

Finally, the best models were re-evaluated using the same metrics used during their evaluation alongside other measures. These models underwent assessments of their explainability, in terms of feature and model explainability. These models were also assessed using external explainability techniques to gain a better understanding of how they make decisions (particularly in models that lack inherently explainable features and algorithms).

These results were then directly compared with the results of the other models and data preparation techniques to find which model performed best in each category and overall. These techniques were also described in terms of what they do well and their limitations. Finally, a discussion of future work and considerations was done, discussing how results may be improved in future work and the limitations work done.

Chapter 4

Experimentation

This chapter describes the experiments used in a theoretical sense. For a more practical, code focused description of these experiments, see Chapter 5 - Implementation.

4.1 Data Acquisition/Selection and Exploration Procedure

4.1.1 Considered

Several datasets were considered in the early stages of development, primarily including:

- EFIGI dataset
- Galaxy Zoo 2 (GZ2)
- Galaxy Zoo 10 DECALS (GZD)
- Nair and Abraham (NA)

Most of these datasets are based on the Sloan Digital Sky Survey (SDSS) dataset or are based on datasets which uses the SDSS dataset, such as RC3. All these datasets are also imbalanced, with the majority classes being those in the Spiral category.

The EFIGI dataset Baillard *et al.* (2011) was created in the Extraction de Formes Idealisées de Galaxies en Imagerie (EFIGI) project. This dataset consists of images and morphological information taken from the RC3 catalogue, which uses images from the SDSS dataset. EFIGI is a professionally labelled dataset with 4458 rows each with 16 descriptive labels and a classification label, each having a predicted value along with an upper and lower confidence limit. Separate folders have images associated with each row, including RGB colour images and individual files for different colour bands, including red, green, blue,

Galaxy	Literal type	Code
Elliptical Compact	cE	-6
Elliptical 0-6	E	-5
Elliptical	cD	-4
Lenticular Early	SO-	-3
Lenticular Intermediate	S00	-2
Lenticular Late	S0+	-1
Spiral 0/a	S0/a	0
Spiral a	Sa	1
Spiral ab	Sab	2
Spiral b	Sb	3
Spiral bc	Sbc	4
Spiral c	Sc	5
Spiral cd	Scd	6
Spiral d	Sd	7
Spiral dm	Sdm	8
Spiral m	Sm	9
Irregular Magellanic	Im	10
Dwarf Spheroid Elliptical	dE	11

Table 4.1 EFIGI Labels

infrared and ultraviolet. This dataset has the lowest number of rows associated with any of the datasets, likely has the most accurate labelling system, being labelled by several domain experts. The labelling system used by EFIGI is a modified version of the RC3 labelling system with the following labels:

This dataset was initially considered because of its detailed labelling systems, with detailed fine class labels. This gave more flexibility when choosing what labelling systems to choose. This dataset was also considered as it was labelled by domain experts, providing more reliable labelling.

The NA dataset Nair and Abraham (2010) is a professionally labelled dataset which used images from the SDSS dataset, choosing 14034 images which had a green colour band magnitude brighter than 16 mag and redshift values less than 0.1. Each image is labelled with 28 descriptive labels and a classification label. There are no separate image datasets which are accompanied with this dataset, so images must be extracted from the SDSS dataset before being used. This dataset is the second smallest considered, but is also the only dataset, aside from the EFIGI dataset, which has professional labels. This dataset also uses a modified version of the RC3 labelling system:

Galaxy	**Literal type**	**Code**
Elliptical	E	-5
Lenticular Early	S0-	-3
Lenticular Late	S0+	-2
Spiral 0/a	S0/a	0
Spiral a	Sa	1
Spiral ab	Sab	2
Spiral b	Sb	3
Spiral bc	Sbc	4
Spiral c	Sc	5
Spiral cd	Scd	6
Spiral d	Sd	7
Spiral dm	Sdm	8
Spiral m	Sm	9
Irregular	Im	10

Table 4.2 Nair and Abraham Labels

This dataset was initially considered due to its label quality, as it was labelled by domain experts. I was also considered as this dataset was larger than the only other professionally labelled dataset considered.

The GZ2 dataset Willett *et al.* (2013) is based on the SDSS dataset. I This dataset was labelled by members of the public, who labelled data by answering questions from a decision tree type list of questions. Several different versions of labels were given, including the number and proportion of people who voted for each class, adjusted votes where voters were given different weights based on the quality of their question answering, and a version which attempted to correct biases due to redshift. Voters voted on a collection of 239695 images in total. This was the largest dataset considered but is also less reliable then the EFIGI and NA datasets, due to unprofessional labelling, although steps were taken to elevate the effects on label quality (as described). Voters voted based on the following decision tree:

GZ2 was initially considered the size of its dataset, which provided a great amount of flexibility in data selection. I was also considered for its labelling system, due to it's intricacy and the amount of information provided, which gave a great amount of flexibility when choosing a labelling structure.

