TM470 The computing and IT project - EMA 2024

A Journey into Deep Learning for Astronomical Image Classification

Emily Wills Z8790641

Table of Contents:

Part 1: Problem description **2**
- Nature and context of problem 2
- Proposed solution 3
- Analysis of likely impact & Discussion of wider implications 4

Part 2: A Review of Literature **6**
- Convolutional Neural Networks 6
- Machine Learning in Galactic Astronomy 6
- Citizen Science & Galaxy Zoo 7
- SETI@home 8

Part 3: A talk about Citizen Science **9**

Part 4: An Exploration of Convolutional Neural Networks **10**
- Python & Anaconda 10
- MNIST 11
- CIFAR-10 14
- Can I use the GPU? 18

Part 5: Galaxy Classification **19**
- AstroNN & Galaxy10 DECals 19
- Setup 20
- The Models 22
- Results 29

Part 6: Conclusion & Review of Project Work **32**

Part 7: Review and Reflection **33**
- Review of project management 33
- Review of personal development 33

References **35**

Datasets **36**

Appendix A: Model Diagrams **37**

Part 1: Problem description

Nature and context of problem

This project aims to create a neural network in Python that can run on a home personal computer; exploring what can be achieved within a reasonable timeframe, and the constraints and limitations imposed by consumer hardware.

The ultimate goal is to develop an image classifier using a deep learning convolutional neural network (CNN) capable of determining galaxy types from their images, building upon the efforts of the Galaxy Zoo project.

While artificial intelligence and machine learning are currently hot topics with major efforts being undertaken by large companies with substantial resources, this project seeks to demonstrate that it is possible to explore and have an engaging experience with the fundamental techniques of these technologies without the need for specialised hardware. To this end, the goal is to build and test a neural network capable of classifying complex images on a home PC, examining the approachability of these techniques for individuals curious about exploring them.

Python was chosen for this task due to its accessibility, ease of setup, and the availability of free libraries, guides, and datasets for machine learning tasks.

The project starts with a simpler classification task and progressively works towards galaxy classification, allowing for the identification and addressing of limitations encountered along the way.

Depending on the project's progress, a greater emphasis may be placed on either the Galaxy Zoo project or the personal hardware angle, as exploring both aspects are engaging prospects.

Why is it considered a problem?

AI and machine learning efforts right now feel like all the major efforts are being done by big companies with large resources.

Classifying galaxies based on their morphology is a challenging task, even for experienced astronomers. The sheer volume of astronomical images available from surveys like the Sloan Digital Sky Survey makes it impractical for human experts to manually classify them all in a timely manner, with the largest effort to do so being the crowdsourced Galaxy Zoo project.

Galaxy Zoo tasked volunteers to classify galaxies by hand. Requiring the efforts of thousands of people and spanning several years. This method of classifying galaxies is not ideal with automated telescopes mapping more and more of the sky creating so many images that even community efforts may be unable to classify future discoveries in a timely manner.

Therefore, an automated and reliable method for galaxy classification could provide benefits to astronomers.

What will be the benefits of solving it?

Classifying images by hand is inefficient compared to newer artificial intelligence methods. Applying these techniques could have several benefits over classifying by hand:

1. It could enable faster and more accurate classification of large datasets of galaxy images, crucial for understanding the formation and evolution of galaxies in the universe.
2. It can potentially lead to new discoveries or insights by identifying patterns or anomalies that might have otherwise been overlooked by human observers.

3. It can assist astronomers in prioritising targets for further study or observation based on the classified galaxy types.
4. And it can contribute to the advancement of machine learning techniques in the field of astronomy and astrophysics in general.

This project hopes to show how these machine learning techniques can be implemented and additionally discuss how citizen science initiatives can continue to contribute to these efforts even with the increasing use of machine learning in this domain.

What are the key ICT aspects of this problem?

The key ICT aspects involved in this problem include: Building a neural network to gradually solve more complex image classification tasks to eventually classify galaxies with a discussion on the approachability of these technologies on consumer hardware.

I will use image processing and computer vision techniques for preprocessing and extracting relevant features from large datasets of labelled images. Machine learning algorithms, specifically CNNs, which are well-suited for image classification tasks. Programming languages and libraries, such as Python and deep learning frameworks like TensorFlow.

All these resources are available for free and are not subject to time constraints on their use, all code will be run on my home PC.

What is your existing knowledge of this problem?

I have studied convolutional neural networks with the Open University and have been focusing my programming efforts on python so hopefully the programming aspect of the project won't prove to be an issue. I have used Convolutional Neural Networks in TM358 Machine learning and artificial intelligence, and have successfully applied them to various image classification tasks.

I also have an interest in astronomy and have been aware of the galaxy zoo project since participating in it as a teenager at school. The Galaxy Zoo project is a citizen science initiative that aims to classify galaxies based on their visual morphology with the help of volunteers.

Proposed solution

What might the solution look like?

I am hoping to have a convolutional neural network written in python that can classify images of galaxies into their widely accepted type. The models themselves may build upon those I have developed throughout my Open University studies, namely: TM358 Machine learning and artificial intelligence.

Using data from the Galaxy Zoo project, A CNN model will be developed and optimised for the task of classifying galaxy images based on their morphological features. The solution will demonstrate the application of deep learning techniques to astronomical image analysis, and will potentially explore the ability of consumer hardware to perform such computationally intensive tasks.

Will the solution be within the specialism route of my degree?

I want to build a solution that takes an existing problem and shows that it can now be done by one person with readily available consumer hardware using new data science technologies. The proposed solution of creating a CNN classifier aligns with the modules and areas of focus chosen during my degree, as I selected modules focused around data science and artificial intelligence, specifically TM351 Data management and analysis and TM358 Machine learning and artificial intelligence.

What am I going to deliver as a project output and how will I evaluate it?

I will have built a neural network solution in python that classifies galaxies, present a discussion of how I found the process, and give a review of my best performing model. I will also talk about the context surrounding machine learning techniques and their impact on citizen/community science projects.

