Investigating the Impact of Unsupervised Feature-Extraction from Multi-Wavelength Image Data for Photometric Classification of Stars, Galaxies and QSOs

Annika Lindh
Technological University Dublin, d14128847@mydit.ie

Follow this and additional works at: https://arrow.tudublin.ie/scschcomdis

Part of the Astrophysics and Astronomy Commons, Computer Engineering Commons, and the Computer Sciences Commons

Recommended Citation

This Dissertation is brought to you for free and open access by the School of Computing at ARROW@TU Dublin. It has been accepted for inclusion in Dissertations by an authorized administrator of ARROW@TU Dublin. For more information, please contact yvonne.desmond@tudublin.ie, arrow.admin@tudublin.ie, brian.widdis@tudublin.ie.

This work is licensed under a Creative Commons Attribution-Noncommercial-Share Alike 3.0 License
Investigating the Impact of Unsupervised Feature-Extraction from Multi-Wavelength Image Data for Photometric Classification of Stars, Galaxies and QSOs

Annika Lindh

M.Sc. in Computing (Data Analytics)
Dublin Institute of Technology

A dissertation submitted in partial fulfilment of the requirements of Dublin Institute of Technology for the degree of M.Sc. in Computing (Data Analytics)

2016
I hereby certify that the material which is submitted in this thesis for the award of M.Sc. in Computing (Data Analytics) at Dublin Institute of Technology is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work.

This thesis was prepared according to the regulations for postgraduate study at Dublin Institute of Technology and has not been submitted in whole or in part for another award in any other third level institution.

The work reported on in this thesis conforms to the principles and requirements of the institute’s guidelines for ethics in research.

Signed: _________________________________

Date: 19 September 2016
**ABSTRACT**

This thesis reviews the current state of photometric object type classification in Astronomy and identifies two main gaps: a dependence on handcrafted rules, and a lack of interpretability in the more successful classifiers. To address this, Deep Learning and Computer Vision were used to create a more interpretable model, using unsupervised training to reduce human bias.

The main contribution is the investigation into the impact of using unsupervised feature-extraction from multi-wavelength image data for the classification task. The overall impact is measured by the difference in the macro-averaged F₁ score compared to a baseline model, and the impact on each class is measured by the respective differences in the individual F₁ scores.

The feature-extraction is achieved by implementing an unsupervised Deep Belief Network to extract lower-dimensionality features from the multi-wavelength image data captured by the Sloan Digital Sky Survey. These features are used in a Random Forest classifier alongside 10 color values, calculated from the differences in magnitude between the wavelength bands. These results are compared to those of a separate Random Forest classifier which was trained on only the 10 color values.

A statistically significant increase in the macro-averaged F₁ score of 0.0361 above the baseline result was achieved, indicating that the novel features were useful in the classification task. Increases in the individual F₁ scores of each class were found to be 0.0627 for stars, 0.0206 for galaxies and 0.0252 for QSOs. Relative to the baseline results, the novel features added a 4.35% increase to the overall result, 7.81% to the star class, 2.20% for the galaxy class and 3.32% for the QSO class.

A deeper analysis of the model’s quality was carried out through visualizations of the model’s weights and by sampling from its joint distribution. This analysis suggests that further improvements might be achieved by implementing a sparsity goal for the unsupervised feature-extraction training.

**Keywords:** Deep Learning, Unsupervised Training, Deep Belief Networks, Astronomy, Photometric Classification, Computer Vision, Sloan Digital Sky Survey
ACKNOWLEDGEMENTS

I would like to thank my supervisor, Robert J. Ross, for his guidance and feedback, along with the staff and lecturers at Dublin Institute of Technology for their knowledge and patience.

I would also like to express my gratitude to Gabriele Angeletti for making his Deep Learning implementation in TensorFlow available online. This was of great help when starting out with the TensorFlow framework and the implementation of the model.

A special thank you to Karl Mullen for his love and support, and for the valuable and detailed feedback on the written thesis.

Finally, I would like to thank my father, Jan Lindh, for inspiring my interest in both Astronomy and Machine Learning, and for providing me with valuable feedback on the written thesis.

Funding for SDSS-III has been provided by the Alfred P. Sloan Foundation, the Participating Institutions, the National Science Foundation, and the U.S. Department of Energy Office of Science. The SDSS-III web site is http://www.sdss3.org/.

SDSS-III is managed by the Astrophysical Research Consortium for the Participating Institutions of the SDSS-III Collaboration including the University of Arizona, the Brazilian Participation Group, Brookhaven National Laboratory, Carnegie Mellon University, University of Florida, the French Participation Group, the German Participation Group, Harvard University, the Instituto de Astrofisica de Canarias, the Michigan State/Notre Dame/JINA Participation Group, Johns Hopkins University, Lawrence Berkeley National Laboratory, Max Planck Institute for Astrophysics, Max Planck Institute for Extraterrestrial Physics, New Mexico State University, New York University, Ohio State University, Pennsylvania State University, University of Portsmouth, Princeton University, the Spanish Participation Group, University of Tokyo, University of Utah, Vanderbilt University, University of Virginia, University of Washington, and Yale University.
# TABLE OF CONTENTS

1  Introduction.............................................................................................................. 1
   1.1  Background........................................................................................................... 1
   1.2  Research Project.................................................................................................. 3
       1.2.1  Problem Definition......................................................................................... 3
       1.2.2  Research Question and Scope........................................................................... 3
       1.2.3  Limits of the Scope......................................................................................... 4
       1.2.4  Hypotheses..................................................................................................... 4
       1.2.5  Research Objectives......................................................................................... 5
   1.3  Research Methodologies....................................................................................... 6
       1.3.1  Secondary Research......................................................................................... 6
       1.3.2  Primary Research............................................................................................. 6
   1.4  Thesis Roadmap.................................................................................................... 7

2  Literature Review and Related Work....................................................................... 9
   2.1  Overview............................................................................................................... 9
   2.2  Machine Learning for Astronomy........................................................................ 9
       2.2.1  Astronomy Data.............................................................................................. 10
       2.2.2  Data Surveys.................................................................................................. 11
       2.2.3  Current Approaches....................................................................................... 12
       2.2.4  Summary......................................................................................................... 15
   2.3  Gaps ...................................................................................................................... 15
   2.4  Deep Learning and its Suitability to the Research Problem............................... 16
       2.4.1  A Brief Introduction to Deep Learning.......................................................... 16
       2.4.2  Restricted Boltzmann Machines and Deep Belief Networks....................... 17
       2.4.3  Suitability to the Research Question and the Field of Astronomy.............. 18
   2.5  Summary............................................................................................................. 19
3 Research Design and Implementation ....................................................... 20
  3.1 Overview .......................................................................................... 20
  3.2 Data Preparation ................................................................................. 22
    3.2.1 Baseline Features ........................................................................ 22
    3.2.2 Acquiring the Dataset ................................................................. 23
    3.2.3 Image Preprocessing ................................................................... 26
    3.2.4 Splitting the Dataset into Training, Validation and Test Sets ....... 29
  3.3 Software Implementation ..................................................................... 30
    3.3.1 Model Design .............................................................................. 30
    3.3.2 Software Design .......................................................................... 32
    3.3.3 Memory Management ................................................................... 34
    3.3.4 RBM Implementation .................................................................. 34
    3.3.5 DBN Implementation .................................................................... 36
    3.3.6 Fine-tuning the DBN .................................................................... 36
    3.3.7 Random Forest Classifier Implementation .................................. 38
    3.3.8 The Software Development Process .......................................... 39
  3.4 Experiments ....................................................................................... 44
    3.4.1 Hardware Specifications .............................................................. 44
    3.4.2 Running the Experiments ............................................................. 45
    3.4.3 Hyper-Parameter Tuning ............................................................... 45
    3.4.4 Evaluation Metrics ....................................................................... 46
    3.4.5 Statistical Methods ....................................................................... 47
  3.5 Analysis Techniques .......................................................................... 47
  3.6 Limitations ......................................................................................... 48
  3.7 Summary ........................................................................................... 49
4 Results .................................................................................................. 50
  4.1 Overview ........................................................................................... 50
6.6 Future Work and Recommendations .......................................................... 71

6.6.1 Improving the Discriminative Performance ........................................... 71

6.6.2 Improving the Quality of the Features .................................................. 71

6.6.3 Investigating the Multi-Wavelength Aspect .......................................... 71

6.6.4 Suitability to Transfer Learning ............................................................. 72

6.6.5 Applied Research .................................................................................... 72

6.7 Impact ........................................................................................................ 72

7 References ..................................................................................................... 73
TABLE OF FIGURES

Figure 3.1. Screenshot of the CasJobs interface, showing a finished SQL job. .......... 26
Figure 3.2. Examples of image data after preprocessing, reconstructed into PNG images for visual confirmation of their quality. The images are taken from different wavelength bands and from different object classes. ................................. 27
Figure 3.3. Proportions showing how the data sample was split into training, validation and test sets. ........................................................................................................ 30
Figure 3.4. High-level design of the DBN model and the baseline model, showing the input and output between the steps and the ML techniques used. ............... 31
Figure 3.5. High-level overview of the main classes produced for this thesis, with brief descriptions of their functionalities................................................................. 33
Figure 3.6. The network layers of an instance of the DBNFinetuned class, constructed from a DBN made up of 3 RBMs................................................................. 37
Figure 3.7. Visualization of the activation values of the hidden units from 10 epochs. Larger positive values are whiter and larger negative values are darker. Rows represent data samples, columns represent hidden units. The top rectangle represents initial activations, followed by epoch 1-8 and epoch 100. .................................................................................................................. 42
Figure 3.8. Visualization of 9 features from an RBM. The upper half of the image shows the features without normalization; the bottom half shows the same features with normalization. ................................................................. 43
Figure 4.1. Box plots showing the macro-averaged F1 score from 100 runs each for the baseline model (left), the DBN model before fine-tuning (middle) and the final fine-tuned DBN model (right). ....................................................... 52
Figure 4.2. Box plots showing the F1 score of the STAR class from 100 runs each for the baseline model (left), the DBN model before fine-tuning (middle) and the final fine-tuned DBN model (right). ....................................................... 53
Figure 4.3. Box plots showing the F1 score of the GALAXY class from 100 runs each for the baseline model (left), the DBN model before fine-tuning (middle) and the final fine-tuned DBN model (right) ....................................................... 54
Figure 4.4. Box plots showing the $F_1$ score of the QSO class from 100 runs each for the baseline model (left), the DBN model before fine-tuning (middle) and the final fine-tuned DBN model (right).

Figure 5.1. Relative feature importance of the top 20 features from the DBN model’s Random Forest classifier. The top 10 features are color features, named by the wavelength bands they compare, while the next 10 are unsupervised features from the DBN, named simply by their order of importance, and accompanied by normalized visualizations of their weights.

Figure 5.2. Normalized visualization of the 13 redundant DBN features that were never used in the Random Forest classifier.

Figure 5.3. Normalized visualizations of the DBN features. The top half shows the first 100 (out of 1000) features of the first layer; the bottom half shows all 100 features of the second layer.

Figure 5.4. Samples obtained by sampling from the generative model. The top rows show the actual multi-wavelength training samples that were used to start the Gibbs sampling; the following rows show samples obtained after 100, 500 and 1000 steps of Gibbs sampling between the top two layers.

Figure 5.5. Learning curves for the final model showing the cross-entropy cost for the training and validation sets at each epoch for layer 1 and layer 2, and every 10th epoch for the global fine-tuning.

Figure 5.6. Hyper-parameter evaluation via cross-validation on the classification task.
TABLE OF TABLES

Table 3.1. Early results for the baseline model, expressed in percentages of predictions by the true label, on a sample of 11,711 objects. These do not represent the final results of the baseline model. ................................................................. 23

Table 3.2. The percentages of objects belonging to each class in the sample or population denoted in the top row. The full population consists of all spectroscopically classified objects in the SDSS database; the clean population consists of objects from the full population that meet certain quality criteria. The sizes of the sample and populations are given below the names; for the sample, this is the full sample size before splitting into training and test sets. ....... 25

Table 4.1. Summary of the test results from the experiments. The table shows the means from 100 runs with each model, along with the improvement in the mean of the DBN model over the baseline model. ...................................................... 56

Table 5.1. Comparison of the results of the present work to those of Brescia et al. (2015) after recalculating to the metrics used for the present work. .................. 62
1 INTRODUCTION

The field of Astronomy is as old as our history, and still as important today. While historically used for practical tasks such as navigating the seas and keeping track of the seasonal cycles, its modern-day contributions can be seen in a variety of areas, from direct applications in satellite technology and renewable energy, to technology transfers that have enabled advances in medicine and computer networking (Rosenberg, Russo, Bladon, & Christensen, 2013).

As the fields of optics and engineering advance, astronomers gain access to ever more sophisticated tools for gathering data. This has brought fantastic opportunities to the field, but has also presented it with a new challenge: where the main problem used to be a lack of observational data, the problem is now how to analyze all the new data that has become available (Borne, 2009a).

In parallel to this, there has been a rapid development in Computer Science fields such as Data Analytics and Big Data. The field of Machine Learning (ML) has been promoted from an academic curiosity to a prominent field of practical application. This gives the modern astronomer of today not only access to immense amounts of gathered data, but also the latest Computer Science technologies to analyze this data.

In this thesis, a specific Data Analytics challenge in the field of Astronomy is investigated, and a modern ML approach is suggested as a promising solution. This approach is tested and evaluated in light of its practical results and its suitability to the general area of Astronomy research. The data source used is the Sloan Digital Sky Survey (SDSS) Data Release 12 (DR12) – a recent ground-based astronomical survey covering roughly a third of the sky (Alam et al., 2015).

1.1 Background

With the vast amounts of data gathered by today’s astronomical surveys, the role of ML in the field of Astronomy has grown from interesting to necessary. Unlabeled image data is especially abundant, along with the various photometric parameters that are derived from the image observations, giving rise to the application area of photometric classification – a very relevant application area in Astronomy today (Borne, 2009a; Ivezić, Tyson, et al., 2014; Mickaelian, 2015).

The analysis of unlabeled image data is a problem that is relevant not only in Astronomy. In recent years, this area of research has seen some rapid progress coming from a variety
of domains, from Security to Medicine, with a particularly important research direction being Deep Learning – the ML area dedicated to the study of building and training Deep Neural Networks (Najafabadi et al., 2015).

The focus of this thesis is the cross-disciplinary research area of ML and Astronomy, also referred to as Astroinformatics (Borne, 2009a). Specifically, this thesis investigates a novel approach to the problem of classifying astronomical objects as being either a star, a galaxy or a quasi-stellar object (QSO); the latter term referring to the highly active galactic nuclei believed to be caused by supermassive black holes (Richstone et al., 1998). The traditional way of performing this classification is through the manual inspection of what is called a spectra of the object; a spectra is a very detailed view of the light emitted by an object as a function of its wavelength. From this, it is possible to see whether the source is a point-source (i.e. a star) or an extended source (i.e. actually consisting of multiple objects). It is also possible to detect dips and peaks in the overall curve, known as emission lines and absorption lines in the spectra; these are caused by the elements and chemical reactions of the source and can be used for a highly detailed analysis of its properties (Dawson et al., 2013; Fleisch & Kregenow, 2013). Throughout this thesis, the word object is used to refer to the source of an astronomical observation, and could thus refer to either a singular object, such as a star, or a composite object, such as a galaxy.

While studying the spectra of an object gives access to very precise data for classification, these measurements, known as spectroscopic measurements, are far more time-consuming to obtain than image observations (Djorgovski, Mahabal, Drake, Graham, & Donalek, 2013). The image observations, and the various parameters that can be derived from them, are known as the photometric measurements. While lacking some of the valuable qualities of the spectroscopic measurements, the photometric measurements are far more abundant, so finding ways to extract more detailed knowledge from this data is a highly relevant task in Astronomy (Brescia et al., 2012; Ivezić, Tyson, et al., 2014).