The final dataset considered, GZD, was based on classifications from the GZ2 dataset. This dataset used 17736 classifications from the GZ2 dataset that had corresponding images in the DESI Legacy Imaging Surveys (DECALS) dataset, images of the same galaxies with better resolution. The images used in this dataset, from DECALS, where higher quality than any of the other datasets considered, but also suffered from unprofessional labelling, as its

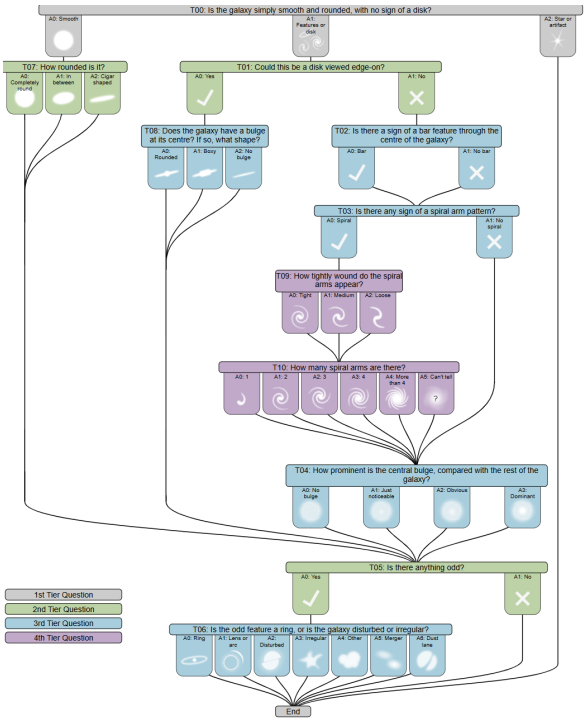

Fig. 4.1 Galaxy Zoo 2 Labels

classes were based on predictions from GZ2. This dataset refined labels from GZ2 into the following labels:

This dataset was initially considered due to its image quality, as it was the only dataset which was not based on the SDSS dataset. It was also considered for the size of the data set, as it was larger than either of the professionally labelled datasets.

4.1.2 Chosen

After considering all datasets, the EFIGI dataset was selected.

The EFIGI dataset preferred over both Galaxy Zoo datasets for its professional labelling, despite having less data points then both and lower image quality than the GZD dataset. EFIGI was chosen over the NA dataset as it had a more preferable labelling system than the NA dataset. The EFIGI dataset had more class labels, providing more options for classifications during experimentation. It was especially preferred for its separate elliptical class labels. It also featured additional labels which could be extracted from images during the data processing phase of experimentation, providing an additional reference for feature accuracy. This dataset was also the most rigorously filtered dataset, with images being assessed for their brightness, resolution of the galaxy itself and ensured to select galaxies

Galaxy	Code
Disturbed	0
Merging	1
Round Smooth	2
In-between Round Smooth	3
Cigar Shaped Smooth	4
Barred Spiral	5
Unbarred Tight Spiral	6
Unbarred Loose Spiral	7
Edge-on without Bulge	8
Edge-on with Bulge	9

Table 4.3 Galaxy Zoo 10 Decals Labels

of all types and subtypes, including rarer types such as dwarfs, irregulars and all elliptical subtypes. For these reasons, the EFIGI dataset was chosen.

4.1.3 Limitations

The primary limitation of the EFIGI dataset is its data imbalances, with some labels (including Elliptical Compact and Elliptical cD) representing less than 2 % of the total dataset. This data imbalance is especially an issue given that this dataset is the smallest of those considered, with only 4458 data points. During Data preparation and modelling, several experiments were conducted to attempt to reduce the impact of these imbalances.

4.1.4 Data Filtering and Creation

No data filtering or data creation techniques have been implemented so far. However, experimentation is planned to attempt to generate data to increase underrepresented classes representation within the training dataset. This include generating new images through rotations and translations, since these procedures should not affect the models ability to learn patterns associated with different galaxy types.

4.2 Feature Extraction

Currently only two feature extraction techniques have been implemented, Local Binary Patterns (LBP) and CNN feature extraction. More feature extraction techniques will be implemented later experimentation, including both single value and feature extraction and methods to techniques.

Local binary patterns are used to represent the texture of an image. It does this by comparing the brightness of a pixel to its neighbouring pixels within a given radius. It then converts these differences into a code which is calculated for each pixel in the image. By adjusting the radius of pixels compared for each code, the scale of the textures can be adjusted, with lower radius values capturing finer textures, in galaxies this could be used to highlight the presence of dust lanes, and higher radius values capturing larger textures, in galaxies this could be used to highlight clumps of stars. In this implementation, radius 3 was used, comparing the neighbouring 24 pixels. This is a moderate radius value and was used to try to general textures within the images.

CNN feature extraction uses a neural network, which learns features automatically through training. CNNs will learn patterns in the data, such as textures and shapes, which distinguish between different classes in the training data. CNNs were implemented to try to capture the most relevant features within the images and was the primary method of feature extraction used in baseline modelling.

4.3 Experimental Procedure for Modelling

4.3.1 Classification Approaches and Algorithms

As discussed in earlier sections, galaxy morphologies follow a hierarchical structure, so several different hierarchical class representations have been implemented during experimentation, which can generally be broken down into flat architectures, classifiers per layer, classifiers per node, classifiers per parent node.