I also want to have a discussion of how easy this task was to do using hardware not necessarily specialised for the task further demonstrating the power of these technologies, what measures and compromises I undertook to get it to work, or if the solution fails how far I am able to get and what went wrong.

As part of the project, a gradual approach will be taken, starting by implementing and testing the CNN model on simpler, more manageable image datasets before progressing to the more complex galaxy image classification task. This phased approach serves two key purposes:

1. It allows for an incremental evaluation of the CNN model's performance and the identification of potential challenges or limitations that may arise during training, especially with regards to hardware limitations.
2. It enables me to further gain familiarity with the process of building and optimising CNN models.

Once satisfactory results have been achieved on the simpler image classification tasks, and the necessary skills and experience have been acquired, the project will then transition to the primary objective of classifying galaxy images.

Throughout the process I shall analyse the model's predictions, identify any potential biases or errors, and refine the model accordingly. I shall accomplish this by evaluating the trained model's performance on validation and testing datasets using appropriate metrics, such as accuracy, precision, recall, and F1-score.

Analysis of likely impact & Discussion of wider implications

Legal

I shall ensure that the data used for the training and testing of the CNN model is obtained legally and does not violate any copyright or intellectual property rights. The datasets used in this project aren’t connected to any individuals and are either simple images or astronomical phenomena, so they are free of ethical or security considerations relating to personal data protections. This project also involves no human participants.

Social

Projects like Galaxy Zoo, that this project is building a model based on, are not just about data collection but also serve as platforms for public engagement in science, education, and citizen participation. The success of Galaxy Zoo lies in the collaborative effort of thousands of volunteers, fostering a sense of community and shared purpose. Automating the classification process could diminish these important aspects, potentially leading to a loss of interest and involvement from the general public in scientific endeavours.

Ethical

All AI models are capable of containing and perpetuating biases, so efforts should be taken to minimise this within this project. CNNs, like other AI models, are often considered "black boxes," making it difficult to understand the reasoning behind their predictions. This lack of transparency could raise ethical concerns, especially if the model's classifications are used to inform scientific decisions or allocate resources.

Professional

Scientific research and machine learning models should be reproducible and subject to peer review. Properly documenting the methodology, data sources, and model architecture will enable others to validate and build upon the work. This project shall adhere to professional codes of ethics and best practices in machine learning and data science, particularly those presented by the BCS Code of Conduct.

Part 2: A Review of Literature

This literature review is broken up into two sections. Firstly, a collection of papers from scientific journals looking at convolutional neural networks and their use in galactic astronomy. And secondly, a mix of sources talking about citizen science initiatives in general and their use in the context of the intersection of astronomy and computer science, with a view to look at how new technologies have impacted these efforts.

Convolutional Neural Networks

These two papers discuss the merits of Convolutional Neural Networks for image classification. Both papers were found via the Open University's library, so while I hadn't heard of the journals before they are peer reviewed and are up to date, having been published within the last 4 years.

Wang, P. *et al.* (2020) in their paper 'Comparative analysis of image classification algorithms based on traditional machine learning and deep learning', perform a comparison of the performance of Support Vector Machines and Convolutional Neural Networks. The paper discusses the background of why machine learning of images is important in our modern age and gives a brief overview of some of the different types of models used in a clear and organised way making the paper easy to follow.

They then performed tests with two different machine learning algorithms, a more traditional Support Vector Machine and a modern Convolutional Neural Network. They trained and tested them both with two datasets: the MNIST handwritten digits dataset and the Corel1000 picture set. They tested to see how both algorithms performed against differing size datasets, different picture sizes and the number of categories of images. They noted the accuracies and the time taken for both algorithms. They showed that for smaller scale datasets traditional machine learning techniques have a notable advantage of being quicker to train, but their accuracy performance is surpassed by the newer convolutional neural network based models when faced with larger scale datasets.

This paper gives a good example of the power of convolutional neural networks for image classification over other technologies, and makes a good case for this project's use of a CNN for classifying galaxy images as they are a reasonably large scale dataset and also demonstrate the use of the MNIST dataset that is one i've used before and makes for a good initial testing dataset for this project.

Lv, Q. *et al.* (2022) in their paper 'Deep Learning Model of Image Classification Using Machine Learning', demonstrate their own proposed CNN model against some other well known models. This paper was published in the journal 'Advances in Multimedia'. I had hoped it would be more focused on how a range of CNN models compared to each other and have some practical advice on how to optimise my own models for use in this project. However, I found this paper harder to follow, thanks in part to some grammatical errors, but also as it is mainly focused on how they optimised their own proposed model and is rather technical with mathematical concepts I haven't covered fully.

Machine Learning in Galactic Astronomy

I also looked at a number of papers discussing the use of machine learning specifically in relation to galaxy classification. While the actual astrophysics content of these papers is beyond the scope of this project and my expertise, they use data drawn from Galaxy Zoo at various points in its development.

The first two papers are from the monthly notices of the Royal Astronomical Society, a highly respected 200 year old institution. I feel they tell a compelling narrative of how the progress of machine learning has influenced the field of astronomy over time:

In their 2010 paper, 'Galaxy Zoo: reproducing galaxy morphologies via machine learning', Banerji, M. *et al*. discuss a study to use machine learning to classify galaxies, the network they used seems to be a standard multi layer

perceptron to distinguish between three classes of object from the Sloan Digital Sky Survey (SDSS) data. They were able to get classifications for almost one million objects and were able to reproduce the Galaxy Zoo human classifications of this dataset with better than 90% accuracy for their three morphological classes. Their conclusions focus on their view that such huge labelled datasets are excellent use cases for, and demonstrate the power of, neural networks, but also that with these datasets, incompleteness and human biases are issues that will need work to overcome.