The specific research problem studied in this thesis is the task of photometric classification of stars, galaxies and QSOs. The approach to this is to use an unsupervised ML technique called Deep Belief Networks (DBNs) to extract useful features from astronomical image data. DBN is a Deep Learning technique for finding lower-dimensionality features from high-dimensionality data through unsupervised training of a generative model whose success was originally demonstrated on a computer vision
task (Hinton, Osindero, & Teh, 2006). Considering astronomical image data has been successfully used in manual classification tasks, even by non-professional volunteers, this lends support to the idea that there is unused potential in this data (Lintott et al., 2008; Willett et al., 2013).

1.2 Research Project

1.2.1 Problem Definition
Accurate object type classification in astronomy is currently done mainly by comparing the fit of spectroscopic data to libraries of templates that have been constructed from well-studied sources and simulated models based on current theories (Conroy, 2013). This is a problem because spectroscopic measurements are time-consuming to acquire and only available for a small fraction of the observed sources compared to photometric data. In the SDSS survey, the number of objects with spectroscopic measurements are roughly 1% in number compared to those with photometric measurements (Alam et al., 2015). Better classification results from photometric data would mean far better classification coverage.

However, current research into photometric classification mainly make use of parameters that can be derived according to different rules, giving multiple values for the same parameter and leaving it up to individual researchers which ones to use (Stoughton et al., 2002); this introduces a bias that is hard to measure and makes it more difficult to compare results (Coughlin et al., 2015; De Lucia, Muzzin, & Weinmann, 2014). Coughlin et al. (2015) argue that if all sources can be investigated through the same method then the results can be compared directly and biases can be more readily quantified. Since many photometric parameters are derived from image data, then analyzing the images directly by use of unsupervised methods might provide a step towards a more uniform approach.

1.2.2 Research Question and Scope
The research question investigated in this thesis is the following:

Can unsupervised feature-extraction from multi-wavelength astronomical image data provide a significant advantage in the photometric classification of stars, galaxies and QSOs?
The main intention is to provide an evaluation of the approach as such, so the emphasis is on the *relative improvement* of introducing the unsupervised feature-extraction from image data. The insights gained from this investigation are intended to be useful in the design of future research projects.

1.2.3 Limits of the Scope
While some exploration into different hyper-parameter settings for the DBN was carried out during the project, extensive tuning for optimizing the performance of the final model was not a prioritized task.

It was also outside the scope of this thesis to investigate different classification algorithms in which the unsupervised features can be used, so this is left as a recommendation for future work.

1.2.4 Hypotheses
To answer the research question, a set of hypotheses were formulated. These hypotheses compare the use of two different models: the *baseline model* consisting of a Random Forest classifier trained on 10 color values which are the differences between the magnitudes in 5 different wavelength bands; and the *DBN model* consisting of a Random Forest classifier trained on features that were extracted from astronomical multi-wavelength image data by an unsupervised DBN, in addition to the 10 color values. Both models were trained on the same dataset with the same task of discriminating between the object class labels STAR, GALAXY and QSO.

The first hypothesis states the assumption about the overall performance of the model based on the macro-averaged F$_1$ score which gives equal weight to the precision and recall of each class:

*Hypothesis 1: The DBN model performs better than the baseline model on the three-way classification of STAR / GALAXY / QSO, measured by the macro-averaged F$_1$ score.*

A further 3 hypotheses were formulated on the assumption of an improvement of the F$_1$ score for each individual class:

*Hypothesis 2: The DBN model performs better than the baseline model at classifying the STAR class in the three-way classification of STAR / GALAXY / QSO, measured by the F$_1$ score for the STAR class.*
Hypothesis 3: The DBN model performs better than the baseline model at classifying the GALAXY class in the three-way classification of STAR / GALAXY / QSO, measured by the F$_1$ score for the GALAXY class.

Hypothesis 4: The DBN model performs better than the baseline model at classifying the QSO class in the three-way classification of STAR / GALAXY / QSO, measured by the F$_1$ score for the QSO class.

To statistically test the differences, independent two-sample t-tests were carried out in PSPP\(^1\) with a confidence interval of 95%, tested against a p-value cut-off of .05.

1.2.5 Research Objectives

The following tasks were carried out as part of the research project. Naturally, they overlapped in time, but the list below gives an idea of their dependency order.

1) A literature review was undertaken to understand the domain concepts from the Astronomy field and the technical details of the ML techniques.
2) Suitable features were selected and a data source was found based on previous experiments and the data understanding.
3) Data was downloaded, preprocessed and split into training, validation and test sets.
4) A local development machine and a remote testing machine were prepared for running the experiments.
5) The code for the unsupervised DBN was written and tested, including code for visualizing what the model learned.
6) Different storage and memory management techniques were implemented and tested for handling the Big Data aspect of the project.
7) The code for the classifier model was written and tested, with the ability to output tables of results used for hyper-parameter tuning.
8) The baseline model was implemented, tested and optimized.
9) Multiple experiments were run with the DBN model, followed by hyper-parameter evaluation.
10) The results from the final DBN model were compared to the results of the final baseline model through statistical tests.

\(^1\) PSPP is a statistical analysis application available from https://www.gnu.org/software/pspp/ under the GNU General Public License version 3. PSPP version 0.8.5 was used for this project.
11) Finally, the results were analyzed and compared to state-of-the-art results from previous literature.

1.3 Research Methodologies

1.3.1 Secondary Research
A literature review, which can be found in chapter 2, was carried out to establish the current gaps and formulate a research question based on the analysis of previous works. Further reading was also carried out throughout the development of the experiments to ensure the validity of their implementation. Finally, the reviewed literature was used during the analysis and evaluation of the results.

1.3.2 Primary Research
The first component of primary research for this project consisted of constructive research to develop an unsupervised learning model for feature extraction followed by a supervised classifier to be trained on the extracted features. Quantitative deductive methods were used to analyze the empirical results of the data-driven model during the development and hyper-parameter tuning. The choice of techniques was based both on articles describing their internal qualities and others showing the empirical results of previous experiments. An unsupervised DBN, and the underlying Restricted Boltzmann Machine (RBM) algorithm, were implemented based on a synthesis of theoretical articles and previous practical implementations. As the main programming language, Python was chosen for its popularity in the Astronomy community and the availability of suitable supporting code libraries. Additionally, the TensorFlow framework (Abadi et al., 2016) was used for building a scalable implementation that could handle the Big Data aspect of the Astronomy domain by enabling the use of high-end computing resources. A flexible memory-mapping solution was implemented to also allow the algorithms to be run on lower-end systems and development machines.

The data sample for the project was extracted from the public SDSS database containing data from its large-scale high-quality astronomical survey (Alam et al., 2015). The sample was selected by random sampling stratified on the class labels, from a population that met certain quality and feature-availability criteria. Various Python scripts were written for the data preparation process, including a sample selection script and an automated script for downloading and extracting the image data while keeping disc
space usage to a minimum. The necessary preprocessing of the image data and the splitting of the data into training and test sets were also carried out through Python scripts. SQL was used for retrieving most of the data and for the initial data exploration phase.

The second part of the primary research consisted of a data-driven investigation into the feature-extraction and classification tasks, where the constructive component was applied to a practical research problem in the large-scale Astronomy data domain. Quantitative evaluations of the empirical results from these experiments were carried out in light of the research question by measuring the statistical significance of the difference in performance when using the novel features in addition to a set of baseline features from the literature on a classification task, compared to using only the baseline features. Automated Python scripts were constructed to run the experiments with different configurations a number of times while recording the results in a suitable format to be used in a statistical analysis tool.

Finally, a qualitative evaluation of the novel model was carried out based on visualizations of what had been learnt by the model.

1.4 Thesis Roadmap

This document covers the details of the research project carried out for this thesis. Chapter 2 introduces the reader to the research area with a literature review that covers both the relevant Astronomy concepts and Deep Learning techniques. Current approaches to the research problem are reviewed and analyzed, and gaps in the current research are identified.

Chapter 3 goes into the details of the research design and model development. It starts with an overview of the research project’s phases and then describes how each of these were carried out in practice. A large part of the chapter is dedicated to the software development of the model, including details of the technical implementation of the RBM and DBN techniques. This chapter also describes how the actual experiments were run and what methods were used during the analysis of the results.

After this, chapter 4 presents the final results from running the models; these are set into context of each of the four hypotheses, and the outcomes of the statistical tests are reported.

Following the results, chapter 5 goes into detail of the meaning of those results. An analysis is conducted on what type of information the models carry, followed by a
discussion in the context of the current state-of-the-art. Next, some insight is given into
the training of the model, and finally some additional benefits of the model besides its
main properties are covered.

In chapter 6, the research project is concluded with a brief summary of the project and
its results, followed by some ideas for future works to build on this one. The final section
concludes with the implications of the work in a larger context.
2 LITERATURE REVIEW AND RELATED WORK

2.1 Overview

Because of the cross-disciplinary nature of this project, the literature review is divided into two parts. The first part provides the reader with an introduction to the use of ML in Astronomy research and the nature of available domain data. Following this, state-of-the-art results relevant to the research problem are reviewed, and current gaps in the literature are identified.

The second part focuses on the Deep Learning techniques used in the experiments of this thesis. It gives a brief history on Deep Learning, followed by the theory behind Restricted Boltzmann Machines and Deep Belief Networks. The choice of these techniques is put into context of the insights gained from the first part of the review.

2.2 Machine Learning for Astronomy

Modern astronomical surveys are producing larger and larger amounts of data; see Mickaelian (2015) for a review of the size and scope of modern surveys up until 2015. Currently under construction is the Large Synoptic Survey Telescope (LSST) which is expected to gather image data at a size of roughly 15 terabytes every night for 10 years (Ivezić, Tyson, et al., 2014). With such volumes of data, it is clear that manual inspection is not feasible. Instead, astronomers are turning to modern techniques such as Data Mining and ML in their research (Coughlin et al., 2015; Ivezić, Tyson, et al., 2014).

However, incorporating these technological advances as a standard approach is an ongoing effort since many of the current field experts would be used to a tradition of inspecting each source of interest manually. Djorgovski et al. (2013) describe this as a change of culture within the field and point out that the advanced Computer Science areas are complex fields of their own where astronomers would benefit from exchange and collaborations with experts of those areas. Borne (2009b) also discusses this change of the field. He concludes that for the new techniques to become widely accepted and trusted by astronomers, it is essential that the ML techniques be such that research and analysis can be carried out as a joint effort between the human researcher and the machine algorithm.

In conclusion, it would be preferable to use techniques that can be understood and interpreted by astronomers, rather than black-box techniques that only output a result. A
more interpretable technique would allow the researcher to verify that the result was based on sensible reasoning and could possibly also contribute more to the understanding of the underlying models of our universe than a black-box technique could do.

2.2.1 Astronomy Data

To understand the use of ML in Astronomy, it is necessary to have some basic understanding of the data that is available and how it relates to the studied phenomena. For the present work, three types of astronomical objects are relevant: stars, galaxies and a third type known as QSOs or quasars, which are assumed to be caused by the accretion of gas into a supermassive black hole at the center of a galaxy (Richstone et al., 1998). To detect and measure the stars, galaxies, QSOs and other astronomical phenomena, astronomical surveys mainly make use of two types of observational methods: photometric observations that capture image data (though not limited to visible light) and spectroscopic observations that capture a detailed view of the source’s total radiation as a function of wavelength. Photometric data is generally faster and less expensive to acquire, while spectroscopic measurements require a slower, more expensive, procedure (Djorgovski et al., 2013). For the three-class object type classification, spectroscopic measurements give a more precise answer since they can capture the composition of chemical elements within a source.

However, because of the relatively small amount of spectroscopic measurements available, along with the stronger limitations on their measurable magnitude depth, finding better photometric classification methods is an active area of research (Ball & Brunner, 2010). Additionally, aside from using photometric classification as an actual classification tool, it is a useful tool for selecting targets for the more detailed spectroscopic surveys (Alam et al., 2015).

Among the most widely used features in photometric classification tasks are the relative color values that describe the differences in magnitude between the wavelength bands (Ball & Brunner, 2010). These color values depend on the photometric system used, and are usually referred to by two-letter names based on the wavelength bands they compare, such as B-V or u-g (Bessell, 2005). They should not be confused with the single-color terms used to describe subclasses of stars, such as red giants or white dwarves.
When it comes to using the actual image, the features of interest tend to be some estimations of the object’s spatial structure that is not captured by the spectroscopic data; this aspect has been successfully used for subclass labeling of galaxies, both manually and by use of ML (Banerji et al., 2010; Lintott et al., 2008; Willett et al., 2013).

2.2.2 Data Surveys

For the collection of astronomical data, a multitude of modern sky surveys have been carried out with different science goals in mind. These science goals influence the type of observations made and how suitable their datasets are for different research questions (Djorgovski et al., 2013; Mickaelian, 2015). Listed below are some of the criteria to be taken into consideration when selecting a suitable survey to acquire a dataset from:

- **Wavelength coverage** – Are the studied phenomena within the upper and lower wavelength limits of the study?
- **Sky coverage** – Is the coverage so limited that it would severely bias the research?
- **Depth, resolution and precision** – Are the differences the model is looking for actually detectable by the survey’s instruments?
- **Data products and availability** – Has the survey produced the necessary data products and are they generally available?

For this project, data from the SDSS DR12 database was used; it is a recent survey covering more than a third of the sky, with spectroscopic and photometric measurements available through a well-documented public interface, along with relatively high-resolution image data in 5 different wavelength bands (Alam et al., 2015; Djorgovski et al., 2013). For a detailed description of the SDSS survey, including its sky coverage, wavelength coverage and its photometric and spectroscopic pipelines, the reader is referred to Stoughton et al. (2002) and Alam et al. (2015).

Because of its ease of access and the large amounts of data available through the SDSS database, it has become something of a standard for ML applications in Astronomy (Brescia, Cavuoti, & Longo, 2015). Plenty of published works are available, which was convenient when assessing current approaches and comparing results.
2.2.3 Current Approaches

While manual inspection of data is still used for the golden records in astronomical classification tasks, there are plenty of published works on research using ML techniques. This section specifically reviews some current state-of-the-art approaches that use only photometric parameters and image data as inputs to classification tasks. For a broader review that compares different types of ML approaches to Astronomy problems, the author recommends the work of Ball and Brunner (2010).

The first two experiments covered below show two approaches to separating galaxy subtypes; this is a slightly different classification task from that of the present work, but their methods provide some relevant insights. After that, two experiments are covered whose goals are closer to the research question: one which separates QSOs from stars and unresolved galaxies, and a second that separates stars, galaxies and QSOs.

In an experiment performed by Banerji et al. (2010), classification of galaxy subtypes is carried out by training a Multi-Layer Perceptron to agree with the data labels from the Galaxy Zoo project. The Galaxy Zoo project was a large-scale citizen scientist effort where non-expert users collaboratively classified galaxy subtypes by looking at image composites made from 3 out of the 5 wavelength bands in the SDSS database (Lintott et al., 2008). In the ML experiment by Banerji et al. (2010), three sets of input features are compared: the first uses two color values, g-r and r-i, together with a set of SDSS parameters intended to represent different aspects of the object’s shape; the second set of features uses a different set of shape features that were derived from the image data using an algorithm specifically created for the experiments, but does not use any color values; the third feature set uses all parameters from the previous two feature sets. They found that using the first feature set, their model was able to correctly identify 93% of the Early Type galaxies, 90% of the Spiral galaxies and 82% of the galaxies labeled as belonging to neither class. When adding their own shape features to this, they were able to improve these figures to 97% for both Early Type and Spiral galaxies and 86% for the third category. Using their own shape features alone performed slightly worse on the first two types, with 92% and 89% respectively, and was unable to identify any of the galaxies that were labeled as belonging to neither class. Clearly, the traditional features are able to produce rather good results on their own, but adding the new image-derived shape features seem to have added an additional improvement on top of this. This suggests that spatial features found in the image data can be useful in describing some
relevant characteristics of the object that is not captured by the colors or by the more traditional shape indicators alone. One drawback of their method is that their novel shape features also rely on a set of specific, handcrafted methods. This risks introducing a bias when choosing the finer details of the shape features, and is potentially less flexible than using an unsupervised method where the algorithm decides which details are the most relevant in describing the images.