The most basic implementation of hierarchical classifications is to use a flat architecture, or to "flatten" the hierarchical representation of the class structure when modelling and treat leaf classes as if they do not have any hierarchy. Although flat models do not model hierarchy, predictions from these models can benefit from additional algorithms to calculate conditional dependencies or combined probabilities in leaf classes to calculate probabilities for coarse classes.

The next level of complexity up from is the classifier per layer architecture. This model structure features one classifier per level in the hierarchy, which classifies for all the nodes on its level across all trees. Although this architecture does produce predictions for nodes at all layers, basic implementations do not always weigh predictions for sub nodes with predictions from higher nodes by default, so methods for implementing this when making predictions can be useful.

The next level of complexity is the classifier per node architecture. This architecture features a classifier for each node (except leaf nodes) which makes predictions for all possible child nodes of that node. In basic implementations of this model architecture, predictions are made by following the child node with the highest probability, then checking its predictions and doing the same until a leaf node is reached, however, more complex algorithms can be implemented to use probabilities from each layer as weights instead. In either implementation, predictions are inherently hierarchical.

The final hierarchical classifier type is classifier per parent node. This architecture features a classifier for each node, like classifier per node, only in this architecture, each classifier predicts all leaf nodes in its sub tree, rather than predicting its own child nodes. Basic implementations of this architecture use the predictions from the root node when predicting classes, although different algorithms can be implemented to combine predictions from each classifier in the hierarchy to make a final prediction which uses all classifiers.

Several types of models were implemented for different purposes throughout experimentation. NN models, such as CNNs were used both for classification and automatic feature extraction from image data since they perform well when at recognising complex patterns in image data. Several models which either used automatic feature extraction from CNNs, using two stage training, or other pre extracted features were also implemented. These models included more basic NN architectures, Support Vector Classifiers (SVC) and decision tree-based classifiers, such as a basic decision tree, XGBoost, or Random forest models. As previously mentioned, these classifiers were also implemented in hierarchical architectures using several models, generally of the same type.

4.3.2 Hyperparameter Tuning

When experimenting with these algorithms, several hyper parameters were considered for each. In all models, hyperparameter for loss function and optimizers were considered. Loss functions are used to calculate the loss over each training cycle, which is a measure of inaccuracy across all predictions. An important hyper parameter for the loss function, especially in imbalanced dataset, is class weight. Class weights refer to how loss is calculated for different classes, based on their weight or proportion in the data. Aside from loss functions, most other hyper parameters are algorithm specific.

Within NN based models, including CNNs, the main hyper parameter to be considered is the optimizer. The optimizer is what NN use to modify the network weights and biases based on the loss given by the loss function. When configuring an optimizer, the learn rate must also be specified. This describes the magnitude of the changes made to the model during optimization, with lower learn rates resulting in finer changes. However, the main

consideration when creating NN models is the model structure, more specifically, the number of layers and nodes at each layer, the types of layers used, such as linear, convolutions, normalization and dropout, and the hyper parameters given to each layer, such as the dropout rate, which describes proportion of layers dropped, in dropout layers.

In SVC models, an important hyperparameter considered was C, the regularization parameter. This describes how strict the model is when making distinctions between classes. A low C value will allow the model to have more of the training data be miss labelled, which can help with reducing the effect of noise, but also lead to overgeneralization (underfitting) if the C value is too small. Higher C values mean less classes within training data can be mislabelled, meaning the model is more specific in its generalizations, however, this can also lead to overfitting if the C value is too large. Another important hyperparameter considered was the kernel, which describes the algorithm used to make class boundaries. The default value for this is the radial basic function, which works by measuring the distance between data points in all dimensions.

In most tree-based models, the main hyperparameter considered were the max depth and n estimators. Max depth describes the maximum number of levels which can be made in the tree. Increasing this value increases the number of decisions made and the overall size of the tree, which can allow the tree to learn more complex patterns, but can also lead to overfitting if more levels are created then is needed, and also leads to larger trees, which can increase the compute needed. Lower max depth values keep the tree small, which reduces the chance of overfitting and compute time, but can also lead to underfitting if the tree is too small and hence cannot learn more complex patterns. N estimators describe the number of unique trees created in the model and is used in implementations like random forest and XGBoost. More estimators lead to greater amounts of compute needed for training and classification, but also leads to better generalization, since predictions of all trees are considered together when making predictions.

4.3.3 Evaluation metrics

Several evaluation metrics were considered during model development, with the primary metric being the macro average, which is calculated as:

$$\text{MA} = \frac{\sum_{i=1}^{n} F_1}{n}$$

Where n is the number of classes and F1 is the F1 score for class i, calculated as:

$$F_1 = 2 \times \frac{Precision_i \times Recall_i}{Precision_i + Recall_i}$$

Where Precision and Recall for each class are calculated as:

$$\text{Precision}_i = \frac{TP_i}{TP_i + FP_i}$$

$$\text{Recall}_i = \frac{TP_i}{TP_i + FN_i}$$

With TP being the true positive rate, or the number of times this class was correctly predicted, FP is the false positive rate, the amount of times this class was predicted when the true class was a different one and FN is the false negative rate, or the number of times this class was predicted to be a different one.