By 2021 Cheng, T-Y. *et al*.'s paper, 'Galaxy morphological classification catalogue of the Dark Energy Survey Year 3 data with convolutional neural networks', now talks about how while Galaxy Zoo's techniques of using volunteers can not keep up with the sheer amount of data generated by modern surveys, they can still be used to help train a model to then classify images from another source. It also continues to talk about some of the flaws in Galaxy Zoo and how redshift introduced errors humans struggled to account for. This paper definitely goes into more of the physics detail especially on the topic of redshift which I only have a passing knowledge of. But does talk about using insights gained from CNN models to create a new classification catalogue for galaxies. Showing a move from using these machine learning models to do what humans can do but faster, to using them to discover new insights that arise from the scale of these large datasets when seen by new AI systems.

The third paper I looked at was Wuyu Hui *et al.'s* 2022 paper 'Galaxy Morphology Classification with DenseNet.' It is published in the Journal of Physics: Conference Series from the Institute of Physics. This source is peer reviewed but the publication has had some issues in the past retracting a number of articles in 2022. However, I am giving it the benefit of the doubt in this case as the paper is still available. The paper applies a variety of different CNN models on the Galaxy10 DECaLS dataset and compares the results. Their best performing model used DenseNet121 and I thought it would be interesting to see if I could get this model to run in my project. I found this paper as it was given as an example of a study that used the Galaxy10 DECaLS dataset from AstroNN, a resource I also planned to use in this project. The study itself is relatively simple, but I think it is a good source of inspiration for a small project looking to compare CNN models with a single dataset and a good point of comparison for this project.

Citizen Science & Galaxy Zoo

As I was wanting to touch on how improvements in machine learning affects citizen science efforts, I've looked at some sources that discuss citizen science and also focused on the Galaxy Zoo project as a bit of a case study.

The book 'Citizen Science: Innovation in Open Science, Society and Policy' published by UCL Press (2018), presents a case for citizen science and how "The current increase in citizen science shows clearly the societal desire to participate more actively in knowledge production, knowledge assessment and decision-making.". The book itself is a huge text at nearly 600 pages presenting insights and arguments from teams around the world, leading to a series of recommendations for how citizen science can be harnessed, encouraged and supported. And concluding that "Citizen science contributes to innovation in science itself, and indeed a genuine science outcome is a main principle of citizen science projects" and "citizen science projects can be designed to involve many volunteer contributors in large collaborative projects, such projects have the possibility to pursue research that could not be done otherwise.". It is an interesting read, though I include it mostly as background for this project's context.

The paper 'Crowdsourcing citizen science: exploring the tensions between paid professionals and users' by Woodcock, J. *et al*. (2017) explores the relationship between paid labour and unpaid users within the Zooniverse platform. It is published in the '*Journal of Peer Production'*. I had not heard of this journal before and could not find much information about it. However, as it talks about the Zooniverse platform it is especially relevant if I am to include a section in this project outlying tensions with citizen science.

Galaxy Zoo is a groundbreaking citizen science project launched in 2007 that revolutionised astronomical research. It engaged members of the public to classify galaxies based on their morphological characteristics using images from various sky surveys. By harnessing the power of crowdsourcing, Galaxy Zoo has classified millions of galaxies, far surpassing what professional astronomers could achieve alone. The project has since evolved into the Zooniverse, an online platform that hosts numerous projects where volunteers contribute to real research in fields such as biology, climate science, medicine, history, and more. It also serves an important role in science education and public engagement with research. The project is well documented on the Zooniverse Website, with regular Galaxy Zoo updates still being posted to their blog. (Zooniverse, 2024).

Over the years BBC news has mentioned Galaxy Zoo and the Zooniverse platform a number of times. I was unsure how much I would make use of the BBC as a reference in this project. However it is a good source for how citizen science projects are seen by non scientists and the media, as the BBC is a trusted institution regarded as trying to be as unbiased as possible. In 2007, BBC news talked about the new Galaxy Zoo project, giving an overview of what the team's hopes were for the project back when they were starting, and provided a good summary of how the classification worked, (BBC News, 2007). Then again in 2013 while reporting on a boom in citizen science in the UK (BBC News, 2013). And in 2017 while talking about Galaxy Zoo's 10 year anniversary, they featured a story on a new kind of astronomical phenomenon discovered by one of the citizen scientists and even attached a link to the research paper on the new objects, showing that this kind of project captured the public's imagination (BBC, 2017)(Keel, W. *et al*., 2016).

SETI@home

Another project that combined community participation with new technologies to aid scientific efforts was SETI@home. It was a pioneering example of how advances in technology transformed citizen science, enabling large-scale participation in scientific research. Launched in 1999 by the Search for Extraterrestrial Intelligence (SETI) project, SETI@home leveraged the growing availability of personal computers and the growing internet to involve ordinary citizens in the search for extraterrestrial signals.

People could participate by running a free program that downloaded and analysed radio telescope data using their computer's idle processing power. This decentralised approach massively expanded the computational capacity available to scientists, far surpassing the limitations of traditional supercomputers at the time.

The two original goals of SETI@home were:

1. To do useful scientific work by supporting an observational analysis to detect intelligent life outside Earth
2. To prove the viability and practicality of the "volunteer computing" concept

The second of these goals is considered to have succeeded. The current BOINC (Berkeley Open Infrastructure for Network Computing) environment, a development of the original SETI@home, is providing support for many computationally intensive projects in a wide range of disciplines. Making this particularly relevant to the computing domain of this project. While the first of these goals has to date yielded no conclusive results: no evidence for ETI signals has been shown via SETI@home. (In other words no sign of aliens but still a fun project).

SETI@home is an interesting project as it touches on the ideas of how changing technology impacts the ways the public can interact with science projects and has led to some real changes in how citizen science projects can be done, but I decided to focus more on Galaxy Zoo due to its ties with the dataset I intended to use.

Part 3: A talk about Citizen Science

Citizen science projects have allowed non-professionals to contribute meaningfully to important discoveries. These projects offer numerous benefits to both the scientific community and the public. They can enable researchers to process vast amounts of data that may otherwise be impractical or costly for small teams to handle, they can help with increasing public understanding and engagement with scientific topics, and additionally these projects can play a significant role in educational settings, inspiring the next generation of scientists.