In another experiment on galaxy subtype identification, also with the Galaxy Zoo (Willett et al., 2013) labels, Dieleman, Willet and Dambre (2015) use a Convolutional Neural Network trained on the same SDSS image composites that were used in the manual classification task. More specifically, their model learns to replicate answers to the questions that were asked to the human participants about the presence of certain visual features in the images. Agreement between the model and the human participants ranged between 96.04% and 70.73% for the different questions. Aside from demonstrating their results, the authors also discuss the benefits of using automatically extracted features rather than handcrafted rules. They further point out the usefulness in being able to visualize what is learned by the model and they demonstrate this feature of their deep architecture. Since their model used only 3 of the 5 available wavelength bands, Dieleman et al. (2015) suggest that future research on the use of the full multi-wavelength image data would be a promising direction.

The following two experiments focus on discriminating between the top-class labels which is also the case of the present work.

In an experiment by Abraham and Philip (2010), the goal is to separate point-source QSOs from other point-source objects with a high degree of confidence by using the magnitude value of the u-band along with 10 color values calculated from differences in magnitudes between SDSS’s 5 wavelength bands. They were able to correctly identify 99.01% of the QSOs in their dataset to a precision of 96.96%. However, their dataset consisted of a very specific sample of the total SDSS dataset which comprised a color space previously shown to contain mostly QSOs; additionally, only objects that had been categorized as unresolved point-sources were selected. The proportion of unique objects that have been spectroscopically classified as QSOs in the total SDSS database is 12.88% while the sample in the reviewed work had 86.14% QSOs. In contrast, galaxies accounted for only 0.65% of the sample, while in the SDSS dataset they account for
64.29%; they were classified only to a precision of 6.77% and a recall of 9.06%. This is not so much a problem in the reviewed experiment where the focus is specifically on the QSO results. However, for the research project of this thesis, the difficulty of generalizing to all classes is a problem that needs to be tackled.

An experiment that fares far better on overall classification accuracy is that of Brescia et al. (2015). Their approach was to use one set of features comprised of the same 10 colors as in the experiment by Abraham and Philip (2010) with the addition of the magnitude value for one of the wavelength bands, and another set of features comprised of the 5 magnitude values from two different sets of SDSS magnitude parameters that have been derived according to two different sets of rules. In their experiments, Brescia et al. (2015) used a 2-layer Multi-Layer Perceptron with Quasi Newton Algorithm to separate between stars, galaxies and QSOs. Their best result on the three-class separation used the two sets of 5 magnitude values and gave a precision of 93.82% for stars, 93.49% for galaxies and 86.90% for QSOs, with recall values of 86.40% for stars, 97.02% for galaxies and 90.49% for QSOs. The results when using the color values instead, with the addition of either the u-band or r-band magnitudes respectively, were slightly lower: for stars the precisions were 90.21% and 89.93% with recalls of 82.57% and 82.27%, for galaxies the precisions were 88.00% and 88.03% with recalls of 92.69% and 92.64%, and finally for QSOs the precisions were 85.56% and 85.60% with recall values of 87.83% and 87.77%.

A downside to the architecture used in the experiments by Brescia et al. (2015) is that it is a black-box method. This makes it more difficult to draw conclusions about what it has learned which could otherwise help improve the understanding of the studied phenomena.

Indeed, one aspect of the model that would have been interesting to investigate is the aspect of how the two feature sets influenced what was learned. A conclusion drawn by Brescia et al. (2015) is that their results on a two-class task of separating QSOs from stars confirm previous results regarding the better performance of using color values rather than the magnitudes. However, on the three-class separation of stars, galaxies and QSOs, their results show the opposite to be true, though the implications of this is not discussed in the article.
Since magnitude values are affected by both the actual strength of the source and the distance to the measured objects, using these values directly might unintentionally train the model to fit to this hidden composite feature. While this may be a relevant feature due to cosmological reasons, it also carries the risk of overfitting to a bias in the target selection and the instrument quality of the survey. Color values, on the other hand, are less sensitive to distance because of their relative nature (although complications still arise due to the influence of redshift), and they are considered to carry information that is indicative of the physical qualities of the measured object (Bessell, 2005); hence, it may be possible that they provide a slightly less biased means of classification.

2.2.4 Summary
From the previous literature, there is strong evidence to suggest that photometric data carries a great deal of information about the object class of astronomical sources. Although perfect mapping to the spectroscopic classifications have not yet been achieved, it is clear that Deep Learning has come a long way towards this goal.

All the state-of-the-art experiments covered above make use of the multi-wavelength aspect of the data in some way. In fact, state-of-the-art results on the SDSS dataset have been possible to achieve using only the color features calculated from the differences in magnitude between its 5 wavelength bands. The use of such color features is a well-studied area of photometric classification, making them very suitable for use as baseline features in experiments that seek to investigate the usefulness of other types of features.

From the first two experiments reviewed, there is evidence to suggest that there is additional information in the image data, not captured by the color values or the shape features in the SDSS dataset, that would be useful to astronomical classification tasks.

2.3 Gaps
Accurate classification still relies heavily on spectroscopic measurements which are far less abundant than image data and the derived photometric parameters, but the use of ML techniques are starting to challenge the need for spectroscopy in some cases.

However, while good results have been achieved using only the traditional photometric parameters, less attention has been given to exploring the usefulness of unsupervised feature extraction from the image data. Most often, features are derived through carefully handcrafted rules, and Hoyle (2016) argues that this approach introduces a bias that can
be avoided by instead using an ML algorithm for feature extraction. Even the well-performing color values fall under this bias category since they originate from the magnitude values which are secondary parameters derived from the actual image data in way that is not consistent across all sources (Stoughton et al., 2002). Using unsupervised feature-extraction from the image data brings the algorithms one step closer to the actual measurements and could potentially unlock latent features that have not yet been well-captured by the currently used features.

Finally, a problem that arises when using advanced ML techniques for classification tasks is that it tends to be rather difficult to gain insight into what they base their decisions on. This makes such methods less likely to be accepted by astronomers who may wish to verify that the algorithms actually learned something sensible, and whose end-goals may be to gain further insights into why a certain model performs well on the studied phenomena. A possible remedy to this is to favor interpretative models such as generative models and models that use layer-wise training where the features of each layer can be visualized. Dieleman et al. (2015) successfully used such a method on SDSS image data for galaxy subtype classification, and it might not be unreasonable to consider such an approach for the separation of stars, galaxies and QSOs as well.

2.4 Deep Learning and its Suitability to the Research Problem

2.4.1 A Brief Introduction to Deep Learning

Deep Learning is an area of ML that covers the use of multi-layered non-linear models that can be either supervised, semi-supervised or unsupervised (Deng, 2014). It builds on previous research into Neural Networks which is a technique inspired by our understanding of the biological learning process in humans and other animals. The breakthrough that led to a renewed surge of interest into Neural Networks in the early 2000s came from a combination of the rapid increase in processing power at a lower price scale, along with the work of Hinton, Osindero and Ten (2006) who proposed a more efficient way of training Deep Neural Networks than the previous standard of using backpropagation (Deng, 2014). Their approach was to train the networks layer-by-layer rather than globally, so that increasingly sophisticated features could be found at each level. This not only improved the speed of training and the final result – it also made it possible to leverage large amounts of unlabeled data. Their motivation for using such an approach was that it should be possible to find a clearer link between what causes the
image (i.e. the actual object being imaged) and its class label than there would be between individual pixel values and the class label of the object (Hinton, 2010b); in practice, such a model learns the underlying features that cause the pixel values to take on certain distributions.

2.4.2 Restricted Boltzmann Machines and Deep Belief Networks
To achieve the greedy layer-wise training described above, Hinton et al. (2006) constructed a deep non-linear model from a stack of two-layer Restricted Boltzmann Machines (RBMs) followed by a final discriminative softmax layer. Each RBM is a network where all the units of one layer are connected to all the units of the next, with no connections between units within the same layer. The weights are symmetrically tied in both directions: in a two-layer RBM, the weight matrix from the visible (input) layer to the hidden layer is the transpose of the weight matrix from the hidden layer to the visible layer. In addition to the weights, the two layers have their own set of bias units corresponding to the connected units of that layer. The weights and biases are trained using a learning technique called Contrastive Divergence (CD) which was introduced by Hinton (2002) as an efficient approximation of maximum likelihood learning.

Hinton et al. (2006) showed that CD works well for training a deep model layer-by-layer, as an alternative to using global training via backpropagation, thereby avoiding the vanishing gradient problem where the estimated gradients grow smaller as they are distributed back through the layers towards the input layer. After the deep model has been trained layer-by-layer, a final phase of global fine-tuning is performed by a contrastive version of the wake-sleep algorithm (Hinton, Dayan, Frey, & Neal, 1995), also referred to as the up-down algorithm. This type of deep model is referred to in the literature as a Deep Belief Network (DBN); however, the term is used somewhat ambiguously as it sometimes implies that the network includes a supervised softmax layer while at other times it refers to the unsupervised version, and yet other times it is used for a Multi-Layer Perceptron that has been pre-trained by CD and then fine-tuned by traditional backpropagation (Deng, 2014).

In this work, DBN refers to an unsupervised model trained by CD and fine-tuned in an unsupervised manner through the contrastive version of the wake-sleep algorithm.
2.4.3 Suitability to the Research Question and the Field of Astronomy

As discussed in section 2.2, it would be highly desirable to employ ML techniques that can be understood and interpreted by field experts in Astronomy rather than black-box techniques that only outputs the final results. This aspect has influenced the choice of technique for the present work: the choice was made to use a generative technique that can be visually inspected and interpreted by subject experts. The RBM and DBN are both generative models which can be used to generate visual samples that show what type of data the model has learnt to recognize. Both models also provide a straightforward way to visualize what sort of structures each unit is sensitive to in the images.

Furthermore, it is worth mentioning that unsupervised layer-wise training (which made up most of the training time for the model used in this thesis) has been shown to be useful in Transfer Learning; this has been studied under the area of Representation Learning where a set of representations learned by deep unsupervised layer-by-layer training have been successfully applied to accomplish a set of different classification tasks within the same domain (Bengio, Courville, & Vincent, 2013). This further increases the usefulness of such techniques for Astronomy research, where the studied population often remains the same for a variety of different research tasks. It also prevents the model from being locked-in to the current understanding of the labels we assign to the different phenomena. Current definitions go through re-evaluations, and new definitions appear, as our theories and measuring techniques improve (Djorgovski et al., 2013).

Finally, RBMs and DBNs can also be seen as dimensionality reduction techniques where the full-pixel data is compressed into a more informationally dense representation; this is an aspect that will only grow in importance along with the size of new survey data. Combined with the reasoning above on Transfer Learning, a library of such representations could potentially provide a new set of features for further Astronomy research on a variety of tasks, in addition to those features currently available in the SDSS database (or other surveys). While other dimensionality reduction techniques could also be considered for this, deep non-linear models have been shown to be a good candidate for tasks where linear models such as Principal Component Analysis (PCA) induce too strict assumptions about the underlying distribution of the signal in the data compared to the noise (Bengio, 2012).
2.5 Summary

In this literature review, an introduction to ML in the field of Astronomy was given along with a basic understanding of the data involved. Current state-of-the-art approaches to photometric classification tasks were assessed in light of the research question of the present work. It was found that photometric-only classification can reach high levels of accuracy and that the well-studied color features make a suitable set of baseline features when assessing the additional usefulness of novel features.

Gaps in the current research were identified, with the specific scientific requirements of the field in focus. Specifically, attention was given to unsupervised learning and interpretability of the models, and Deep Learning was proposed as a promising solution.

Finally, a brief introduction was given on the theory of Restricted Boltzmann Machines and Deep Belief Networks, and the rationality behind applying these techniques to the research problem was covered.
3 RESEARCH DESIGN AND IMPLEMENTATION

3.1 Overview

To answer the research question, defined in section 1.2.2, regarding the advantage of using unsupervised feature-extraction from astronomical image data in classification tasks, an experiment was designed where the results of two competing models could be compared and the difference in results could be tested for statistical significance.

The first model, the *baseline model*, uses the color values that, based on the literature review, were found to provide a suitable set of baseline features for the classification task. This baseline model was run at an early stage of the project on a preliminary data sample to confirm that it provided a sensible approach and to get an indication of the baseline results to improve upon.

The second model, the *DBN model*, uses the novel features extracted by the unsupervised DBN, in addition to the baseline features, to allow evaluation of whether these new features provide significant advantage beyond the baseline or not.

Both models use a Random Forest classifier to assign class labels. For the baseline model, this is the entire model; for the DBN model, the Random Forest classifier is the final stage where the DBN features and the baseline features are used as inputs.

The models were compared by a one-tailed independent two-sample t-test on the macro-averaged F1 score from 100 runs each from the baseline model and the DBN model. The test set for these runs was a reserved sample that was not used in the training or cross-validation of the models.

Finally, to enable a deeper analysis of the results and demonstrate the interpretability of the model, the individual contributions of the top 20 features were reported, along with visualizations to show what the DBN features of this list represent. The generative capabilities of the model were also demonstrated by sampling from its joint probability distribution.
Overall, the research project can be divided into four phases with a set of high-level objectives:

1) *Data preparation phase.*
   - Identifying suitable baseline features.
   - Selecting and acquiring the dataset.
   - Data cleaning and preprocessing.

2) *Implementation phase.*
   - Designing and developing the software for the experiments.
   - Testing the correctness of the software implementations.

3) *Experiment phase.*
   - Running the experiments with different hyper-parameter settings.
   - Evaluating results to find the best performing models.
   - Running the experiment with the best performing models on a reserved test set.
   - Carrying out statistical tests on the results.

4) *Analysis phase.*
   - Setting the results into context of the research question and previous literature.
   - Visualizing what the model has learnt.

The rest of this chapter is split into sections according to these phases. Section 3.2 describes the data preparation phase. This includes selecting a suitable dataset and baseline features to compare performance against. It also describes how the data sample was acquired and how the image data was preprocessed.

After that, section 3.3 describes the software design and implementation of the DBN model. An overview of the overall model design is given, followed by the design of the actual software implementation. After this, the memory management aspect of the implementation is covered. Next, the technical details of the RBM and DBN implementations are given, including explanations of the theoretical models they represent. After also covering the implementation of the Random Forest classifier, the final part of this section describes the software development process for the project.
Section 3.4 describes how the actual experiments were run, and how the optimal hyperparameters were chosen. Details are also given on the statistical tests that were carried out, and the reasoning behind the choice of metrics.

Finally, section 3.5 explains how the actual results were analyzed and how the quality of the model was investigated.

Some limitations of the methodologies used for this project are identified in section 3.6, before concluding in section 3.7 with a brief summary.

### 3.2 Data Preparation

#### 3.2.1 Baseline Features

The baseline features are a set of features whose usefulness has previously been tested and reported in the literature. As discussed in chapter 2, a common set of features are the color values which describe the difference in magnitude between the wavelength bands. Color values have been known for a long time to discriminate well between different object classes, and even subclasses (Bessell, 2005). Depending on the experiment, and the magnitudes available in the dataset, one or more such color values may be used.