The F1 score was chosen as overall accuracy was the main concern, so precision and recall should both be considered. The macro version of the F1 score was chosen as the accuracy should be consistent across all classes, and classes with a greater support do not have priority over classes with lesser support, and since the macro average does not take class weights into account, this was the ideal metric of choice.

Although this was the primary metric considered when evaluating the performance of models, other metrics were also evaluated, particularly the F1 score, precision and recall for individual classes. These metrics were evaluated to assess what the underperforming classes were and to attempt to gain insight into what changes needed to be made enhance performance in these classes, such as implementing different data preparation techniques, model algorithms, hyperparameter or data selection techniques. The hierarchical measures of F1, precision and recall were also considered, which are calculated as follows:

$$\text{hPrecision}_\text{i} = \frac{\left|Y^{\text{hier}}_{\text{pred}} \cap Y^{\text{hier}}_{\text{true}}\right|}{\left|Y^{\text{hier}}_{\text{pred}}\right|} \tag{4.1}$$

$$\text{hRecall}_\text{i} = \frac{\left|Y^{\text{hier}}_{\text{pred}} \cap Y^{\text{hier}}_{\text{true}}\right|}{\left|Y^{\text{hier}}_{\text{true}}\right|} \tag{4.2}$$

Where Y hier pred is the predicted class and all its ancestors classes, and Y hier true is the actual class and it's ancestors classes. The top of the fraction is the number of layers in the hierarchy where they intersect, or where the classes on each level are the same, and the bottom is the total number of nodes in the predicted and true class, respectively. The hierarchical F1 and macro average F1 are calculated the same way as normal F1 and macro average scores. These scores give the predictions partial credit if part of the hierarchy is predicted correctly.

4.3.4 Testing and Validation Procedure

In the experiments which were implemented so far, split validation was used. Split validation is done by dividing the dataset up into separate validation and training datasets. During training, the model would only be trained on the training portion of the original dataset, while the validation portion would be used to evaluate the model performance at every epoch in the training cycle. In experimentation, a third dataset was also created for final testing at the end of the training cycle. The data was split into 70 % training, 10 % validation and 20 % testing data.

4.3.5 Preparation Techniques

No data preparation techniques have been implemented so far. However, several preparation techniques will be experimented with, including dimensionality reduction and feature selection techniques, to ensure that the most relevant and best performing features are used.

4.3.6 Baseline

The first implementations were used as baselines to compare later implementations too. All these baseline models used a pre-trained Resnet 18 model which extracted features from the raw images in the EFIGI dataset. All the baseline models were single stage, training the CNN for feature extraction and the classifier heads at the same time. Each of the baseline models used one or more single linear sequential heads for classification. All these models were trained with the following:

- Default hyperparameter
- 50 epochs with batches of size 64
- The iteration with the best validation accuracy was saved
- Cross entropy loss function with balanced class weights
- The Adam optimizer with a learning rate of 0.001

The first of these models had a single classifier head, which performed flat classifications. The second baseline model was a classifier per level model, with two classifier heads. This model also featured methods for combining coarse and fine class predictions, using coarse class probabilities as weights.

The third baseline model was a classifier per node model, with one coarse classifier head and 2 fine classifier heads (only 2 nodes of the 4 coarse nodes have sub classes). When producing probabilities for sub classes, this model combined the probabilities coarse classes with the sub classes associated with its child nodes.

Chapter 5

Implementation

This chapter describes the practical implementation of experiments. For higher level and theoretical explanations of experimentation, see Chapter 4 - Experimentation.

5.1 Tools and Technologies Used

The main programming language used during experimentation was Python. This language was chosen as it is the standard language used in machine learning. It has many libraries and methods for both machine learning and data exploration which made it ideal for this project.

The main Python libraries used during experimentation were PyTorch, SKlearn, Numpy, Pandas and the Python Image Library (PIL). PyTorch was the main library used for modelling as it has many built in classes and methods for modelling. This library was also chosen as it provides more flexibility than other alternatives and supports GPU computing, which greatly increases processing time when working with large amounts of data, such as image data. SKlearn was chosen as it has many methods related to modelling, such as class weight computation, metric calculation, and dataset manipulation. Numpy and Pandas were used for data processing and manipulation and are also needed when working with tensors and arrays needed by PyTorch. PIL was used to work on and manipulate image data during feature extraction and preparation.

Jupyter Notebook was the main IDE used during experimentation. Jupyter Notebook was specifically made for data analytics and data science implementations and has features to make data manipulation, visualization and modelling easier with code cells which allow individual sections of code to be ran at different times, with data from each cell being saved after each new run (as long as that session continues). Jupyter Notebook can also be used with Google Colab for hosted GPU computing, which greatly increased processing time during experimentation.

5.2 Feature Extraction

As previously mentioned, the only feature extraction methods used so far have been CNN feature extraction and LBP. These methods were carried out as follows.