One prominent example of citizen science in astronomy is the Galaxy Zoo project. Launched in 2007 this project invited members of the public to classify galaxies based on their visual characteristics, a task that humans initially performed better than computers. The current Zooniverse Project includes tools for teachers to incorporate projects into their curriculum, allowing students to engage directly with professional research (Zooniverse, 2024). These experiences, often coupled with field trips to observatories or science centres, can spark long-lasting interest in STEM fields among young learners. This educational aspect not only enhances scientific literacy but also helps to build a diverse pipeline of future researchers and science enthusiasts.

Galaxy Zoo demonstrated the power of crowdsourcing in scientific research, with participants classifying millions of galaxies and contributing to numerous peer-reviewed publications. Galaxy Zoo has led to unexpected discoveries, such as Hanny's Voorwerp, an astronomical object found by Dutch schoolteacher Hanny van Arkel that at the time was unknown to science (Keel, W. *et al*., 2016). The success of this project highlighted two key points: The public's enthusiasm for participating in scientific endeavours, And the potential for distributed human intelligence to tackle large-scale data analysis problems

As technology has advanced, the role of citizen scientists has evolved. Machine learning techniques, particularly Convolutional Neural Networks (CNNs), have become increasingly proficient at tasks like image classification (Wang, P. *et al.*, 2020) reducing the usefulness of human volunteers in these areas.

This shift does not necessarily diminish the importance of citizen science; rather, it transforms the nature of public participation in research.

The intersection of citizen science and technological advancement has taken various forms over the years. The SETI@home project, launched in 1999, allowed individuals to contribute their computer's idle processing power to analyse radio telescope data (Anderson, D.P. *et al.*, 2002). This early initiative paved the way for more direct forms of technological participation in science.

Today with the increasing power of consumer-grade hardware, individuals can now not only contribute data or spare runtime to perform analysis for others but also develop and train sophisticated machine learning models on their own personal computers. The release of consumer-grade GPUs with significant computational power, such as NVIDIA's RTX series, has made it possible to train complex neural networks at home. This commercialisation of AI technology allows for a deeper level of engagement with cutting-edge research methods.

By developing a CNN for galaxy classification on a home PC, this project bridges the gap between traditional citizen science initiatives and professional research tools. It demonstrates how individuals can now engage with and contribute to important scientific and technological advances using consumer hardware, potentially opening new avenues for public participation in scientific discovery.

Part 4: An Exploration of Convolutional Neural Networks

Convolutional Neural Networks (CNNs) can serve as an excellent starting point for those venturing into machine learning and AI. They offer a relatively intuitive structure that mirrors how the human visual system processes information, making them easier to grasp conceptually compared to some other machine learning algorithms. CNNs also excel in computer vision tasks, allowing learners to quickly see tangible results and appreciate how these technologies have real world uses in key application areas in modern AI: image classification, object detection, and facial recognition to name a few.

While CNNs are particularly suited for these image-related tasks, understanding them also provides a solid foundation for exploring other machine learning technologies. Additionally, learners who later branch out to explore other important machine learning algorithms will find that the concepts of layers, weights, and backpropagation introduced in CNNs translate well to other neural network architectures.

I worked on CNN image classifiers during my studies with the Open University and have gained a good understanding of machine learning techniques. However, throughout my studies the bulk of the code was provided pre-written and I was only tasked to manipulate it and discuss the results. The coding environment was preconfigured and ran on a cloud system, which was great for learning but left me wanting more hands-on experience. I decided it was time to learn how to set up my own environment and run CNNs on my personal hardware. This would allow me to apply these techniques to tasks beyond my university studies and give me more control over the entire process.

Before going further, my computer hardware environment configuration is as follows: the system is 64-bit windows 10, the processor is AMD Ryzen 7 7800X3D CPU@4.20 GHz, the memory is 32.0 GB RAM, and the graphics card is Nvidia Geforce Rtx 3070 (Though the GPU does not end up getting used in this project as discussed later). This PC is admittedly above specifications of the average home PC, however, I do not believe it is vastly above what could be expected for an enthusiast in the technology space, and should not cause an issue for this project in that regard.

Python & Anaconda

I had previously explored Python independently but I had only used the standalone version from Python's own website and its default Integrated Development Environment (IDE), IDLE. While looking for instructions on how to install TensorFlow, I kept coming across instructions to download it via Anaconda, a system I had not heard of before. A quick Google search revealed that it was perfect for what I wanted to do. It is a platform that includes Python and many data science libraries, making it ideal for machine learning projects. In addition it handles the downloading and installation of Python Libraries while also ensuring compatibility between all the various parts.

I started by downloading the Anaconda distribution from the official website. The installation process was straightforward and once all set up it opens to the Anaconda Navigator, which provides a graphical interface to manage its components and allows you to easily download additional software.