For the present work, 10 colors were derived from the 5 magnitudes; this does introduce some redundancy in the values, but previous experiments have shown that this provides better empirical results in modern ML algorithms than what using a non-overlapping subset of them does (Brescia et al., 2015). The magnitude parameters used are the variants from the SDSS database that have already been corrected for extinction\(^2\). The color features were named after the two wavelength bands they compare, so the magnitude in the u-band minus the magnitude in the g-band becomes feature \(ug\). The 10 baseline features are then: \(ug, ur, ui, uz, gr, gi, gz, ri, rz, iz\).

Since these features were to provide the baseline results, it was important to inspect their usefulness early on with the experiment setup of the present work. This early test seemed

\(^2\) Extinction here refers to the absorption of some of the light due to the interstellar medium that it passes through on its way to us. For details on how these corrections were made to the SDSS parameters with the prefix \textit{dered}, see: http://www.sdss.org/dr12/algorithms/magnitudes/#extinction

For details on calibrations with regards to atmospheric extinction and telescopic calibrations, which are carried out for all SDSS magnitude parameters see: http://www.sdss.org/dr12/algorithms/Fluxcal/
to confirm the conclusion from previous literature about their suitability to the classification task and verified the validity of using the color values as the baseline features. It also provided an idea of what the baseline results were that the novel features were intended to improve on. Table 3.1 shows these early results of the baseline model’s ability to predict the true labels, expressed in percentages of predictions based on the true label, on a sample of 11,711 objects.

<table>
<thead>
<tr>
<th>True labels.</th>
<th>Predicted labels.</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAR</td>
<td>86.67%</td>
</tr>
<tr>
<td></td>
<td>11.03%</td>
</tr>
<tr>
<td></td>
<td>2.31%</td>
</tr>
<tr>
<td>GALAXY</td>
<td>8.12%</td>
</tr>
<tr>
<td></td>
<td>87.23%</td>
</tr>
<tr>
<td></td>
<td>4.64%</td>
</tr>
<tr>
<td>QSO</td>
<td>9.94%</td>
</tr>
<tr>
<td></td>
<td>30.94%</td>
</tr>
<tr>
<td></td>
<td>59.12%</td>
</tr>
</tbody>
</table>

Table 3.1. Early results for the baseline model, expressed in percentages of predictions by the true label, on a sample of 11,711 objects. These do not represent the final results of the baseline model.

3.2.2 Acquiring the Dataset

The experiments of this work were carried out on a sample from the SDSS dataset, selected and extracted specifically for this piece of work. This section describes how the sample was selected and turned into a usable quantity.

To understand the selection criteria of the sample, the reader is reminded that the goal of the model is to assign class labels to astronomical objects based on a set of photometric baseline features and a set of features extracted from the image data. The assigned class labels should, as best as possible, agree with the labels that have been assigned through the SDSS spectroscopic pipeline. Hence, a random sample was selected from a population that met certain photometric and spectroscopic quality criteria, stratified on their spectroscopic classification labels. These quality criteria were based only on predefined SDSS flags, and were mainly used to sort out observations that had serious measuring problems, objects that were duplicates and observations that measured background sky levels rather than actual objects. No objects were discarded based on additional assessment of the image data or parameter values; this decision was made with the intention of testing the unsupervised feature-extraction on images of more realistic quality expectations.
The following selection criteria were used to find the total population of suitable objects:

1) The object must have received a classification label based on a valid\textsuperscript{3} spectroscopic observation.

2) The object must have valid\textsuperscript{4} photometric data marked as clean\textsuperscript{5} by the SDSS photometric pipeline.

3) Only the primary\textsuperscript{6} photometric and spectroscopic observations of the objects were used.

This left a “clean” population of 3,140,923 individual objects from the full SDSS spectroscopic population of 3,537,411. From this, a sample of 10,000 objects was selected by random sampling stratified on the object class labels. The total number of images obtained from this sample was the sample size multiplied by 4 since all images were used in 4 rotations as explained further on in section 3.2.3.

Due to resource limitations, this is a rather small sample compared to the total population. However, since the research question is concerned with testing the approach rather than drawing conclusions about a population of unknown astronomical objects, this compromise in size was considered acceptable. To put the sample size into the context of training an unsupervised model, it should be mentioned that Hinton (2010b) discusses the training of unsupervised generative models regarding the constraint that the inputs place on this type of model, which is concluded to lead to a lower demand on the sample size of the training data; an example is given where a training sample of $10^6$ unlabeled images is deemed sufficient for training an unsupervised generative model with $10^8$ parameters. In comparison, the unsupervised model in the present work consists of at most $4.2 \times 10^6$ parameters with a training sample size of $1.4 \times 10^4$ images (after adding rotational variations and splitting into training and test sets). Future work that

---

\textsuperscript{3} The spectroscopic measurement was considered valid if none of the following zWarning flags were set: UNPLUGGED, BAD\_TARGET, NODATA. See http://www.sdss.org/dr12/algorithms/bitmasks/#ZWARNING

\textsuperscript{4} The photometry was considered valid only if the first bit of the calibStatus flag was set for all five wavelength bands. See http://www.sdss.org/dr12/algorithms/photo\_flags/#calibstatus

\textsuperscript{5} The photometric data was considered to be of sufficient quality if the SDSS ’clean’ flag was set to 1. See http://www.sdss.org/dr12/algorithms/photo\_flags\_recommend/

\textsuperscript{6} There is some overlap in the observations of the SDSS, further explained here: http://www.sdss.org/dr12/algorithms/resolve/

To avoid duplicates, only the best observation for each object was selected by using the database view SpecPhoto which maps the primary spectroscopic and primary photometric observations for each object. See http://skyserver.sdss.org/dr12/en/help/browser/browser.aspx#&history=description+SpecPhoto+V
replicate the experiments of this thesis with a larger sample size would be a valuable addition.

To assess the representativeness of the sample used for this work, Table 3.2 shows the percentages of each object class in the sample compared to the clean population (meeting the selection criteria discussed previously) and the total population of all spectroscopically classified objects in SDSS DR12 (irrespective of whether they met the selection criteria or not).

<table>
<thead>
<tr>
<th>Class label</th>
<th>Sample n = 10,000</th>
<th>Clean population n = 3,140,923</th>
<th>Full population n = 3,537,411</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAR</td>
<td>23.97%</td>
<td>23.97 %</td>
<td>22.83 %</td>
</tr>
<tr>
<td>GALAXY</td>
<td>62.10%</td>
<td>62.10 %</td>
<td>64.29 %</td>
</tr>
<tr>
<td>QSO</td>
<td>13.93%</td>
<td>13.93 %</td>
<td>12.88 %</td>
</tr>
</tbody>
</table>

Table 3.2. The percentages of objects belonging to each class in the sample or population denoted in the top row. The full population consists of all spectroscopically classified objects in the SDSS database; the clean population consists of objects from the full population that meet certain quality criteria. The sizes of the sample and populations are given below the names; for the sample, this is the full sample size before splitting into training and test sets.

To retrieve the data in a usable format, an online interface called CasJobs\(^7\) was used. CasJobs provides access to the SDSS database online through an internal flavor of SQL. It allows users to specify their own queries and store the results in their own tables on the CasJobs server. The data can be downloaded as csv files, manipulated locally, and uploaded back to the server. This functionality was used to first download only the internal object IDs along with the class labels for the stratified sample selection before requesting more detailed parameters on the chosen sample. Through this procedure, new samples could easily be extracted if additional computing resources would allow. Figure 3.1 shows a screenshot of the CasJobs interface.

\(^7\) http://skyserver.sdss.org/Casjobs/
The SDSS image data, on the other hand, is available in a different way from the parameter data. It can be downloaded from the DR12 Science Archive Server\(^8\) via a link constructed from a set of parameters from the CasJobs database. This lets the user download a file containing various details on all the objects of a single *field*\(^9\) which is the SDSS term for a patch measuring 10 by 13 arc-minutes of the sky, captured in one continuous camera run.

Each field is stored in the Flexible Image Transport System\(^{10}\) (FITS) format, a scientific data format commonly used in Astronomy. These contain descriptive header data and one or more matrices of pixel-level measurements. The SDSS fields use an internal non-standard variant of this format, so a special tool called read_atlas_image\(^{11}\) was needed to extract the individual images in each wavelength band. The output from this extraction process is in the standard FITS format that can be loaded through the Astropy Python library (The Astropy Collaboration et al., 2013).

### 3.2.3 Image Preprocessing

The main goal of the image preprocessing, or rather the DBN input data preprocessing, was to extract the data from the astronomical data files and turn this into a format that could be used by the DBN model. While the data files are not conventional image files, it helps to think of them in this way since they represent measurements in 2-dimensional

---

\(^8\) [http://data.sdss3.org/](http://data.sdss3.org/)

\(^9\) For more details on how a field is captured, see [http://www.sdss.org/dr12/imaging/imaging_basics/](http://www.sdss.org/dr12/imaging/imaging_basics/)


\(^11\) A link to the read_atlas_image tool and a brief description of its usage can be found here: [http://www.sdss.org/dr12/imaging/images/](http://www.sdss.org/dr12/imaging/images/)
space as captured by the survey’s CCD cameras. The measurements in each wavelength band are captured by cameras using a specific filter for each band (Gunn et al., 1998). To gain a better understanding for these measurements, a tool called SAOImage DS9\(^{12}\) was used; this is a scientific tool specifically developed for reading and manipulating astronomical image data.

The final output from the preprocessing pipeline is a set of binary NumPy\(^{13}\) files in the npy format, each holding a flat array of the pixel values from all 5 wavelength bands for a single object. To verify that these had been stored correctly, and to inspect the effects of the preprocessing steps, a Python script was constructed that loads these files and converts them into per-wavelength PNG\(^ {14}\) image files for manual inspection. Some examples of what these reconstructed images look like are given in Figure 3.2.

![Figure 3.2. Examples of image data after preprocessing, reconstructed into PNG images for visual confirmation of their quality. The images are taken from different wavelength bands and from different object classes.](http://www.libpng.org/pub/png/)

The preprocessing script starts by loading the measurements in the FITS files into NumPy matrices. These measurements represent per-pixel count data corresponding to what the instruments measured, with an additional base value of 1000 added to each pixel value to avoid storing negative values. This base value was subtracted, leaving the correct count data. The count data from the files has already been corrected for extinction but are otherwise uncalibrated. Since the relative strength of each wavelength band is already captured by the baseline color features, the goal of the image preprocessing was to bring out any structure present in each band; as such, no further calibration was considered necessary. If future work investigates using the image features without the

\(^{12}\) http://ds9.si.edu/site/Home.html
\(^{13}\) NumPy is a Python library for scientific computations, found here: http://www.numpy.org/
\(^{14}\) http://www.libpng.org/pub/png/
colors, then it would be recommended to perform calibration based on the more involved process described by Smith et al. (2002).

The next step was the noise-reduction. This step was added at a rather late time during the development when it became apparent that the DBN was overfitting to the noise in the data, especially when it came to the u-band and the z-band where the signal-to-noise level was rather poor. Thanks to the layer-wise training of the DBN, it was possible to get a clear view of what each layer learned through visualizations; this was used to discover the problems mentioned here and to later verify they had been fixed. Only the background was de-noised by simply zeroing out any pixel values below twice the absolute minimum value. If there is no noise in an image, this value should be zero after subtracting the SDSS base value as explained in the previous paragraph. Given this, the assumption was made that any value below zero represented noise in the measurements and would make a suitable indicator of the noise-level. Manual inspection of the data when reconstructed into PNG image files confirmed that this seemed a reasonable assumption for the dataset.

After the noise-reduction step, the image data for the current object was normalized on a per-wavelength basis. As previously mentioned, the differences in magnitude between the wavelength bands is already captured by the color values, so each wavelength band was treated individually to bring out the structure therein. This step also ensured that the values were in the range expected by the input layer of the DBN. Additionally, it helped when training the DBN to find spatial features based on relative pixel intensities compared to if it would also need to take into account the less relevant aspect of apparent magnitudes.

Next, all wavelength images for a single object were centered and cropped by the same bounds. It turned out that the SDSS algorithm that decides which pixels contain relevant measurements to an object tends to assign a rather large area compared to what would seem to be the object of interest to a non-expert human eye. This area is used by the SDSS pipeline when cutting out the individual objects from the field images, so some images would appear to show mostly a large amount of empty space around a tiny object. While this large area around some objects is necessary for the SDSS pipeline when obtaining correct measurements of an object’s projected light, it was not helpful to the DBN when learning to identify representative generalized features. Before proper
cropping and centering was achieved, it was found that the DBN was overfitting by learning uninformative features such as where in the image a certain object appeared to be located; such features help the DBN to recognize specific objects in the training set but are very misleading when classifying unknown objects.

To achieve the centering and cropping, the normalized g, r and i bands were used in the calculations; if nothing at all was found in these three bands, the remaining, more noise-prone, u and z bands were used instead. First, the center-point was considered to be the median row-index and median column-index of all pixels with a normalized value of 0.5 or higher. After that had been found, the cropping bounds were assigned by stepping in towards the center, column by column from the left and right sides, and row by row from the top and bottom sides, until any normalized value of 0.1 or above was found for each of these directions. This procedure was repeated for each of the involved wavelength bands (according to the same logic as for the centering), and the 4 medians were selected. All wavelength images were then cropped according to the highest up-down and left-right bounds and padded as needed to keep the object centered in the images.

Finally, the images were rescaled to 20 by 20 pixels in each wavelength band. These measurements were chosen based on histograms of the post-cropping bounds for the full dataset, so that the vast majority of the images would be scaled up rather than down.

After the preprocessing steps were completed, each image was rotated in steps of 90 degrees to produce four versions of the same image so the model would learn to recognize objects regardless of their apparent orientation.

3.2.4 Splitting the Dataset into Training, Validation and Test Sets

When training and testing ML algorithms, care must be taken not to overfit to irrelevant features or noise in the data. This risk can be mitigated by separating the data into a training set which the algorithm bases its learning on, and a validation set which is used to monitor the learning; while the algorithm would be expected to become better and better at fitting its parameters to the data it has been shown (i.e. the training set), it would be expected to improve its fit to previously unseen data (i.e. the validation set) only as long as it maintains a sufficient amount of generalization. In this way, by monitoring the performance on the validation set, the optimal hyper-parameters can be selected, such as the number of units and the training time of the model. Finally, it is also often recommended to have a third set that is not used in training or cross-validation; this is
called the test set and is intended to be used to obtain an unbiased measure of the performance of the final model (Kelleher, Mac Namee, & D’Arcy, 2015a).

For the research project of this thesis, the splitting was a bit more complex since the full DBN model was actually made up of two chained models. To avoid biasing the cross-validation results of the second model by using data that the DBN-features fitted well to, the first model had its own training and validation sets as shown in Figure 3.3. This extra splitting was perhaps a bit on the strict side, and more data could have been available at less disk space if the extra splitting was avoided.

![Figure 3.3. Proportions showing how the data sample was split into training, validation and test sets.](image)

For the DBN training, the validation set was used when inspecting the curves of how the cost estimates changed during training. A test set was not used for the DBN part since the real metric of interest was only available from the result of the Random Forest classifier. For the Random Forest classifier, the validation sets were used for selecting the optimal number of trees for each model. Finally, the reserved test set was used when testing the performance of both the baseline model and the DBN model and comparing them in the statistical tests.

### 3.3 Software Implementation

#### 3.3.1 Model Design

The software implementation of the DBN model consists of two parts: the first part is the unsupervised DBN for feature extraction from image data, followed by the second
part where a Random Forest classifier is trained to map the extracted features, along with the 10 baseline features, to spectroscopically determined class labels. For the baseline model, the Random Forest classifier is the entire model. Figure 3.4 gives an overview of the high-level design of the two models.