LBP feature extraction was carried out using the SKlearn libraries SKimage class method, local_binary_pattern. To create these features, images were read into the python script as a tensor array and were converted from a 3-dimensional tensor (3 2 dimensional arrays for each colour band) to a single 2-dimensional greyscale version of the image. Greyscalling was done using the SKimage method rgp2gray, which took in the image to be greyscalled as its only argument and returned the greyscale version of the image. The greyscale image was then passed into the local_binary_pattern method, along with hyper parameters for the radius, or the distance from a given pixel at which other pixels were compared (set to 3), the number of points, or the number of pixels which were compared with each pixel (set to 24) and the method, which describes the method used to generate each pixel value (set to 'uniform', meaning the greyscale invariant and rotational invariant were used). Each 2 dimensional associated with each image was then flattened to a 1-dimensional array and was saved in a csv file, with the corresponding image index.

For CNN feature extraction in the baseline models, single stage CNN training was used, meaning the CNN for feature extraction and the classifier heads were trained at the same time. This meant that hyperparameter like loss function and class weights varied in different implementations of classifier heads. All baseline models used a pretrained resnet18 model for feature extraction, which was retrained during classifier training. All implementations used the PyTorch implementation of the Adam optimizer, which had a learn rate of 0.001 and PyTorch's ReduceLROnPlateau scheduler, which used a patience of 3 and mode 'min'.

Specific hyperparameters used in each implementation will be discussed in the implementation section.

5.3 Modelling

5.3.1 Testing and Validation Procedures

All implementations have used the split method for training. This was done using individual datasets for training, testing and validation which were split using SKlearn's train_test_split method. Hyper parameters used this method were random_state 0, stratify using the label column and test_size 0.2, to output train and test datasets. The train dataset was then used to create a new train and validation dataset, with the same hyperparameter aside from 0.125 for

test_size. This generated a training dataset with 80%, validation dataset with 10% and test dataset with 20% of the original dataset.

Data loaders were also used during training, with one for each of the train, test and validation datasets. All three used PyTorch's data loader class. Each data loader took in its corresponding dataset, batch_size 64 for the train data loader and 32 for test and validation, shuffle true in the train data loader and false for the test and validation.

5.3.2 Classification Approaches and Algorithms

The first classification algorithm implemented was the flat baseline model. This implementation replaced the original final classifier layer of the resnet18 model with a single linear layer, which had an input features value of the same value as the original classifier head and 9 output classes. This model used the PyTorch implementation of cross entropy as the loss function with class weights generated SKlearn's compute_class_weight method, which had hyperparameters 'balanced' for class_weight, the unique class values, 0-8 for classes and the label column of the original dataset for y.

The second classification algorithm implemented was the hierarchical classifier per level model. This implementation replaced the original fine classifier layer with two linear layers, one for fine classes and one for coarse classes, which each took in the original number of input features as the original classifier head and had 4 and 9 output classes, respectively. This model used a custom class for hierarchical loss, which combined the cross-entropy loss for each layer, using PyTorch's cross entropy loss function with corresponding class weights for each layer, calculated using the same method and hyperparameters as in the flat model. This method calculated the loss as follows:

$$\text{Loss}_t = Loss_f + Loss_c \times \alpha + Penalty \times \beta$$

Where Loss t is the total loss, Loss f is the loss on fine classes, calculated with cross entropy, Loss c is the loss on coarse classes, calculated with cross entropy, alpha and beta are hyperparameter to adjust the weights of coarse loss and Penalty, which describes whether the predicted fine class belonged to the same coarse class as the predicted coarse class.

This implementation also featured a method to use coarse class predictions as weights when making fine class predictions. This method combined coarse and fine class predictions as follows:

$$\text{P(Combined Fine)} = P(Coarse) \times P(Fine|Coarse)$$

Where P(Coarse) is the probability for a coarse class, and P(Fine|Coarse) is the probability for a fine class, given that the corresponding coarse class prediction is correct, where P(Fine|Coarse) is calculated as:

$$\text{P(Fine|Coarse)} = \frac{P(Fine)}{\sum P(FineInGroup)}$$

$$\text{Loss}_t = \sum Loss_i$$

Where P(Fine) is the base probability for the fine class, which is divided by the combined sum of all the fine class probabilities within this coarse class.

The third classification algorithm implemented was the hierarchical classifier per node model. This implementation replaced the original fine classifier layer with three linear layers, one for predicting coarse classes and one for predicting each of the fine classes in coarse class 0 and 2, which had multiple fine classes. Each of these classifiers took in the original number of input features as the original classifier head and their respective number of output classes. This model also used a custom class for hierarchical loss, which was also based on PyTorch's cross entropy loss function with corresponding class weights. This method calculates loss by combining the loss of all the classifier heads in the model though summation. Coarse class predictions are simply based on the predictions of the coarse class classifier, whoever, fine classes predictions need to be calculated as a combination of coarse and fine class predictions, since there is no global fine classifier. Fine class predictions are calculated as follows:

$$\text{P(Final Fine)} = P(Coarse) \times P(Fine|Coarse)$$

Where P(Final Fine) is the final probability for that fine class and P(Coarse) is the probability for the associated coarse class. The difference here to the method used in the classifier per layer approach is that P(Fine|Coarse) is the probability output for the fine classifier, since this classifier is trained to predict the fine class given the coarse class is correct. If a coarse class has only one fine class, and hence no fine class classifier associated with it, P(Fine|Coarse) is 1, and the final fine class probability is the same as the coarse class probability.