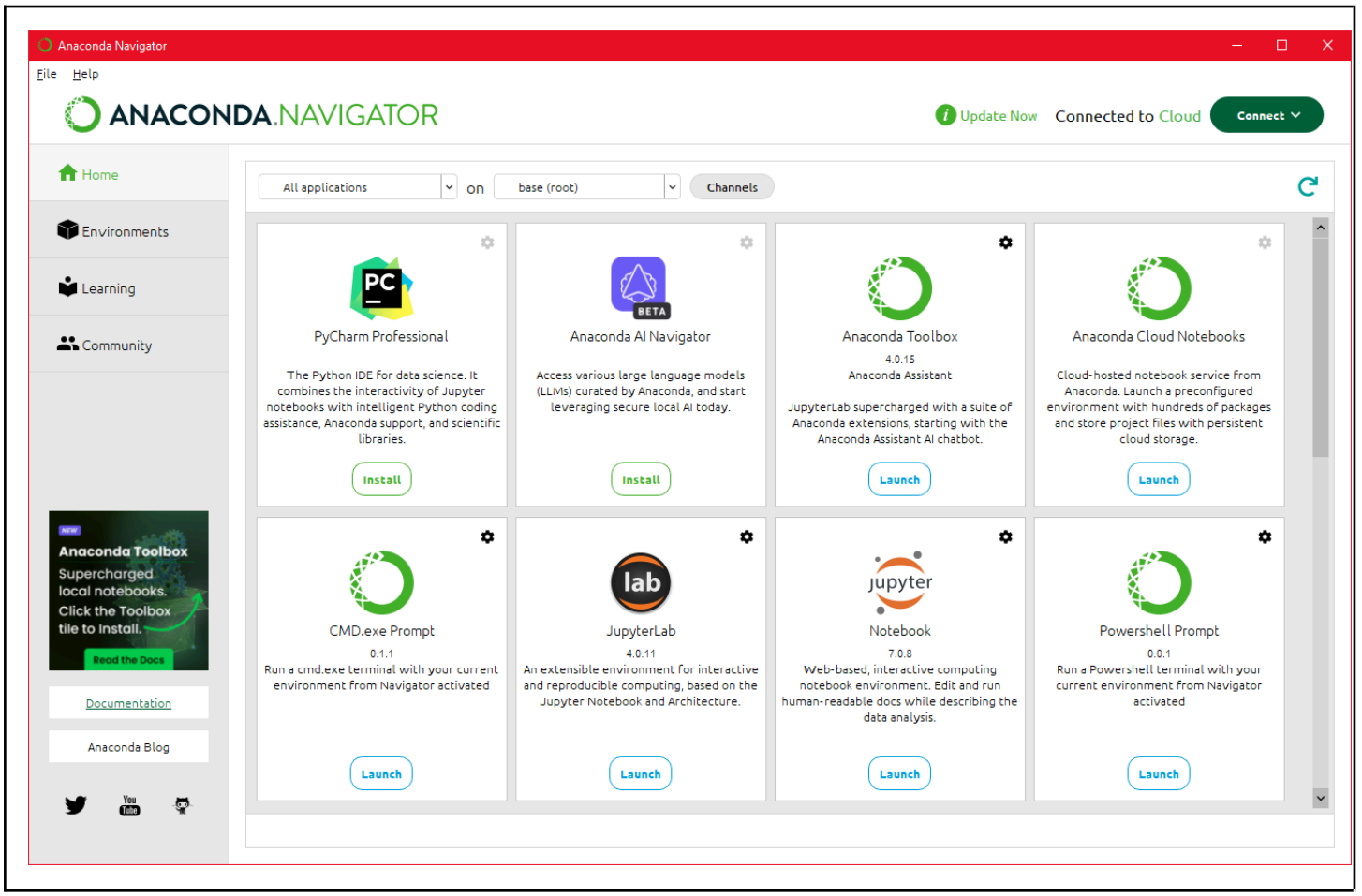


Figure 1: Anaconda Navigator

Among these options are a couple of IDE options. Initially I tried using Spyder as my Python IDE but after experimenting with it for a while I opted to return to using Jupyter notebooks, as that was what I used the most during my studies. And I particularly like its cell based interface allowing for both code cells and markdown cells to be in the same document for easy readability.

Anaconda's environment management system allows you to create isolated workspaces for different projects, each with its own set of dependencies designed to avoid conflicts with other projects. Within this system Anaconda also handles the installation process of python libraries and automatically manages the dependencies. For example I knew I wanted to use a dataset from tensorflow_datasets, so all I had to do was select that package to be installed and Anaconda automatically set about fetching its dependencies. Additionally, when any others are installed, Anaconda makes sure to download the latest version that is compatible with what is already installed. This streamlined approach not only saves time but also reduces the likelihood of compatibility issues down the line, especially for those who are new to setting up their own coding environments.

MNIST

To check everything was set up correctly, I recreated a simple CNN classifier by adapting code that I had already used previously in my studies and that used the MNIST handwritten numbers dataset. This classifier aims to identify what handwritten digit is in each of images from 0-9. This dataset makes an ideal quick benchmark due to it being relatively simple, consisting of small 28x28 pixel grayscale images. The dataset was split into 3 parts to create training, validation, and test subsets.

The CNN model itself was relatively simple but effective for basic image classification tasks. It had a couple of convolutional layers to extract features, max pooling layers to reduce spatial dimensions, and then dense layers to perform the final classification.

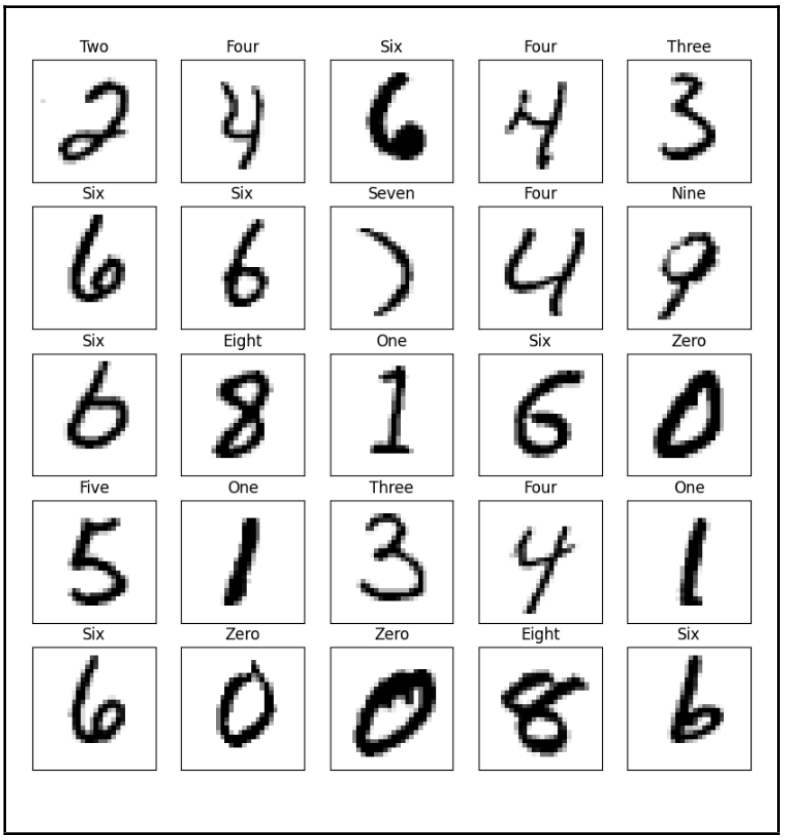


Figure 2: Examples from the MNIST dataset

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 flatten (Flatten)           (None, 1600)              0