![Image of high-level design of DBN model and baseline model]

*Figure 3.4. High-level design of the DBN model and the baseline model, showing the input and output between the steps and the ML techniques used.*

The DBN technique was chosen mainly for two reasons, further discussed in section 2.4. Firstly, it is a strong yet interpretable technique, and secondly, it uses unsupervised training. The first point means it can be understood well by human astronomers which makes it more likely to be accepted as a viable research tool; it also enables it to contribute more to the understanding of the results. The second point makes it less biased to prior assumptions and means that it can potentially leverage the vast amounts of unlabeled data that are available.

The choice of a Random Forest classifier was based on the need for a classification technique that could be trained, tested and compared in a straightforward and fair way. Since the only hyper-parameter that was experimented with was the number of trees, it was easy to optimize the performance of the classifier, which reduced the risk of biasing the training towards the author’s model by spending more time and effort on its classification performance.
Another reason for selecting the Random Forest technique was because of its ability to report the level of contribution that each feature brought to the model. This helped when interpreting the final results in light of the research question regarding the usefulness of the unsupervised features. This method was previously used by Hoyle et al. (2015) who applied a similar technique called Adaboost to measure feature importance for photometric redshift estimation.

3.3.2 Software Design

The software code for this project was written in Python 2.7, a choice that was made based on its popularity in the Astronomy community and the availability of relevant code libraries. The author hopes that this will increase the potential usefulness of the code beyond this project. The final state of the code can be found on GitHub under user AnnikaLindh’s project DBNTensorFlow\(^\text{15}\).

Another design decision was to use the TensorFlow framework for the DBN implementation. TensorFlow was developed by the Google Brain Team for internal use on their Deep Learning projects, but was recently released as open source (Abadi et al., 2016). It implements some common functionality for Deep Learning techniques along with a convenient interface for monitoring the training of the model. The main motivation for using TensorFlow in this work was because of its built-in scalability to multicore and GPU architectures so that the performance of the model could easily scale with the available hardware. This aspect seemed suitable for the Astronomy domain where Big Data is a big concern.

The main piece of software development for this project was the implementation of the unsupervised learning techniques. Since fully functioning implementations of these techniques weren’t readily available, they had to be implemented based on research articles describing them. Some code and tutorials were available online, but none of them focused on the unsupervised architecture of interest for this project.

A good starting point however was found in an open source Deep Learning library built on the TensorFlow framework. The starting state of the code can be found on GitHub under user blackecho’s project Deep-Learning-TensorFlow commit 039592b\(^\text{16}\). The TensorFlow usage remains mostly the same in the present work, while the RBM and

\(^{15}\) https://github.com/AnnikaLindh/DBNTensorFlow

\(^{16}\) https://github.com/blackecho/Deep-Learning-TensorFlow/tree/039592b580e28a704f0a1e713bb48607e71ead36
DBN implementations have been reworked and the supervised fine-tuning phase has been replaced by a new class that uses a completely different, unsupervised algorithm for fine-tuning.

An overview of the main source code classes produced for this thesis is given in Figure 3.5 with notes on what each class is responsible for. In addition to these classes, separate scripts were written that launches the creation of instances of these classes, the loading of the datasets and the training.

**Figure 3.5. High-level overview of the main classes produced for this thesis, with brief descriptions of their functionalities.**

For performance reasons, training was performed on mini-batches rather than on one sample at a time. This lets the computer leverage vector and matrix operations for simultaneous calculations on a whole batch rather than the less optimal way of going through each single sample in a loop. To avoid influencing the effect of the learning rate by the batch size, the updates are averaged over the current mini-batch. Hinton (2010a)
recommends keeping the size of a mini-batch rather small, between 10 and 100 samples. For this work, a mini-batch size of 10 was found to work well through empirical tests.

The following parts of this section first go into the details of the memory management in section 3.3.3, followed by the implementation of the RBM in section 3.3.4, an explanation of the DBN in section 3.3.5 and details on the unsupervised fine-tuning of the DBN in section 3.3.6. For the theory on the RBM and DBN techniques, the reader is referred to section 2.4 of chapter 2.

The final two parts of this section cover the Random Forest classifier and the software development process.

3.3.3 Memory Management
With large datasets, it is not feasible for a consumer computer to load all data into memory at once. On the other hand, if the model is run on a more powerful system, it would be desirable to make use of its resources. With this in mind, a scalable solution to the memory management was implemented.

The adopted approach uses the batch-feeding paradigm enabled in TensorFlow, where a single batch of data is made visible to the TensorFlow pipeline each time a full step is run. To further decouple the batching, the author’s model accepts a dataset as an instance of any class that implements the functionality of a provided data handler base class. For the present work, a subclass is implemented that loads a dataset from binary NumPy files through a flexible memory solution: by setting a flag, the caller can switch between loading the full dataset into memory or using NumPy’s on-disk memory mapping functionality; a further flag specifies whether to load the current batch into memory or refer to its memory mapped representation. Shuffling of the training order is done by shuffling a list of indices rather than the actual data.

3.3.4 RBM Implementation
The implementation of the RBM class follows the methods described in Hinton et al. (2006) and Hinton (2010a). Each RBM has a single weight matrix and two bias vectors. RBMs work under the assumption that there is no correlation between the states within the same layer, so all states of a layer can be updated simultaneously based on the states of the other layer. When going from the visible layer to the hidden layer, the input is multiplied by the weight matrix and the hidden biases are added to the result; when going
from the hidden layer to the visible, the hidden representation is multiplied by the transpose of the weight matrix and the visible biases are added to the result. Hence, the weights of the RBM are symmetrical while the biases are individual to for each layer.

Depending on the type of units that are used in the network, these results may be passed through a non-linear activation function, which may also be followed by stochastic sampling of the states. The implementation for this work supports four different unit types: Gaussian units with linear activations followed by sampling from a truncated normal distribution; logistic units, also known as truncated exponential units (Larochelle, Bengio, Louradour, & Lamblin, 2009), with sigmoidal activations and no sampling; binary stochastic units, with sigmoidal activations and stochastic sampling of values 1 and 0, using the activations as probabilities; and Rectified Linear Units (Nair & Hinton, 2010) with a minimum value of zero and linear activations from all positive values. In the final model for this work, logistic units were used for the visible layer of the first RBM, while binary stochastic units were used for all other layers. This combination was found to provide stable training with steadily improving weights.

To understand how the learning happens, it helps to keep in mind that an RBM is a generative model. It actually learns the ideal weights to be able to generate data samples that resemble those it has seen. The way it does this is by the CD algorithm (Hinton, 2002). The intuition behind CD is that it changes the weights to make the model more likely to generate a sample similar to the current training sample, compared to what it currently generates from a state that is closer to the model’s equilibrium state (which is based on what it has learned so far). To implement the learning rule that finds the proper gradient and changes the weights accordingly, the code first computes the correlation between the visible state for the current sample and the hidden state when sampled from that visible state; this is called the positive association. After that, the model is brought to a state closer to its equilibrium state by performing a number of steps of sampling new visible states from the current hidden states and then sampling new hidden states from those visible states. Each such pair of steps (hidden to visible and visible to hidden) is referred to as one full step of Gibbs sampling. Gibbs sampling is a Markov Chain Monte Carlo method where each new state in the chain is sampled based on the connected states and the associated conditional probability (Kelleher, Mac Namee, & D’Arcy, 2015b). CD is sometimes referred to as CD-k where the k denotes how many
steps of Gibbs sampling are performed. For the present work, the standard value of k=1 was used, but the implementation allows the user to change this value by setting a flag.

After performing alternating Gibbs sampling for k steps, the *negative association* is computed as the correlation between the final visible and hidden states. The change in weights is then simply the positive association minus the negative association. The theory and mathematics behind this learning algorithm can be found in Hinton et al. (2006).

### 3.3.5 DBN Implementation

The basic DBN is simply a stack of the RBMs explained in the previous section, trained layer-by-layer. The first RBM is trained by setting its visible states to be equal to the actual data samples, while each subsequent RBM’s visible state is set to the hidden state of the previous RBM when starting off the first RBM from an actual training sample. Each RBM is fully trained before starting the training of the next. In this implementation, “fully trained” is simply when it has run for a given number of epochs which is settable by the caller, but it could potentially be extended to stop training when a given condition is detected.

### 3.3.6 Fine-tuning the DBN

The intention with the fine-tuning phase is to globally adjust the weights after their initial layer-by-layer training. The reasoning behind this step is that the correctness of the layer-by-layer training is somewhat constrained because of the assumptions made when training each RBM in isolation. Each RBM is trained under the assumption that the next RBM perfectly models the joint probability distribution further on in the net; because of this, that next RBM and any subsequent RBMs can be ignored when calculating the weight updates as they are already assumed to be perfect. As long as the RBM is the last one in the chain, this assumption holds. However, when the next RBM is connected and trained, its weights will most likely *not* satisfy this assumption, so the previous RBM now has weights based on an assumption that is no longer accurate (Hinton et al., 2006).

The fine-tuning attempts to correct this discrepancy by adjusting the weights when this assumption is dropped. To achieve this, the symmetrical aspect of the weights is first dropped so that the recognition weights (i.e. the ones leading *away from* the input layer) are no longer the transpose of the generative weights (i.e. the ones leading *out towards* the input layer). The new network layout can be seen in Figure 3.6 where each arrow
represents a weight transformation; the states of the top layer represent the unsupervised features for a training sample when this sample is propagated up through the recognition layers.

Figure 3.6. The network layers of an instance of the DBNFinetuned class, constructed from a DBN made up of 3 RBMs.

In the implementation, this is achieved by instantiating an object of a separate fine-tuning class which is initiated by the parameters from the stack of RBMs. This class, called DBNFinetuned, implements a contrastive version of the wake-sleep algorithm, based on a synthesis of the original unsupervised wake-sleep algorithm described in Hinton et al. (1995) and the contrastive aspect described in Hinton et al. (2006) where a supervised version is defined.

The learning rule is similar to that of the RBM training, but because the symmetry of the weights is dropped, the recognition weights and the generative weights are now trained separately. In what is referred to as the *wake-phase*, the generative weights of each layer are updated to better reproduce one mini-batch of the training samples when generating a lower-layer representation based on a higher-layer representation according to the network in Figure 3.6. The updates are based on the contrastive learning rule where the associations are calculated after updating the states of each layer once from the input layer up through the recognition layers and back down through the generative layers. The positive associations are calculated as the association between the states of the recognition layer, at the same level as the input for the current generative weights, and
the states of the recognition layer at the same level as the outputs from those weights. The negative association is calculated as the association between the states of the same recognition layer as in the positive step and the activations (without sampling) of the generative layer that would receive the outputs of the weights being updated.

The generative biases are simply updated by the differences between the states of the corresponding recognition layer and the activations of the generative layer that they are connected to.

After this, alternating Gibbs sampling is performed for a number of steps (adjustable by setting a flag) by going up through the recognition layers and down again through the generative layers and starting off the next step from the resulting reconstructed input.

Finally, the recognition weights are updated in what is called the *sleep-phase* with the intention of becoming better at accurately representing the data in the lower layers with its states in the higher layers. The data in this phase is not a mini-batch of actual training samples but rather a mini-batch of “fantasy” samples, generated by going down through the generative layers from the final top states after the Gibbs sampling chain has ended. This mini-batch of “fantasy” samples is then used to initiate a full pass up through the recognition layers of the network. The updates are calculated by the same logic as in the wake-phase but in a mirrored fashion. The positive associations are calculated as the association between the states of the generative layer, at the same level as the input for the current recognition weights, and the states of the generative layer at the same level as the outputs from those weights. The negative association is calculated as the association between the states of the same generative layer as in the positive step and the activations (without sampling) of the recognition layer that would receive the outputs of the weights being updated.

Similarly to the wake-state, the recognition biases are updated by the differences between the states of the corresponding generative layer and the activations of the recognition layer that they are connected to.

### 3.3.7 Random Forest Classifier Implementation

Random Forests is an ML technique based on the Decision Tree algorithm. A Decision Tree is trained by building a tree of branches where each split into a set of branches is guided by partitioning on the value of a feature where the information gain from such a partitioning carries an optimal amount of information about the class label; class labels
are assigned to each leaf node based on the most common class of the training samples that would fall into that node. A Random Forest classifier combines a number of such trees, each trained on a subset of the possible features, and assigns a class label based on the consensus of the trees (Ivezić, Connolly, VanderPlas, & Gray, 2014).

The Random Forest classifier for this project was implemented in Python 2.7 by using the RandomForestClassifier class from the Scikit-learn library (Pedregosa et al., 2011). Two datasets were used, both separate from the ones used to train the DBN: one was used to train the Random Forest classifier and the second was used as a cross-validation set for selecting the optimal number of trees. Neither of these were used when obtaining the final test statistics.

The input data files containing the baseline features and the class labels were created by extracting these values from the SDSS database and saving them as a single binary npy file per dataset. The data files for the DBN features was created by first encoding each of the images from the training set by passing each image through the DBN and saving the hidden states from the deepest recognition layer as npy files. These npy files were then combined with the baseline features to create the full training and validation sets for the DBN model as one npy file per dataset.

The output from the RandomForest classifier is a vector containing the predicted labels for the inputs given. These are given as numerical values where 0=STAR, 1=GALAXY and 2=QSO.

3.3.8 The Software Development Process
Since the software development made up a considerable part of the research project, some details are given here as to how this effort was organized. This was also the biggest risk-factor of the project since the author was new to both the framework and the relevant ML techniques. The code needed to be completed on time and to a high enough quality that the actual experiments of the project could be run.

From the start of the project, Git17 was used for source control of all relevant files through the online repository site BitBucket18. Git is a version control system where changes to the files are tracked through a checkpoint system where the user can easily go back to an earlier version of the code to re-trace where bugs could have crept in along

17 Git is available from https://git-scm.com/ under the GNU General Public License version 2.0.
18 https://bitbucket.org/
the way. It also provides a work-history through the notes that are attached to every update that is committed. Multiple simultaneous versions of the work, referred to as branches, can also be kept and updated in parallel. This was very useful, for example when experimenting with new versions of the image preprocessing. Finally, it provides a reliable way to back up the work, and enabled the author to work from different physical workstations as needed.

Another important part of the software development process was the effort of ensuring that the implementation of the unfamiliar techniques was correct. Testing was not straightforward since the “correct” output for a certain set of inputs was not always obvious. Most of the testing used visualizations in one way or another to allow the author to visually confirm that the output “made sense”. An example of this has been covered in section 3.2.3 where the image preprocessing is explained and reconstruction of the image qualities of the data was important.

Another part of the testing was the testing of the RBM implementation which provides the base for all the unsupervised training. To guide this process, the author followed the recommendations of Yosinski and Lipson (2012) who provide a set of guidelines for visually debugging the implementation and training of RBMs. This allowed the author to find unexpected behavior in the learning and track down whether it was due to a software bug or less-than-optimal hyper-parameter settings. Naturally, these steps were also used for hyper-parameter tuning, so their details are relevant to that phase as well.

One type of visualization that was used was cost monitoring which is usually a good way to monitor the learning of any ML algorithm. For RBMs, the metrics used are usually either the reconstruction error or the cross-entropy (or both). These were already part of the Deep Learning library used, though the calculation of the reconstruction error was modified and the ability to record both measurements simultaneously was added. To visualize their change over time, the TensorFlow framework comes with a built-in tool that generates graphs for this purpose.

The reconstruction error was calculated by taking the mean squared difference between the original input data and the reconstructed data (after going from visible to hidden and back from hidden to visible). This is not exactly what the learning is minimizing, but it gives some indication of how well the algorithm is learning and can be understood as how well the RBM acts as a dimensionality reduction technique or compression
algorithm. If it perfectly captures the input, then the reconstruction error would become zero. However, it will most likely never reach zero for a realistic dataset, and this is also not the goal as the intention is to find a generalized representation rather than overfitting by perfectly reconstructing any noise in the data. Yosinski and Lipson (2012) report that the expected behavior is for the reconstruction error to decrease slowly at first, followed by a quick drop when the training is about halfway towards a desired goal, and then continue to decrease very slowly, while Hinton (2010a) observes an initial rapid decrease followed by a slow decrease. A rapid increase of the reconstruction error is taken as a sign that the changes in the weights are too high, causing the learning to diverge. This happened during the development of the RBM for the current project, and the reconstruction error helped in spotting it. After consulting the guidelines from Hinton (2010a), it was found that the initial weights were given too large values, and fixing this issue stabilized the reconstruction errors. Later on, this new weight initialization was further confirmed by use of a visualization of the hidden activations, described further on in this section.