Chapter 6

Results and Discussion

Overall, the performance of the baseline models was mostly consistent across all metrics. All 3 of these models severely struggled at classifying underrepresented classes, particularly the compact elliptical class (fine class 0) as this was the most underrepresented class in the dataset. Specific metrics and performance for each baseline are as follows:

The flat model was the worst performing of the baseline models. This model gave the following classification report:

Class	Precision	Recall	F1-score	Support
0	0.3333	0.7500	0.4615	4
1	0.6949	0.9111	0.7885	45
2	0.5833	0.7778	0.6667	9
3	0.8571	0.6729	0.7539	107
4	0.7023	0.6815	0.6917	135
5	0.7302	0.6970	0.7132	198
6	0.6646	0.7133	0.6881	150
7	0.8563	0.8232	0.8394	181
8	0.7808	0.9048	0.8382	63
Accuracy			0.7466	892
Macro avg	0.6892	0.7702	0.7157	892
Weighted avg	0.7543	0.7466	0.7473	892

Table 6.1 Flat Baseline Classification Report

As previously stated, the main metric for evaluation is the macro F1 score, which in this model was 71.57%. As indicated by a higher absolute F1 score of 74.66%, this model performed better on over represented classes, with class 0 having the lowest support, 4, and

the worse performance, with an F1 score of 46.15%, almost 20% less accuracy then second lowest scoring class, class 2.

The next best performing of the baseline models was the hierarchical classifier per level model, which gave the following classification report:

Fine classes

Class	Precision	Recall	F1-score	hF1-score	Support
0	0.7500	0.7500	0.7500	0.6667	4
1	0.8333	0.7778	0.8046	0.8939	45
2	0.4615	0.6667	0.5455	0.7778	9
3	0.7143	0.8879	0.7917	0.9112	107
4	0.7315	0.5852	0.6502	0.6926	135
5	0.7418	0.6818	0.7105	0.8384	198
6	0.6707	0.7333	0.7006	0.8633	150
7	0.8537	0.7735	0.8116	0.8287	181
8	0.6951	0.9048	0.7862	0.9048	63
Accuracy				0.7399	892
Macro avg	0.7169	0.7512	0.7279	0.8337	892
Weighted avg	0.7462	0.7399	0.7386		892

Coarse classes

Class	Precision	Recall	F1-score	Support
0	0.8361	0.8793	0.8571	58
1	0.7287	0.8785	0.7966	107
2	0.9871	0.9247	0.9549	664
3	0.7125	0.9048	0.7972	63
Accuracy			0.9148	892
Macro avg	0.8161	0.8968	0.8515	892
Weighted avg	0.9269	0.9148	0.9184	892

Table 6.2 Hierarchical Classifier Per Level Classification Report

This model only performed marginally better than the flat approach, with a macro F1 score of 72.20%. This model underperformed most on class 2, with an F1 score of 54.55%. However, this model also predicted coarse classes, which it preformed much better at, likely because of better class distributions and less classes to predict. On coarse classes, this mode achieved an F1 macro accuracy of 85.15%. However, this model still struggled on some classes, but based on class F1 scores, this was not entirely based on class support, with class 0 having the lowest support and still out performing class 1 and 3 by 6%, although class 2 had the most support and the best F1 score by far at 95.49%

The best performing of the baseline models, but also the most complex, was the classifier per node model, which gave the following classification report:

Fine classes

Class	Precision	Recall	F1-score	hF1-score	Support
0	0.7500	0.7500	0.7500	0.8890	4
1	0.7213	0.9778	0.8302	0.9889	45
2	0.8333	0.5556	0.6667	0.7222	9
3	0.7500	0.8411	0.7930	0.9040	107
4	0.7938	0.5704	0.6638	0.6852	135
5	0.7157	0.7374	0.7264	0.8687	198
6	0.6543	0.7067	0.6795	0.8533	150
7	0.8642	0.7735	0.8163	0.8370	181
8	0.7500	0.9048	0.8201	0.8968	63
Accuracy				0.7489	892
Macro avg	0.7592	0.7575	0.7495	0.8427	892
Weighted avg	0.7555	0.7489	0.7466		892

Coarse classes

Class	Precision	Recall	F1-score	Support
0	0.7778	0.9655	0.8615	58
1	0.7563	0.8411	0.7965	107
2	0.9872	0.9322	0.9589	664
3	0.7568	0.8889	0.8175	63
Accuracy			0.9204	892
Macro avg	0.8195	0.9069	0.8586	892
Weighted avg	0.9296	0.9204	0.9231	892

Table 6.3 Hierarchical Classifier Per Node Classification Report

This model had the best fine class F1 macro accuracy, at 74.95%. However, what makes this model particularly good is how similar the performance is per class, with all class F1 scores being between 66.38% and 83.02%. This indicates that the classifier per node approach may be best at classifying underrepresented classes. This is likely because underrepresented fine classes are compared to less overrepresented classes, since each fine classifier only has to classify its child nodes, unlike the previous 2 approaches which had one classifier predict all fine classes. The coarse class accuracy was similar to the classifier per level approach, with a macro F1 score of 85.86%, about 0.5% higher than the per level.