 dense (Dense)               (None, 64)                102464

 dense_1 (Dense)             (None, 10)                650

=================================================================
Total params: 121,930
Trainable params: 121,930
Non-trainable params: 0
_________________________________________________________________
```

Figure 3: Structure of the CNN model used for the MNIST dataset

This simple model only took a little over 3 minutes to train on the CPU and even with training for only 5 epochs it achieved an accuracy result of 97% on the test set.

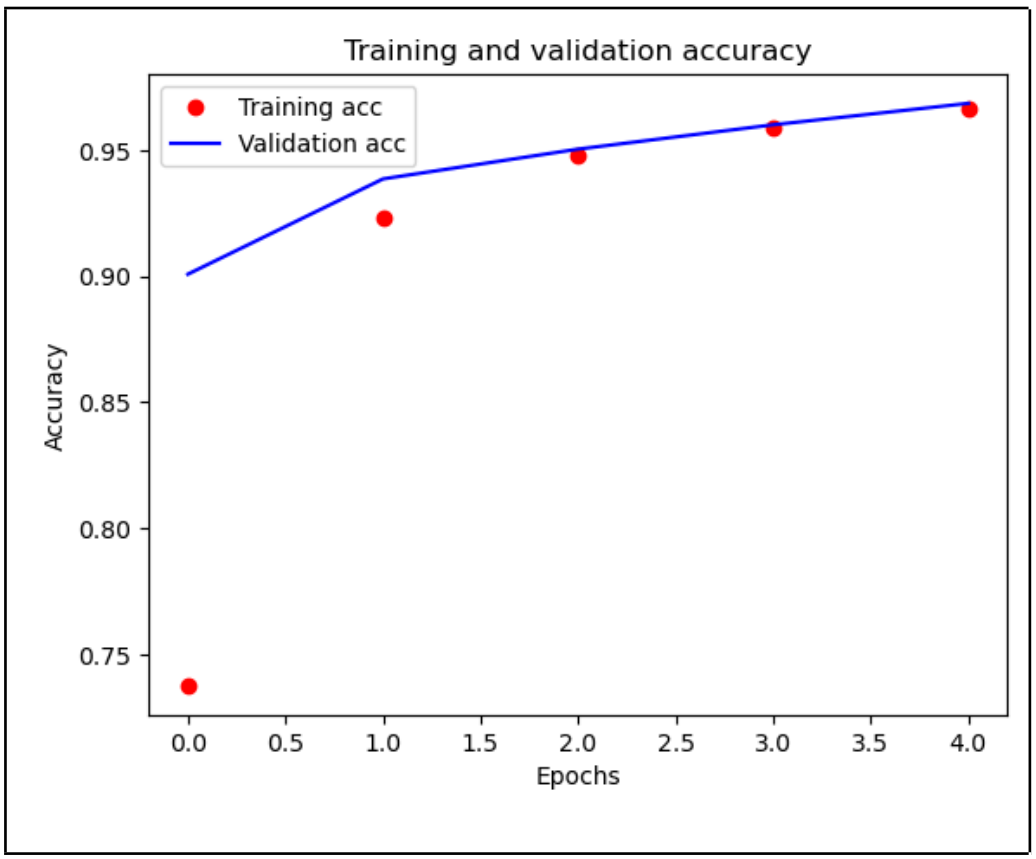


Figure 4: Graph showing Training & Validation Performance for the MNIST model

While not a particularly interesting test, it does confirm that I have set everything up correctly and that I am able to build and train a neural network on my pc.

CIFAR-10

After seeing that Python, Jupyter, TensorFlow, and other libraries were properly set up, the next stage was to see how my PC handled a slightly more complex dataset. For this I opted to use the CIFAR-10 dataset also available from tensorflow datasets. It is a collection of 60,000 colour images, each 32×32 pixels, split among 10 classes. It is a step up from the MNIST dataset and again is widely used as a benchmark for testing and comparing machine learning algorithms.

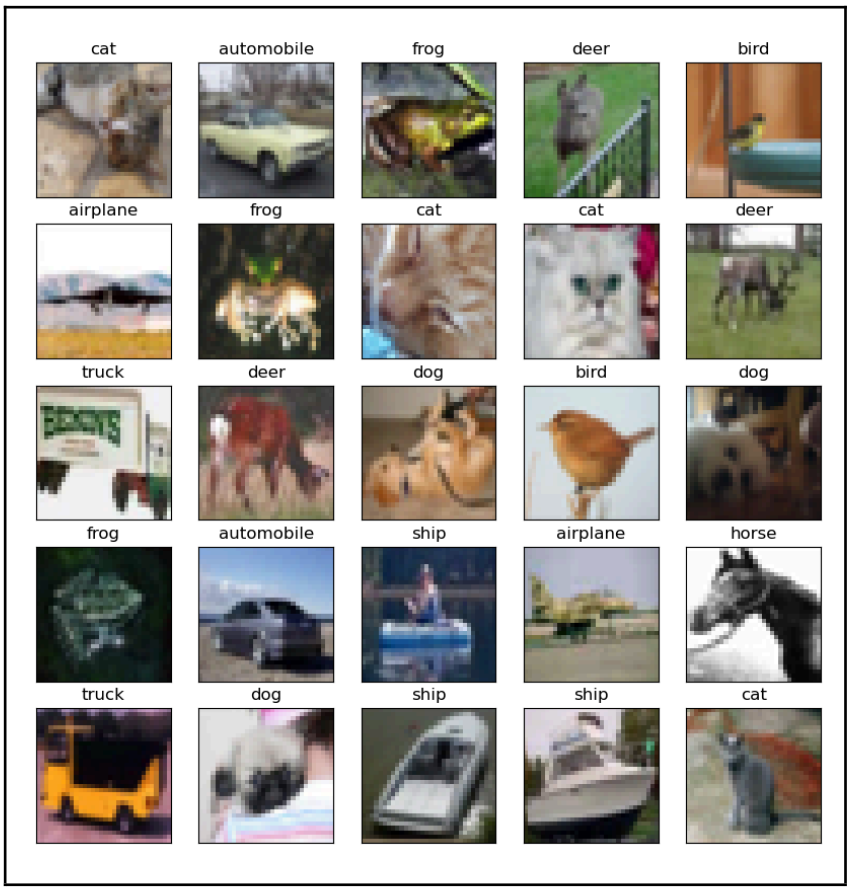


Figure 5: Grid showing CIFAR10 dataset example images and their labels.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d_1 (Conv2D)            (None, 32, 32, 32)        896

 max_pooling2d_1 (MaxPooling  (None, 16, 16, 32)        0
 2D)

 conv2d_2 (Conv2D)            (None, 16, 16, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 8, 8, 64)          0
 2D)

 conv2d_3 (Conv2D)            (None, 8, 8, 64)          36928

 max_pooling2d_3 (MaxPooling  (None, 4, 4, 64)          0
 2D)

 flatten_1 (Flatten)          (None, 1024)              0

 dense_1 (Dense)              (None, 64)                65600

 dropout_1 (Dropout)          (None, 64)                0

 dense_2 (Dense)              (None, 10)                650

=================================================================
Total params: 122,570
Trainable params: 122,570
Non-trainable params: 0
_________________________________________________________________
```