The other cost that was monitored was the cross-entropy which is more closely related to what the model is actually learning. Given the approximated maximum likelihood learning of the RBM, the cross-entropy should reasonably decrease as the model improves (Cosma Rohilla Shaliz & Aryeh Kontorovich, 2010; Hinton et al., 2006). The cross-entropy w.r.t. one input sample is calculated here according to equation (1) where $P_{data}$ are the probabilities of the visible units for the input sample (corresponding to the values of the input data) and $P_{reconstruction}$ are the probabilities of the visible units given the states of the hidden units (corresponding to the activations of the visible units after going from visible to hidden and back from hidden to visible).

$$\text{cross-entropy} = - \left( P_{data} \times \log P_{reconstruction} + (1 - P_{data}) \times \log(1 - P_{reconstruction}) \right) \quad (1)$$

One of the visualization techniques from Yosinski and Lipson (2012) was implemented to show how the activations for the hidden units change over time for the same mini-batch. The output from this visualization can be seen in Figure 3.7, where each rectangle shows the activations (with 0.0 showing as grey, -1.0 and below as black and 1.0 and above as white) from one epoch including epoch “zero” before training has begun; each row in the rectangle represents one data sample, and each column represents one hidden unit.
This visualization helped confirm whether the weights and biases were initialized properly, and whether the learning was working as expected. Before learning takes place, the slightly noisy grey top rectangle from Figure 3.7 is expected if the weights have been initialized to small positive and negative values near zero. Then, as learning takes place, the activations of the units are expected to slowly converge towards their preferred values for each training sample. However, they should not show too many perfectly vertical lines of black and white as this would indicate that they are either always turned on or off regardless of the given training sample.

Figure 3.7. Visualization of the activation values of the hidden units from 10 epochs. Larger positive values are whiter and larger negative values are darker. Rows represent data samples, columns represent hidden units. The top rectangle represents initial activations, followed by epoch 1-8 and epoch 100.

Finally, as recommended both by Yosinski and Lipson (2012) and Hinton (2010a), visualizing the features that were learned by the model was of great help. While the model was found to be learning something through an apparently working implementation, it was essential to its usefulness to find out what it was learning. This is a technique that can be used for any network that is trained layer-by-layer and it can help detect overfitting and underfitting as well as problems with the input data. The implementation for visualizing the features of the first layer simply takes the weight matrix for each unit and converts it into an image where values above zero give increasingly white pixels and values below zero give increasingly black pixels. For subsequent layers, the weight matrices that connect to a unit are added up to show the features of that unit. A thing to note here is that it helps to output the normalized versions of these visualizations since the weights can turn out to be so small that the non-normalized images seem to be empty of any features; this was indeed the case at one
stage of the current project which led to some confusion as to whether the model was actually learning anything at all. An example of the visualization of 9 features, without and with normalization, can be seen in Figure 3.8.

![Figure 3.8. Visualization of 9 features from an RBM. The upper half of the image shows the features without normalization; the bottom half shows the same features with normalization.](image)

As mentioned in section 3.2.3, the technique described here helped detect a problem with the input data which caused the model to learn the non-generalizable feature of where in the image the object of interest happened to be located.

Finally, with regards to the software development, as is so often the case, it turned out that some tasks took up an unexpectedly large portion of the time. Due to rather limited computer memory, disk space and internet bandwidth resources, quite some effort was put into handling the Big Data aspect of the project. One path of the DBN development that was later abandoned was the implementation of a data handler that made use of an external Amazon RDS\(^\text{19}\) database where mini-batches were requested from the database one at a time to conserve disk space; unfortunately, the connection latency of this solution was far too high to be useful.

Perhaps the least expected time-sink was the task of image preprocessing which turned out to be far from straightforward. The author had not expected the process of acquiring the images from SDSS and converting them into usable numerical data to be so involved; perhaps the details of that process described in this thesis can be of help to someone attempting a similar project in the future. The original expectations on the image data,

\(^\text{19}\) [https://aws.amazon.com/rds/](https://aws.amazon.com/rds/)
based on the more accessible thumbnails in the Explore Tool\textsuperscript{20}, turned out to not hold for the full image data. Much time was spent on turning the images into usable quantities, complicated further by the conflict in coordinate systems between image libraries and numerical matrix libraries.

The main lesson learned for future projects was: always have a very clear picture of the data before writing the project plan.

3.4 Experiments

3.4.1 Hardware Specifications

During development, the experiments were run on the following machines: a VirtualBox\textsuperscript{21} 64-bit Ubuntu 14.04 LTS instance on the author’s personal desktop and laptop, and also an Amazon Web Services\textsuperscript{22} (AWS) EC2 t2.micro instance running Ubuntu 14.04.5 LTS 64-bit.

The AWS EC2 machine enabled the author to perform long-running experiments without leaving personal computers running and occupied over days, and it also enabled downloading and data preparation tasks to be run remotely.

The final model was trained on a desktop with a quad-core 3.5GHz i5 processor running a 64-bit Ubuntu 14.04 LTS instance with 4GB RAM on VirtualBox. This training was surprisingly fast, taking only around 60 minutes. If a larger dataset had been used, this time would of course have increased accordingly.

None of the machines were able to run GPU calculations, so a task for the future would be to compare the training time when using a GPU for the matrix computations.

The biggest resource constraint on this project was the time, bandwidth and disk space needed to download and prepare the image data. This was done through the AWS EC2 instance, taking 5 hours for the 10,000 object sample and approaching the disk space limit. Additionally, because of the hyper-parameter tuning and active development of the code, the reasonably short training time per model still had an impact.

\textsuperscript{20} http://skyserver.sdss.org/dr12/en/tools/explore/
\textsuperscript{21} VirtualBox is a virtualization application from Oracle, available from: https://www.virtualbox.org/
\textsuperscript{22} https://aws.amazon.com/
3.4.2 Running the Experiments
The experiments were run a large number of times during the development of the model to ensure it was working as intended. Originally, this was done through a script that would launch a single run of the experiment. As the software development neared completion, a more powerful Python script was written that could launch any number of experiments for unsupervised training, unsupervised fine-tuning, classification training and testing, and generating visualization. Additionally, a range of metrics from the classification tests were written by the script to a csv file for further analysis in other programs. Which steps were included, how many models were trained, and what hyper-parameter settings were used was all defined through human readable configuration files.

This setup made hyper-parameter tuning a very convenient process and also prepared the experiments for potentially being run at a larger scale in the future if the resources become available.

3.4.3 Hyper-Parameter Tuning
Hyper-parameter tuning was necessary for both the baseline model and the novel DBN model. This was an aspect that was considered early during the design of the experiments. In an effort to keep the comparison between the models as unbiased as possible, the classification technique was chosen to be one with as little need for hyper-parameter tuning as possible. For the Random Forest classifier, optimization was straightforward since the only hyper-parameter that was changed in this project was the number of trees. When it comes to the baseline model, this classifier makes up the entire model, so finding the optimal settings was only a matter of trying different settings for the number of trees.

Finding the optimal DBN model was more involved as it meant first training the full DBN and then comparing the results of the classifier. The first step was trying different layer-architectures and unit types, and finding reasonable learning rates for these. The methods used for testing the software implementation, described in section 3.3.8, were used for this stage.

Once some reasonable settings had been found, classification tests were performed by using the cross-validation set for the Random Forest classifier. As described in section 3.4.1 above, this was done through a script that also recorded several metrics, such as
the F₁ scores for each of the classes and the macro-averaged F₁ score which is the main metric of the final tests. More details on these metrics can be found in section 3.4.4 below. To find the optimal hyper-parameters, these metrics were manually evaluated with regards to the relevant hyper-parameters through a set of visualizations made in Tableau. Through these visualizations, trends (and sometimes a lack thereof) could easily be spotted which guided the next set of model settings to try. This provided a quick and easy way to find a decent set of hyper-parameters. Since the focus of this project was not to find the absolute optimal settings, this process was a sufficient in its rewards for effort.

3.4.4 Evaluation Metrics

As the evaluation criteria for the models’ performance, the F₁ score was used. The F₁ score for an individual class is the harmonic mean of the precision and recall for that class, calculated by the equation (2) where the number of true positives (TP), false positives (FP) and false negatives (FN) are known.

\[
\text{precision} = \frac{TP}{TP+FP} \quad \text{recall} = \frac{TP}{TP+FN} \quad F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
\]  \hspace{1cm} (2)

When looking at the F₁ score for a single class, this metric balances the score between a model’s ability to not miss any instances of that class, while also being precise in its predictions rather than simply predicting that class more often. This metric is used when testing and comparing the models’ performance on a per-class basis.

To also find a single value to represent the overall performance of each model, the macro-averaged F₁ score was used. The macro-averaged F₁ score is a single value that summarizes the performance on all classes where equal weight is given to each class regardless the class proportions (Sokolova & Lapalme, 2009). It is calculated from the average precision and recall scores for all classes as shown in equation (3).

\[
\text{macro-averaged } F_1 = 2 \times \frac{\text{precision}_{\text{average}} \times \text{recall}_{\text{average}}}{\text{precision}_{\text{average}} + \text{recall}_{\text{average}}}
\]  \hspace{1cm} (3)

Macro-averaging is used here for the evaluation to ensure that the model generalizes well to all classes. In the case of Astronomy tasks, the rarer classes tend to be considered as important, or even more important, than the more common classes.

---

23 http://www.tableau.com/
Taking this into account becomes especially important since the DBN model uses unsupervised learning on an imbalanced dataset. One of the strengths of unsupervised learning is that it can leverage the much larger amounts of unlabeled data that is available, but by doing so, there is a risk of biasing the model towards data with a more common profile. Evaluating the models based on the macro-averaged F1 score lets us assess their ability to generalize.

In addition to the overall comparison, the contribution of individual features was also evaluated for the DBN model. Specifically, the top 20 features were examined to see how many of these were DBN features. Each of these DBN features were also visualized to provide a better understanding of the information they carry.

3.4.5 Statistical Methods
Since the research question concerns whether the DBN features make a significant contribution to the model or not, emphasis was placed on producing results that could be readily compared and interpreted. The aim of the experiment was to provide evidence of whether the lower dimensionality features derived from the stack of DBNs are indeed providing useful information for a discriminative model or not. Since both models used a Random Forest classifier, a direct comparison of their results could be made between the DBN model and the baseline model.

The final training and testing of the models were performed 100 times for each model with a different seed for the random number generator each time to ensure that differences between the models were due to actual differences and not due to random chance. The test for statistical significance in the difference between the models’ results was done in PSPP with a one-tailed independent two-sample t-test using a 95% confidence interval and a p-value cut-off of .05 to test for significance.

3.5 Analysis Techniques
The conclusions that could be drawn from testing the hypotheses were concerned specifically with whether the unsupervised feature extraction provided additional useful information to the classification task. Besides this, while not the main objective of this thesis, a comparison was also made to current state-of-the-art results on a similar classification task to set the results into context. This was done by recalculating the reported metrics of the previous experiment to the metrics used in this work. The
performance of the state-of-the-art is compared both to the baseline model and the novel model and the implication of the differences is discussed.

Finally, the novel model of this thesis was also evaluated in a more qualitative light. This was done by visualizing the learned features of the DBN model to visually examine what it actually had learnt as this might give more insights as to how and why it works for the task at hand. For a further understanding of this, the feature importance aspect of the Random Forest classifier was also utilized; a bar graph was produced to show the importance of the top 20 features, and the meaning of these features was reported as either the color value it represented or with a visualization of the DBN features when such was the case.

3.6 Limitations

To test the classification results, a golden record is needed that provides the “true” class labels. When it comes to astronomical observations, the true labels are unknown and the golden record is based on some subset of definitions based on current theories (Borne, 2009b). For the present work, the golden record is defined by the classification rules of the SDSS spectroscopic pipeline and so are subject to any biases therein. Additionally, since only around 1% of the observed objects in the SDSS database have spectroscopic measurements, any biases in the target selection criteria have inevitably been propagated to the dataset used for this work.

Djorgovski et al. (2013) further discuss the basic limitation of Astronomy census research; any results from such research will be biased by some assumption of what distinguishes a certain class of objects from another. Taking this into account, it might be more appropriate to consider the present work as an investigation into the mapping of image data to certain spectroscopic qualities. Nevertheless, such a mapping does provide a basic analytical tool for astronomical census gathering based on the current understanding of the different phenomena.

Another limitation related to the dataset is that the SDSS data for each object is based on the single observation that was considered the best observation of that object. As such, this model does not take into account possible variations in the sources over time.
The strongest human bias in this work is the image preprocessing which unfortunately cannot be avoided. The author has attempted to keep this to a minimum by using threshold values that are relative to the measured values of each individual image.

Finally, due to project resource limitations, the sample size used for this project was very limited compared to the massive amount of data available in the SDSS database, and certainly compared to the potential amount of data to be found from observations of the sky. Nevertheless, by monitoring the learning of the model, there were indications that already at this sample size, there is a large amount of redundancy in the data, so the sample size is not expected to induce a strong limitation on the generalization of the applicability of the method that is tested.

3.7 Summary

This chapter covered the methodologies, project design and software development details of this research project. An overview of the project design was given first, followed by details on each phase of the project. Finally, limitations of the methods used were identified and reported.

A major part of the chapter was dedicated to the software development as this was the part that took up most of the time of the project. In many ways this research project was also a software development project, with a finished software product as an additional output.
4 RESULTS

4.1 Overview

To test the four hypotheses set out in section 1.2.4, the overall model performances were compared, followed by comparisons on a per-class-basis. The reader is reminded that the overall metric of comparison is the macro-averaged F₁ score, while the per-class metric is the F₁ score for each class, both of which give a value between 0.0 and 1.0 where higher is better. These metrics are explained further in section 3.4.4.

In this chapter, the results are presented from the final runs of the baseline model and the DBN model with the final hyper-parameter settings that were deemed to give the best results on the cross-validation set. These final model parameters are reported along with the relevant input features for each model, followed by the results and the statistical outcomes for each of the hypotheses. In addition to the numerical results, the results are also presented in box plots where the differences in means and the variations within the models become more apparent.

The results of the tests were obtained by running the Random Forest classifier for the baseline model and the DBN model 100 times each, with a different random seed each time to make the results more robust to random chance.

All four hypotheses were tested by one-tailed independent two-sample t-tests with a confidence interval of 95% with no assumption of equal variance. A cut-off level of p < .05 was used when testing for statistical significance.

4.2 Input Features and Model Parameters

4.2.1 Baseline Model

The input features for the baseline model consisted of the 10 color values obtained by taking the differences in the de-reddened magnitude values from the SDSS database. The colors used were u-g, u-r, u-i, u-z, g-r, g-i, g-z, r-i, r-z and i-z.

The Random Forest classifier for the baseline model was run with 129 trees and the standard settings in the Scikit-learn implementation.
4.2.2 DBN Model
The input features for the DBN model consisted of the same 10 color values that were used in the baseline model, along with 100 features extracted by the unsupervised DBN.

The final DBN model was built from 2 RBMs, the first with an input layer of 2000 logistic units (for the image pixel data) and a hidden layer of 1000 binary units, and the second with an input layer of 1000 binary units and a hidden layer of 100 binary units. The first RBM was trained for 20 epochs and the second RBM was trained for 50 epochs, both with a learning rate of 1. Fine-tuning of the final DBN was run for 30 epochs with a learning rate of 0.0001. Both the RBMs and the fine-tuning used 1 step of Gibbs sampling.