Chapter 7

Conclusions

The main aim of this project is to develop a machine learning model to classify galaxy types based on their morphological structure. This is an important task to help again a better understanding of galaxy morphology and the composition of the universe. Although this task can be done manually by experts in the field, this is an inefficient strategy for this task, as galaxy image datasets are vast, with some containing hundreds of thousands of images while new images are being taken at all times. This project also aims to produce explainability features, to provide further insight into classifications and how they are made.

This project is being developed with a modified version of the Crisp-DM framework which includes steps for developing an understanding of the problem being addressed, acquiring and understanding datasets to use during development of the project, extracting/creating features from the given dataset, creating baseline models for rigorous comparative performance analysis, using data preparation techniques, such as dimensionality reduction and feature selection before modelling using a variety of model algorithms and configurations with different hyperparameter tuning methods. Once these steps have been carried out, methodologies will be rigorously evaluated and compared based on performance.

The dataset chosen for this project was the EFIGI dataset, which was chosen for its detailed labelling system, which was created by several domain experts, and was based on high quality images selected based on a number of factors including brightness and clarity. The main limitation of this dataset which will need to be addressed is its class imbalance, which is made worse by its smaller size.

So far, only two feature extraction methods have been implemented, including a local binary pattern feature, which is used to measure he textures of images, and automatic feature extraction using a CNN, which was done by training a CNN for to extract features from images while training classifier heads to uses these feature to make predictions.

Three different baseline models have been implemented, all of which used minimal data preparation or hyper parameter tuning. All three models used the Adam optimizer and modified versions cross entropy as the loss function. The first of these three classifiers was a flat classification model which used a single linear sequential classifier head to predict fine classes. The second was a hierarchical classifier per level model, which had two single linear sequential classifier heads, one to predict fine classes and one to predict coarse classes. This model also featured methods to combine the probabilities of fine and coarse classes when producing final fine class predictions. The last model implemented was a classifier per node model. This model featured three single linear sequential classifier heads, one which predicted coarse classes and one for each of the coarse classes with multiple fine classes.

Each of these models was trained using the split training procedure, where the original dataset was divided into three smaller datasets, a train dataset (70% of original) which was used to train the data, a validation dataset (10% of original) which was used to test the model during each epoch of training, and a final test dataset (20% of original) which was used to generate final test results.

The results of each model were compared using several different metrics, including the F1 score, precision and recall for each individual class. However, the main metric used to analyse performance was the macro average of the F1 scores of all classes. This score was chosen as the main goal of modelling was to produce a model which was as accurate as possible across all classes. Choosing the macro average helped ensure that underrepresented classes were judged equally with overrepresented classes, and the F1 score ensures that the overall performance of each class was considered.

Analysis of these baseline results showed that hierarchical approaches were marginally better than the flat approach, with the flat model producing a macro F1 score of 71% on fine classes, while the classifier per level and node approaches produced a score of 72% and 75% respectively. The classifier per node approach showed greater promise than the classifier per level approach, as indicated by the fine class macro F1 scores previously mentioned and the coarse class macro F1 scores of 84% and 85% respectively. However, the main reason the classifier per node approach shows more promise than the classifier per head approach was due to the underrepresented fine class performance, which jumped from an F1 score of 46% 75%.

7.1 Plan for completion

Several experiments will be carried out during the project implementation. Experimentation with data creation will be done to attempt to reduce the impact of underrepresented classes

during training. Rotation and translation of existing images theoretically should not impact model predictions since galaxy classifications are not reliant on these aspects, although this could still lead to overfitting of the created datasets if generated images are too similar to the original ones.

Several different feature extraction techniques will be implemented, including methods to enhance certain aspects in images with a lower dimensionality and to extract single value features to describe specific aspects of images, some of which are based on additional labels from the dataset, such as measures of asymmetry, bulge to total ratio and more.

Once a sufficient amount of feature extraction techniques have been implemented, data preparation techniques such as dimensionality reduction, with methods such as PCA, and feature selection, with methods such as analysing features information gain and impact on performance, will be implemented to ensure only relevant data is used in training.

Different model implementations, such as SVC, decision tree based, and NN based models will also be experimented with. Each implementation of these models will experiment with different hyperparameter and configurations, in several different model structure approaches, such as flat, classifier per node, classifier per level and classifier per parent node.

References

Baillard, A., Bertin, E., De Lapparent, V., Fouqué, P., Arnouts, S., Mellier, Y., Pelló, R., Leborgne, J.F., Prugniel, P., Makarov, D. *et al.* (2011) The efigi catalogue of 4458 nearby galaxies with detailed morphology *Astronomy & Astrophysics* **532**, p. A74

Conselice, C.J. (2006) The fundamental properties of galaxies and a new galaxy classification system *Monthly Notices of the Royal Astronomical Society* **373**(4), pp. 1389–1408

Jordan, M.I. and Mitchell, T.M. (2015) Machine learning: Trends, perspectives, and prospects *Science* **349**(6245), pp. 255–260

Kazmierczak, J. (2024) Galaxy types

Linardatos, P., Papastefanopoulos, V. and Kotsiantis, S. (2020) Explainable ai: A review of machine learning interpretability methods *Entropy* **23**(1), p. 18