Figure 6:Structure of the CNN model used for the CIFAR-10 dataset

This time I used a slightly deeper network which uses three convolutional layers with increasing filter sizes, and trained it for 10 epochs. After training it got an accuracy of 73% with the training performance graph showing signs that the model had not finished learning.

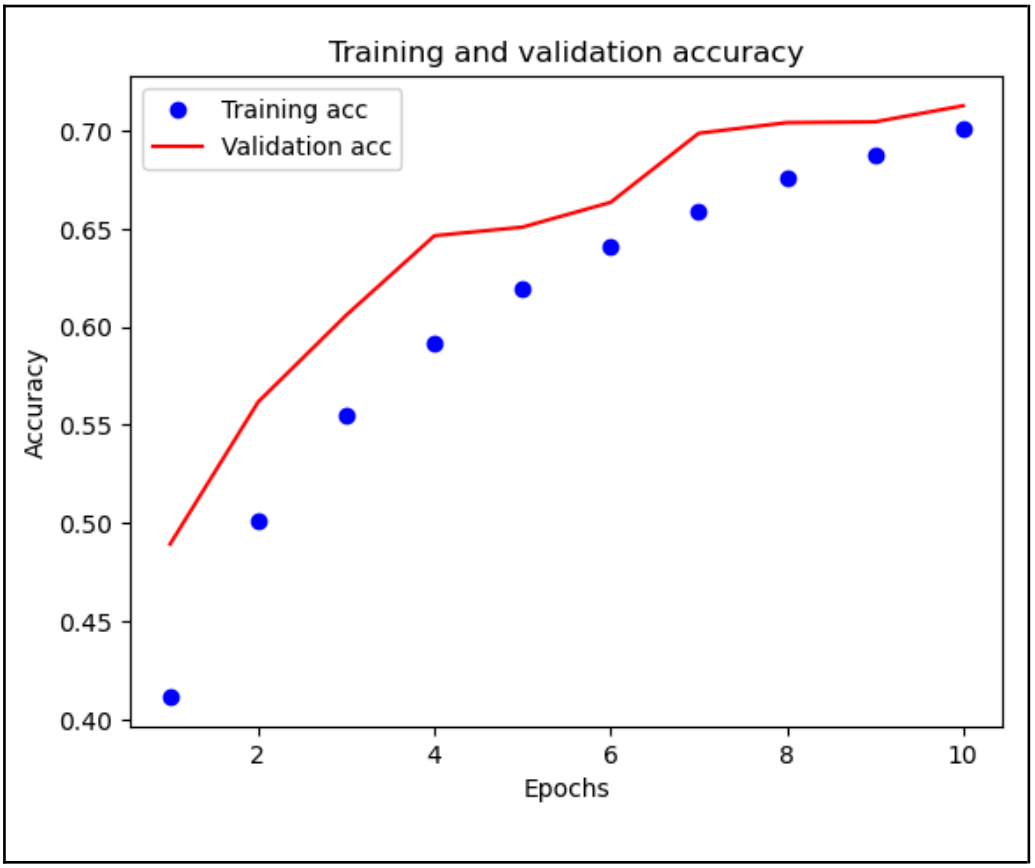


Figure 7: Graph showing Training & Validation Performance for the CIFAR10 Model

I also used this test to try out a couple of ways to view the output: a sample of images and the model's predictions for them and a confusion matrix, to see if they would display the same as I had seen previously on the OU system. I did notice that even with this simple model architecture and small image sizes the model still took 12 minutes to train on the CPU. And with the proposed Galaxy image dataset being even larger still, I wanted to see if I could utilise the processing power of my GPU instead of running these on the CPU.