The Random Forest classifier for the DBN model was run with 80 trees and the standard settings in the Scikit-learn implementation.

4.3 Hypothesis 1
The first hypothesis that was tested was,

$$H_A: \text{The DBN model performs better than the baseline model on the three-way classification of STAR / GALAXY / QSO, measured by the macro-averaged } F_1 \text{ score.}$$

giving the following null hypothesis,

$$H_0: \text{The DBN model does not perform better than the baseline model on the three-way classification of STAR / GALAXY / QSO, measured by the macro-averaged } F_1 \text{ score.}$$

The macro-averaged $F_1$ score of the baseline model was found to have a mean of 0.8321 with a standard deviation of 0.0035, while the macro-averaged $F_1$ score of the DBN model had a mean of 0.8682 with a standard deviation of 0.0029. The difference in the means was tested for statistical significance to the cut-off level of $p < .05$ by a one-tailed independent two-sample t-test with a confidence interval of 95% with no assumption of equal variance. The difference was found to be highly statistically significant with $p < .001$. This provides evidence to reject the null hypothesis and accept the alternative hypothesis that the DBN model performs better than the baseline model on the three-way classification of STAR / GALAXY / QSO, measured by the macro-averaged $F_1$ score. A difference of 0.0362 was found (with a 95% confidence interval between 0.0353
and 0.0371) which translates to an increase of 4.35% in the macro-averaged $F_1$ score compared to the baseline model.

Figure 4.1 shows box plots of the output from the 100 runs from the baseline model (left), the final DBN model (right). For reference, the results from the DBN model before unsupervised fine-tuning is given in the middle.

![Box plots showing the macro-averaged $F_1$ score from 100 runs each for the baseline model (left), the DBN model before fine-tuning (middle) and the final fine-tuned DBN model (right).](image)

### 4.4 Hypothesis 2

The second hypothesis that was tested was,

$H_A$: The DBN model performs better than the baseline model at classifying the STAR class in the three-way classification of STAR / GALAXY / QSO, measured by the $F_1$ score for the STAR class.

giving the following null hypothesis,

$H_0$: The DBN model does not perform better than the baseline model at classifying the STAR class in the three-way classification of STAR / GALAXY / QSO, measured by the $F_1$ score for the STAR class.

The $F_1$ score of the STAR class for the baseline model was found to have a mean of 0.8037 with a standard deviation of 0.0041, while the $F_1$ score of the STAR class for the DBN model had a mean of 0.8664 with a standard deviation of 0.0037. The difference in the means was tested for statistical significance to the cut-off level of $p < .05$ by a
one-tailed independent two-sample t-test with a confidence interval of 95% with no assumption of equal variance. The difference was found to be highly statistically significant with \( p < .001 \). This provides evidence to reject the null hypothesis and accept the alternative hypothesis that the DBN model performs better than the baseline model at classifying the STAR class in the three-way classification of STAR / GALAXY / QSO, measured by the F\(_1\) score for the STAR class. A difference of 0.0628 was found (with a 95% confidence interval between 0.0617 and 0.0638) which translates to an increase of 7.81% in the F\(_1\) score of the STAR class compared to the baseline model.

Figure 4.2 shows box plots of the output from the 100 runs from the baseline model (left), the final DBN model (right). For reference, the results from the DBN model before unsupervised fine-tuning is shown in the middle.

Figure 4.2. Box plots showing the F\(_1\) score of the STAR class from 100 runs each for the baseline model (left), the DBN model before fine-tuning (middle) and the final fine-tuned DBN model (right).

4.5 Hypothesis 3

Next, the third hypothesis to be tested was,

\( H_3: \text{The DBN model performs better than the baseline model at classifying the GALAXY class in the three-way classification of STAR / GALAXY / QSO, measured by the F}_1\text{ score for the GALAXY class.} \)
giving the following null hypothesis,

\[ H_0: \text{The DBN model does not perform better than the baseline model at classifying the GALAXY class in the three-way classification of STAR / GALAXY / QSO, measured by the } F_1 \text{ score for the GALAXY class.} \]

The \( F_1 \) score of the GALAXY class for the baseline model was found to have a mean of 0.9376 with a standard deviation of 0.0014, while the \( F_1 \) score of the GALAXY class for the DBN model had a mean of 0.9582 with a standard deviation of 0.0012. The difference in the means was tested for statistical significance to the cut-off level of \( p < .05 \) by a one-tailed independent two-sample t-test with a confidence interval of 95% with no assumption of equal variance. The difference was found to be highly statistically significant with \( p < .001 \). This provides evidence to reject the null hypothesis and accept the alternative hypothesis that the DBN model performs better than the baseline model at classifying the GALAXY class in the three-way classification of STAR / GALAXY / QSO, measured by the \( F_1 \) score for the GALAXY class. A difference of 0.0206 was found (with a 95% confidence interval between 0.0202 and 0.0210) which translates to an increase of 2.20% in the \( F_1 \) score of the GALAXY class compared to the baseline model.

Figure 4.3 shows box plots of the output from the 100 runs from the baseline model (left), the final DBN model (right). For reference, the results from the DBN model before unsupervised fine-tuning is shown in the middle.

![Box plots](image)

**Figure 4.3.** Box plots showing the \( F_1 \) score of the GALAXY class from 100 runs each for the baseline model (left), the DBN model before fine-tuning (middle) and the final fine-tuned DBN model (right).
4.6 Hypothesis 4

Finally, the last hypothesis to be tested was,

\[ H_4: \text{The DBN model performs better than the baseline model at classifying the QSO class in the three-way classification of STAR / GALAXY / QSO, measured by the } F_1 \text{ score for the QSO class.} \]

giving the following null hypothesis,

\[ H_0: \text{The DBN model does not perform better than the baseline model at classifying the QSO class in the three-way classification of STAR / GALAXY / QSO, measured by the } F_1 \text{ score for the QSO class.} \]

The \( F_1 \) score of the QSO class for the baseline model had a mean of 0.7549 with a standard deviation of 0.0071, while the \( F_1 \) score of the QSO class for the DBN model had a mean of 0.7801 with a standard deviation of 0.0057. The difference in the means was tested for statistical significance to the cut-off level of \( p < .05 \) by a one-tailed independent two-sample t-test with a confidence interval of 95% with no assumption of equal variance. The difference was found to be highly statistically significant with \( p < .001 \). This provides evidence to reject the null hypothesis and accept the alternative hypothesis that the DBN model performs better than the baseline model at classifying the QSO class in the three-way classification of STAR / GALAXY / QSO, measured by the \( F_1 \) score for the QSO class. A difference of 0.0251 was found (with a 95% confidence interval between 0.0233 and 0.0269) which translates to an increase of 3.32% of the \( F_1 \) score of the QSO class compared to the baseline model.

Figure 4.4. Box plots showing the \( F_1 \) score of the QSO class from 100 runs each for the baseline model (left), the DBN model before fine-tuning (middle) and the final fine-tuned DBN model (right).
Figure 4.4 shows box plots of the output from the 100 runs from the baseline model (left), the final DBN model (right). For reference, the results from the DBN model before unsupervised fine-tuning is given in the middle.

4.7 Summary

The statistical tests were satisfied for all hypotheses that were defined in section 1.2.4, confirming the following:

1) The DBN model performs better than the baseline model on the three-way classification of STAR / GALAXY / QSO, measured by the macro-averaged F₁ score.

2) The DBN model performs better than the baseline model at classifying the STAR class in the three-way classification of STAR / GALAXY / QSO, measured by the F₁ score for the STAR class.

3) The DBN model performs better than the baseline model at classifying the GALAXY class in the three-way classification of STAR / GALAXY / QSO, measured by the F₁ score for the GALAXY class.

4) The DBN model performs better than the baseline model at classifying the QSO class in the three-way classification of STAR / GALAXY / QSO, measured by the F₁ score for the QSO class.

The mean of the results from the two competing models are summarized in Table 4.1 along with the DBN model’s increase in performance compared to the baseline model given in percent.

<table>
<thead>
<tr>
<th></th>
<th>baseline model</th>
<th>DBN model</th>
<th>absolute increase</th>
<th>relative increase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macro-averaged F₁ score</td>
<td>0.8321</td>
<td>0.8682</td>
<td>0.0361</td>
<td>4.35%</td>
</tr>
<tr>
<td>F₁ score, STAR</td>
<td>0.8037</td>
<td>0.8664</td>
<td>0.0627</td>
<td>7.81%</td>
</tr>
<tr>
<td>F₁ score, GALAXY</td>
<td>0.9376</td>
<td>0.9582</td>
<td>0.0206</td>
<td>2.20%</td>
</tr>
<tr>
<td>F₁ score, QSO</td>
<td>0.7549</td>
<td>0.7801</td>
<td>0.0252</td>
<td>3.32%</td>
</tr>
</tbody>
</table>

Table 4.1. Summary of the test results from the experiments. The table shows the means from 100 runs with each model, along with the improvement in the mean of the DBN model over the baseline model.
The largest increase was found in the F$_1$ score for the STAR class, while the most modest improvement was found in the F$_1$ score for the GALAXY class which was the class with the highest performance in both models.
5 DISCUSSION

5.1 Overview

This chapter examines the final DBN model with regards to the results and how the model works, and identifies some of its strengths and weaknesses. The discussion starts with an analysis of what the final model actually learned based on a set of visualizations to illustrate this; this part examines whether the model actually learned something sensible or whether it was overfitting to some previously unknown peculiarity of the dataset. After this, the numerical results of the model are analyzed in the context of current the state-of-the-art results from the literature. Following this, some comments are given on how the model was trained with regards to the hyper-parameter tuning. Finally, a brief discussion is given on some general qualities of this type of model that could potentially influence its suitability to other directions of future work.

5.2 Final Model and Results

When applying ML algorithms, it is important to understand not only if learning is taking place but also what the model has learnt. This enables the domain expert to assess whether the model is suitable to a certain task, and it can also provide some insight into the strengths and weaknesses of the model. Since the author is not a domain expert on Astronomy, the discussion here should be seen as mainly coming from an ML point of view, with visualizations provided for further interpretation by astronomers.

5.2.1 Feature Importance

To better understand what the classifier based its decisions on, the feature importance of the model was investigated. In Random Forests, the feature importance assigns a relative value to each feature based on how much impact that feature has on labeling the training set. A bar graph of the relative importance of the top 20 features can be seen in Figure 5.1, where the DBN features are accompanied with visualizations. The visualized features are normalized to values between -1.0 and 1.0 where positive values are brighter and negative values are darker; a drawback of this normalization is that it can’t be said where the actual zero-point of the weights are, but it does show the shapes that were learned as well as some indication of wavelength comparisons, especially prominent in feature DBN 7 where detections in the z-band seem to be given strong negative weights.
Figure 5.1. Relative feature importance of the top 20 features from the DBN model’s Random Forest classifier. The top 10 features are color features, named by the wavelength bands they compare, while the next 10 are unsupervised features from the DBN, named simply by their order of importance, and accompanied by normalized visualizations of their weights.

As can be seen, the top 10 features were the 10 color features. This is not surprising as they have been shown to be very useful to the classification task on their own, but it is interesting that all of them are in the top 10. Clearly the information they carry is very high per feature despite the overlap in information between them.

The DBN features, on the other hand, carry less information per feature, but are more numerous which may enable a more fine-grained decision boundary. When summing up the total contribution of all the DBN features, they add up to 28.43% of the feature importance of the classifier.

Figure 5.2. Normalized visualization of the 13 redundant DBN features that were never used in the Random Forest classifier.
13 out of the 100 DBN features were never used at all by the classifier. Looking at the visualization of these, shown in Figure 5.2, it is clear that they carry highly redundant information, suggesting that a final layer with fewer features could provide similar results at a lower dimensionality.

5.2.2 Feature Characteristics
To understand the function of each layer in the DBN, one can compare the type of features they capture. Figure 5.3 shows the first 100 (out of 1000) features of the first DBN layer and the full set of 100 features of the second layer. The power of the layer-wise training is often attributed to the hierarchical learning it enables, where one would expect to see lower-level features such as edges in the first layer followed by higher-level composite features in the layers beyond the first (Bengio, 2009). Looking at the model trained for this thesis, the two layers actually show rather similar-looking features. While the characteristic edges, visible as images with an area of strong contrast surrounded by mostly smooth grey, can be found in the first layer, there are also quite a few features that seem to identify a complete object. The second layer features look similar, though with a higher proportion of the more “complete” features.

![Figure 5.3. Normalized visualizations of the DBN features. The top half shows the first 100 (out of 1000) features of the first layer; the bottom half shows all 100 features of the second layer.](image)

With this in mind, there might be room for improvement of the model by implementing one or more techniques that encourage this hierarchical aspect, such as adding a sparsity goal (Hinton, 2010a) or implementing dropout (Hinton, Srivastava, Krizhevsky, Sutskever, & Salakhutdinov, 2012). A sparser representation, aside from leading to
better interpretability of the function of each layer, has also been found to lead to better generalization and less sensitivity to noise in the data (Bengio, 2009). It is also possible that a sparser representation in the first layer could lead to less overlap of information in the second layer since it builds its representations from the building-blocks found in the previous layer.

5.2.3 Sampling from the Model

One of the characteristics of generative models is that they learn a probability distribution function that can be used to generate data similar to that of the training data (Bengio, 2009). This provides a way to visualize what sort of data the model has learnt to represent by sampling from a hidden state that approaches the equilibrium of the model. This was done for the final DBN model of this work by initiating the deepest hidden states with the hidden representation of an actual training sample, followed by a number of steps of Gibbs sampling between the top two layers, before using the generative weights to sample from the resulting hidden state. Figure 5.4 shows the samples that were obtained in this way after 100, 500 and 1000 steps of Gibbs sampling, with the initial training samples shown in the top rows; each training sample is a 5-band multi-wavelength image, so a similar 5-part image is obtained at each of the sampling steps.

**Figure 5.4.** Samples obtained by sampling from the generative model. The top rows show the actual multi-wavelength training samples that were used to start the Gibbs sampling; the following rows show samples obtained after 100, 500 and 1000 steps of Gibbs sampling between the top two layers.

From the samples in Figure 5.4, it would seem like the model prefers extended sources. Additionally, it can be seen that it prefers to generate images where the first wavelength band, the u-band, is empty. The first effect can possibly be explained by the lack of a sparsity goal as discussed in the previous section of this chapter; since many units learn a little bit of everything, the extended qualities in the dataset would be found in many of the generative weights. The second observation, regarding the tendency to generate
images that are empty in the u-band, can be explained by the large amount of training samples that show this behavior after the noise-reduction described in section 3.2.3 has been applied. Interestingly though, when using the obtained samples to initiate further samples and repeating this for a number of steps, there was a tendency to eventually generate empty data in all bands except the i-band. In the i-band, however, the signal could become weaker only to pick up again in subsequent sampling, generating a variety of extended and more concentrated shapes. For some reason, the model exhibits a preference to representing the shapes found in this band. Again, this could possibly be related to a lack of sparsity; because of the way the weights work together with regards to the pixels, they would need to be individually smaller, which might just give a too weak signal during the generative process for the other bands.

5.3 Comparison to Current State-of-the-Art Results

Brescia et al. (2015) used a different classifier on the same baseline features as the ones used in this work with the addition of one out of two magnitude values. To compare their results to those of this work, their reported metrics were recalculated into the same metrics used in this work. This was also done for their best model that used the 2 different sets of the 5 magnitude values.