Nair, P.B. and Abraham, R.G. (2010) A catalog of detailed visual morphological classifications for 14,034 galaxies in the sloan digital sky survey *The Astrophysical Journal Supplement Series* **186**(2), p. 427

Roberts, M.S. and Haynes, M.P. (1994) Physical parameters along the hubble sequence *Annual Review of Astronomy and Astrophysics, Volume 32, 1994, pp. 115-152.* **32**, pp. 115–152

Serrano-Pérez, J., Díaz Hernández, R. and Sucar, L.E. (2024) Bayesian and convolutional networks for hierarchical morphological classification of galaxies *Experimental Astronomy* **58**(2), p. 5

Szeliski, R. (2022) *Computer vision: algorithms and applications* Springer Nature

Willett, K.W., Lintott, C.J., Bamford, S.P., Masters, K.L., Simmons, B.D., Casteels, K.R., Edmondson, E.M., Fortson, L.F., Kaviraj, S., Keel, W.C. *et al.* (2013) Galaxy zoo 2: detailed morphological classifications for 304 122 galaxies from the sloan digital sky survey *Monthly Notices of the Royal Astronomical Society* **435**(4), pp. 2835–2860

Appendix A

Project Gantt Chart

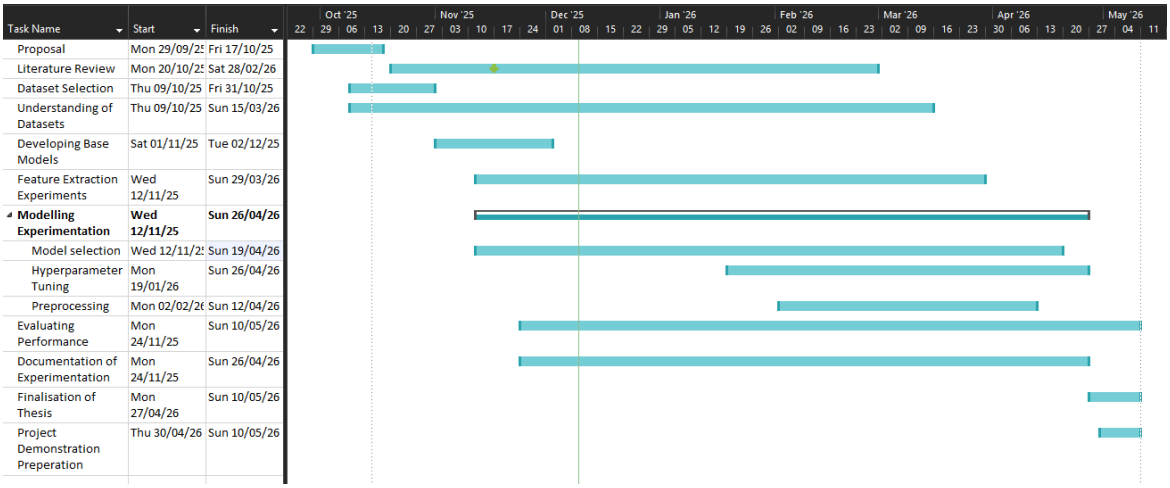

Fig. A.1 Project Gantt Chart

Appendix B

Weekly Progress Documentation

Week 1

Done

A basic understanding of the core concepts needed to understand the project

Proposal started, background and intro, justification and benefits

To do

Continue proposal – to first draught

Finalize datasets to consider in development

Week 2

Done

First proposal draught done

Possible datasets chosen – EFIGI and Nair and Abraham catalog

To do

Finish and submit proposal

Understand chosen datasets, features/labels/attributes

Week 3

Done

Proposal done

Understanding of labels in datasets (core Hubble types and feature scores – EFIGI with uncertainty intervals)

To do

Start literature review – define structure find relevant papers – maybe start on feature extraction processes

Understanding usability of datasets - imbalances, feature quality

Week 4 – 5

Done

Literature review structure and first 2 sections – feature extraction and hierarchical models

Data understanding – image quality and class imbalances across different datasets

Initial dataset choses – EFIGI

To do

Start on baseline model (binary partisan, flat architecture, no preprocessing/manual extraction)

Week 6

Done

Literature review draft 1 complete

First baseline model done, squeezenet, flat architecture, raw images, poor accuracy particularly for underrepresented classes

To do

Implement better evaluation metrics

Try larger models

Deal with data imbalances (weighted sampling/class weights on loss function)

Week 7

Done

Implemented classification report for testing

Used new model, resnet18, which improved performance

Weighted sampling (didn't improve), weighted loss function (improved) and stratified sampling (improved) implemented

Final literature review complete

To do

Get hierarchical baseline

Start methodology section

Week 8

Done

First hierarchical baseline done (classifier per level)

To do

Implement weighted fine classes

Implement classifier per node model

Start methodology section (and document structure)

Week 9

Done

Implemented weighted fine classes (two methods)

Implemented classifier per node model
Started methodology

To do

Implement Hiclass versions (classifier per node, parent node and level)
Finish interim report first draught (Friday)
Start presentation (first draught, after interim)

Week 10

Done

Implemented hiclass versions
Finished interim report
First draught presentation done

To do

Implement hierarchical metrics
Finish presentation