<table>
<thead>
<tr>
<th></th>
<th>10 color and u magnitude (Brescia et al., 2015)</th>
<th>10 color and r magnitude (Brescia et al., 2015)</th>
<th>2 x 5 magnitudes (Brescia et al., 2015)</th>
<th>Baseline model</th>
<th>DBN model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macro-averaged $F_1$</td>
<td>0.8780</td>
<td>0.8769</td>
<td>0.9135</td>
<td>0.8321</td>
<td>0.8682</td>
</tr>
<tr>
<td>$F_1$, STAR</td>
<td>0.8622</td>
<td>0.8593</td>
<td>0.8996</td>
<td>0.8037</td>
<td>0.8664</td>
</tr>
<tr>
<td>$F_1$, GALAXY</td>
<td>0.9028</td>
<td>0.9014</td>
<td>0.9522</td>
<td>0.9376</td>
<td>0.9582</td>
</tr>
<tr>
<td>$F_1$, QSO</td>
<td>0.8689</td>
<td>0.8699</td>
<td>0.8866</td>
<td>0.7549</td>
<td>0.7801</td>
</tr>
</tbody>
</table>

Table 5.1. Comparison of the results of the present work to those of Brescia et al. (2015) after recalculating to the metrics used for the present work.

Table 5.1 shows these recalculated results next to the results of the models in this work. As can be seen, the first two models from Brescia et al. (2015) perform slightly better overall to the DBN model. They also outperform the baseline model by quite a bit, indicating that there could be room for further improvement of the DBN model’s result by exploring more complex classification layers.
Looking at the strength of the DBN model, the galaxy class shows the best $F_1$ performance, with a nearly identical, marginally better $F_1$ score than the best model of Brescia et al. (2015). It would be tempting to explain this by the fact that the class label of an image of a well-resolved galaxy would be rather unambiguous, though this would not explain why the baseline model is also so strong for this class, or why the galaxy class was the one with the smallest increase in performance. Still, this quality would be expected to influence the result, and perhaps it can even explain the differences in results between the models from Brescia et al. (2015); since the 2 sets of magnitudes used in their better model are calculated under different assumptions about their source, it is possible that their better model was able to fit to the difference between these values based on the source that was measured.

Looking instead at the weakness of the DBN model, Table 5.1 clearly shows that it is the QSO class that lags behind in performance. Interestingly, this is the only class where the first two models outperform the DBN model. One explanation for this could be that the classification of QSOs in particular require a more complex classifier. Another explanation could be related to the addition of the extra magnitude feature, especially in light of the further improvement to all classes in their best model which uses the magnitudes instead of colors. As discussed in section 2.2.3, these features, while clearly useful in astronomical classification tasks, may carry the risk of biasing the results to the instrument quality and target selection criteria of the survey from which the data originated. At best, this bias is studied and used in a positive way to enable further study of interesting object classes. At worst, this could lead to a spiral of biases; if objects of a specific class are sought mainly in magnitude ranges where they have previously been found, then naturally most new observations of such objects will also be found in those regions.

However, a different explanation to the usefulness of the magnitude features could be that the classifier used by Brescia et al. (2015) was able to model more complex internal relationships between the magnitudes than could be captured by the pre-calculated color features that were used for the baseline model. It is also possible that the lower performance on the QSO class for this work is due to training the Random Forest classifier on an imbalanced dataset; it is possible that an improvement could be achieved by training this classifier on a re-balanced dataset or by supplying sample weights to the classifier to counter this imbalance.
In conclusion, on the task of three-class separation between STAR, GALAXY and QSO on the SDSS dataset, the model presented in this work performs in the same region as current state-of-the-art on the STAR and GALAXY classes, with a lower performance on the QSO class. Future work is recommended into investigating the impact of the novel features when using different classifiers and additional photometric features.

5.4 Training the Final Model

5.4.1 Overview
This section briefly covers how visualizations that were used for monitoring the training of the model. While hyper-parameter tuning was a necessary task, it was not the main focus of this thesis. Some suggestions are given, based on these visualizations, on possible improvements to the model if more time could be spent on this aspect.

5.4.2 Learning Curves
A common way to monitor the learning of any ML model is to observe the change in the cost that one wishes to minimize. This is usually observed on both a training and a validation set with the intention of finding an optimal number of training steps (or other parameter settings) where the cost for the validation set has planed out. Typically, the cost of the training set tends to continue decreasing after the validation set has reached its near optimal point, and it is not uncommon to see the cost of the validation set starting to increase due to the model overfitting on the training set (Kelleher et al., 2015a).

Figure 5.5 shows the learning curves for the final model based on the cross-entropy measure, explained further in section 3.3.8. As can be seen, there is room for improvement especially for the training of the two RBM layers before fine-tuning. Their learning curves seem to suggest that 10 epochs or less would have been more appropriate. The fine-tuning seems to have a more appropriate number of epochs which could possibly even be increased slightly, though the output from a training run with a larger number of epochs would be needed to assess this. However, due to time constraints and a previously more rough-cut view of the learning curves, these curves represent the training of the final model.
Figure 5.5. Learning curves for the final model showing the cross-entropy cost for the training and validation sets at each epoch for layer 1 and layer 2, and every 10th epoch for the global fine-tuning.

Aside from deciding roughly on the number of training steps for the model, the learning curves were also used to observe the effect of different learning rates and different visible units. While originally Gaussian units were used, it was found that logistic units, referred to as truncated exponential units by Larochelle et al. (2009), enabled far more stable learning due to their limited output range, especially when using larger number of units.
5.4.3 Cross-Validation on the Classification Task

Though the training of the DBN was entirely unsupervised, there was an aspect of semi-supervision in the hyper-parameter tuning. In addition to monitoring the learning curves, discussed in the previous section, some cross-validation was also performed with respect to the classification task. This was done by recording the metrics for the classification task when running model with different hyper-parameter settings. These results were then plotted with respect to the hyper-parameter of interest by use of visualizations made in Tableau. One such visualization is shown in Figure 5.6 where the effect of fine-tuning the DBN was assessed with regards to different learning rates and number of trees for the Random Forest classifier.

![Cross-validation result](image)

*Figure 5.6. Hyper-parameter evaluation via cross-validation on the classification task.*

This method of cross-validation for unsupervised models based on the performance on a classification task has previously been explored by Bengio (2012); interestingly, it was applied it to a task where Transfer Learning was the goal, with a successful outcome, so this suggests that the unsupervised qualities might not suffer despite this bias to the classification task.

5.5 Other Qualities of the Model

As discussed in section 3.6, the golden record for classification tasks in Astronomy is something that is likely to change as our understanding of the phenomena grow. This is another motivation for the use of unsupervised feature extraction. Since the feature extraction process does not rely on the labels or any assumptions made about them, these
features can be applied to a classification task with a new golden record as our understanding of the spectroscopic data improves. Only the final classification layer would need to be retrained in this type of model since the unsupervised layers are trained independently from the labeled data. This approach was demonstrated by Bengio et al. (2013) who successfully used unsupervised pre-training on a Transfer Learning task. A purely supervised model, on the other hand, would likely need to be retrained from scratch since all its weights would be tied to the previous assumptions.

Additionally, because of the unsupervised nature of the DBN model, it could potentially be trained on the full unlabeled dataset available, and use the labeled data on for the final classification layer. This opens up a large amount of potential training data that would otherwise be unusable for training, giving the model access not only to more data but also potentially to more representative data.

Finally, regarding the Big Data aspect of the domain, the additional benefit of acting as a dimensionality reduction algorithm is something that could potentially be very useful for upcoming surveys.

5.6 Summary

This chapter provided an analysis of what the final model actual learned, based on a set of visualizations techniques that leveraged its generative nature and its layer-by-layer training. Some redundancy was found in the features, and there were indications that the multi-wavelength aspect of the data might not have been fully explored by the model. A discussion was also given on the contributions of the different features of the model, where it was found that the color-features were the top 10 contributors, while the DBN features were given a total combined contribution of 28.43%.

Further, the results presented in chapter 4 were compared to current state-of-the-art results from the literature; the model of the present work was found to be of competitive performance, although with a weakness in the QSO class.

Some insight into possible further improvements in the training of the model were identified, concluding that adding a sparsity goal to the training would possibly be the most promising direction.

Finally, some additional qualities of the model, besides its immediate numerical results, were discussed in the final section.
6 CONCLUSIONS

6.1 Research Overview

This thesis has provided results from a statistical test of the usefulness of unsupervised feature-extraction from image data for astronomical classification tasks. The motivations for the suitability of this approach to the domain has also been discussed.

The proposed approach was applied to the real-world problem of photometric classification of astronomical objects in the SDSS dataset, and the results were compared to current state-of-the-art results.

To carry out the experiments, an unsupervised DBN model was implemented in Python 2.7. This implementation handles the Big Data aspect of the problem domain through a flexible memory management model and by enabling computational scalability through the TensorFlow framework. Various utility scripts were produced to manage the data extraction and preprocessing steps, and to enable visualizations for both quality assurance purposes and data exploration tasks. A highly automated approach was designed for running the experiments multiple times with various settings through the use of convenient configuration files.

This thesis has also demonstrated the use of a Random Forest classifier for the evaluation of the impact of different features on a classification task; this can be used during model development to assess the potential usefulness (or lack thereof) of intermediate features while avoiding the more extensive hyper-parameter tuning of a complex classification layer that might be added at a later stage.

6.2 Research Question

The research question that was investigated in this thesis was: Can unsupervised feature-extraction from multi-wavelength astronomical image data provide a significant advantage in the photometric classification of stars, galaxies and QSOs?

This was investigated by comparing the results of a model using the proposed unsupervised feature extraction, to one using only a common set of baseline features. Through this method, the advantage of using the proposed unsupervised features could be tested and quantified.
6.3 Results

The DBN features added a statistically significant increase of 0.0361 to the macro-averaged F₁ score, translating into a 4.35% increase, compared to a model using only the baseline features. This supports the evidence found in previous literature that there is useful information in astronomical image data that can improve the results of photometric classification tasks for stars, galaxies and QSOs.

The statistical significance of the results, presented in chapter 4, confirms the conclusions drawn from the literature review regarding the usefulness of multi-wavelength image data in astronomical classification tasks. Since the test was done on the macro-averaged F₁ score, the model’s ability to generalize to all three classes was emphasized. This is an important aspect to keep in mind for imbalanced datasets when the performance of less common classes is important. For Astronomy this is especially relevant since it is often the rarer events that are of the most immediate interest.

6.4 Contributions

The main contribution of this work is the statistically verified support for the usefulness of unsupervised feature extraction from image data for the automated separation of stars, galaxies and QSOs. This result provides a basis for future research in this direction.

The unsupervised approach enables the use of the massively abundant unlabeled data. This data is growing at a rapid pace, both due to an increasing rate of observations and due to the higher resolution in the observations. Because of this, the dimensionality reduction quality of the unsupervised training is also something that is highly desirable; by extracting representative lower-dimensionality features, this intermediate data provides a more accessible quantity that can be used in other works.

For future works of a practical character in the area of photometric classification, this thesis has provided a method with competitive results and the additional benefit of being interpretable. Thanks to this interpretability, the results can be inspected in a way that is not possible with black-box techniques. This increases the potential impact of this work since its model can provide a better understanding of why it works, which adds more value beside the end-results. Firstly, this can provide additional insights about the studied phenomena, and secondly, it enables human experts to verify that the model is basing its results on sensible decisions.
Finally, as a product of this work, a scalable technical implementation has been provided for the unsupervised training of RBMs and DBNs, along with unsupervised fine-tuning. This implementation has been made available for future work and applications, and can be found on GitHub under user AnnikaLindh’s project DBNTensorFlow\(^{24}\). Since the implementations of these algorithms were not tied to their specific use in this thesis, the produced source code can also be readily applied to unsupervised feature-extraction tasks of other domains.

### 6.5 Limitations

While the limitations of the research methodologies were covered in section 3.6, this section discusses the limitations to the conclusions that can be drawn from the results.

One limitation discussed previously is the small sample size compared to the vast amounts of astronomical data available. This does not put a strong limitation on the conclusions drawn about the general approach that was tested, but it would be valuable to see this work replicated at a larger scale. Additionally, if the final DBN model from this research project is used in future work as a research methodology, then those research projects need to train it on a sample that is of a size and composition suitable to the specific research question being investigated.

Another limitation on the scope of the present work is that it cannot be absolutely concluded whether all of the 5 wavelength bands were necessary or if images from a single wavelength band had been sufficient in combination with the color values. While visualizations of the features indicate that some differences between the wavelength bands were learned, sampling from the generative model showed a strong preference for a single wavelength band. To assess whether the multi-wavelength quality of the images is actually useful to the classification task, additional tests would need to be carried out that compare the performance of the DBN model of this work to a DBN model trained on single-wavelength image data, with and without the color features.

Finally, for a full assessment into the quality of the features that were learned by the model, a joint effort would be needed with the addition of a professional astronomer.

---

\(^{24}\) [https://github.com/AnnikaLindh/DBNTensorFlow](https://github.com/AnnikaLindh/DBNTensorFlow)
6.6 **Future Work and Recommendations**

6.6.1 Improving the Discriminative Performance
In this work, the extracted features were used in a simple Random Forest classifier. While this showed promising results, future work is recommended to try different classification layers for the model, such as adding a softmax layer to the deep network structure or using the unsupervised features as inputs to the classifier used by Brescia et al. (2015). This could potentially improve the results further. Naturally, further work could also go into improving the DBN by exploring a larger set of hyper-parameter settings.

6.6.2 Improving the Quality of the Features
While working directly on improving the discriminative results may be a faster way to better test results, a more long-term approach would be to invest time into improving the quality of the features that model learns. Because of the visual inspection qualities of the model, this aspect can be readily assessed both from an ML perspective and an astronomer’s perspective.

As discussed in section 5.2.2, regarding the characteristics of the DBN features, it might help improve both the performance and interpretability of the model if the features were more hierarchical in nature between layers. Recommendations for future work in this area is to experiment with a sparsity target or the dropout technique for the training.

It is also possible that the quality of the features was affected by the use of input data that held representations of the same object in multiple wavelength bands. It could well be that this over-complicated the learning by simultaneously training the model to recognize both the individual structures and the correlations between these structures. Future work into this aspect is discussed further in the next section.

6.6.3 Investigating the Multi-Wavelength Aspect
As discussed previously in section 5.2.3 with regards to the data that was sampled from the generative model, it would be valuable to investigate whether the multi-wavelength aspect of the images added to their usefulness or whether a single wavelength band would give equal results. Similarly, if the multi-wavelength aspect is found to be significant, it could also be investigated whether adding further wavelength bands, by combining data from multiple surveys, would improve the results.
Another way to investigate the multi-wavelength aspect would be to separate the task to avoid the problem of simultaneously training the model on the full range of wavelengths. Instead, the DBN could be trained on single-wavelength images from each of the wavelength bands, followed by a classifier that accepts one set of DBN features for each wavelength band. Naturally, these sets of DBN features could also be used as inputs to a final unsupervised DBN layer which is trained to recognize the correlations between them.

6.6.4 Suitability to Transfer Learning
Most interesting perhaps would be for future work to investigate the usefulness of the DBN features in Transfer Learning, where the same unsupervised features are used for different classification tasks. If this were to show promising results, then building a publicly available database of such generalized features could potentially enable further contributions to the field.

6.6.5 Applied Research
Finally, the model used in this thesis could potentially be used directly in future works of applied research. This would include any task that requires automated photometric classification, with one such example being target selection for spectroscopic observations.

6.7 Impact
By contributing to the area of automated photometric classification, this work could potentially have an impact on the design of future research projects that depend on such classification results. With the rapid increase in data output from astronomical surveys, this is a highly relevant area.

At a larger scale, better results in this area will help advance the field of Astronomy, leading to a better understanding of our universe and the basic laws of physics which affect all the natural sciences. This is of course not an immediate impact of the work of this humble thesis, but it is an important aspect of a field that is perhaps sometimes viewed as nothing more than an area of interest for academics with an unhealthy level of curiosity.
7 REFERENCES