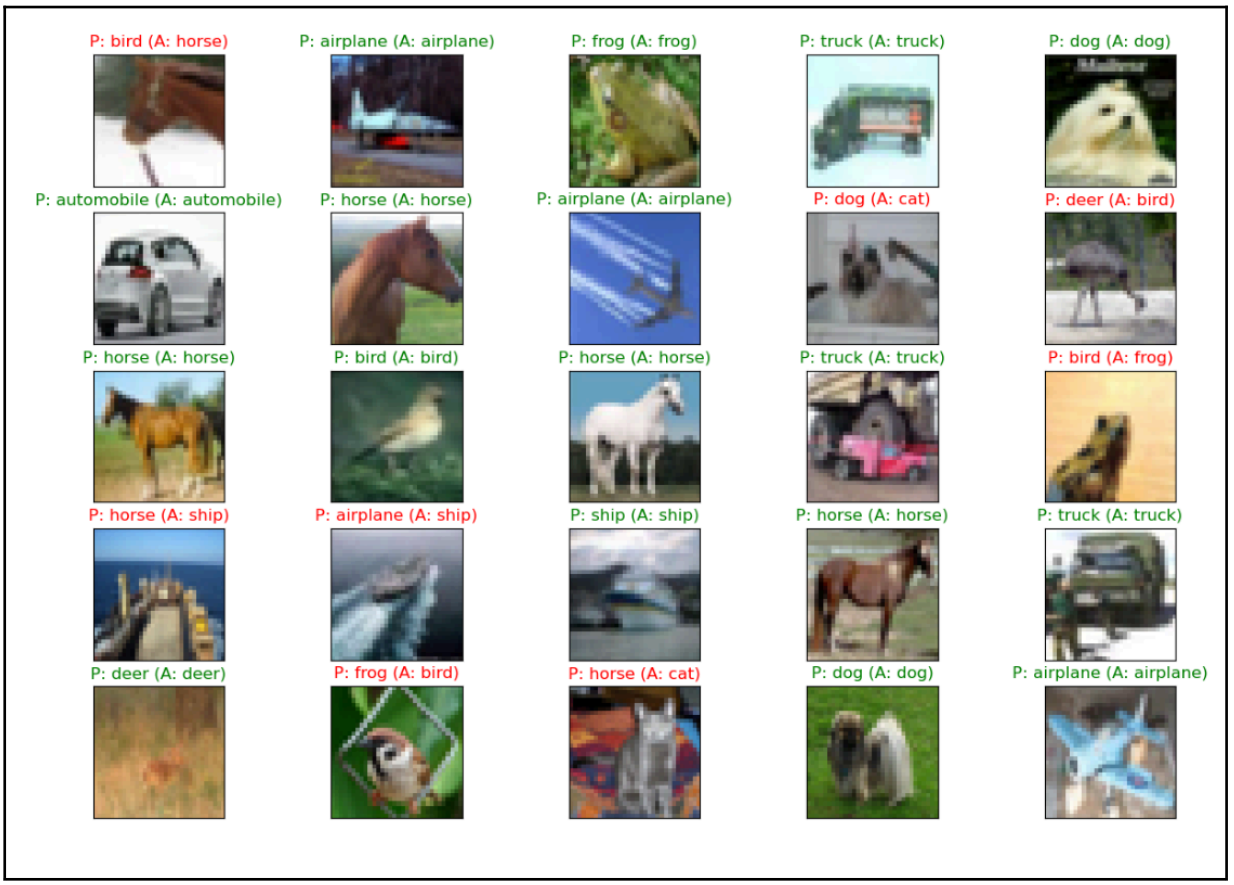


Figure 8: Grid showing model predictions

Confusion matrix

		Predicted labels									
		airplane	**automobile**	**bird**	**cat**	**deer**	**dog**	**frog**	**horse**	**ship**	**truck**
Actual labels	**airplane**	799	17	40	14	11	2	5	9	59	44
	automobile	15	839	1	10	7	0	11	4	24	89
	bird	95	5	550	98	100	37	65	26	17	7
	cat	22	7	61	609	66	92	68	32	27	16
	deer	23	2	51	76	703	17	54	59	11	4
	dog	12	2	54	305	46	481	22	58	13	7
	frog	6	5	29	85	41	5	813	6	8	2
	horse	17	2	29	55	75	32	4	770	2	14
	ship	59	26	6	15	9	1	4	4	860	16
	truck	26	81	1	27	3	1	7	10	17	827

Figure 9: Confusion Matrix for CIFAR10

Can I use the GPU?

The increasing complexity of datasets and models had led to longer training times, particularly when dealing with a higher number of epochs. So the next steps were to see if I could get these models trained using the GPU in my machine instead of the CPU and measure how that affects training time. However, I quickly discovered that configuring TensorFlow to utilise a GPU on a Windows 10 system is significantly more challenging than setting up the CPU version.

The contrast between the two setups is stark: The CPU version of TensorFlow is remarkably user friendly, and functions seamlessly without any additional configuration, while the GPU version presents a considerably more complex installation process.

From what I have been able to discover, while I have a compatible NVIDIA GPU, the setup requires several additional software components, and each must be compatible with the others. Two of the components to be installed are the CUDA Toolkit and the additional cuDNN software from NVIDIA. Both of these must be the correct version to ensure compatibility with each other, the specific TensorFlow version, and the operating system. This is a big stumbling block when using Anaconda to install TensorFlow and automatically fetch the version that is compatible with the rest of the packages in the environment being worked in. For the GPU software, it appears I have to select a single version. This might be fine for a single Anaconda environment while I am working on it, but if later down the line I need a different version, the prospect of having to reconfigure the GPU software manually is daunting for me as someone new to this.

The level of difficulty in setting up GPU support would vary depending on technical expertise, but I find it frustrating that GPU compatibility does not work seamlessly out of the box. Drivers for normal use of a GPU are easy to install with the GPU's own software able to keep it up to date. This added complexity creates a barrier for less experienced users, preventing them from fully utilising the power of their hardware. The situation is disappointing, as it limits the accessibility of advanced machine learning techniques to a broader audience who could benefit from faster training times and more efficient model development.

Despite the promise of significant performance improvements for large models or datasets when using the GPU, after a couple of failed attempts to get my GPU to work with Tensorflow I opted to continue on with just using the CPU.

Part 5: Galaxy Classification

I have always had an interest in astronomy, and when it came time to choose a topic for the final project I thought this would be a great opportunity to combine my personal interests with my academic work. So I decided that I would try and make a Convolutional Neural Network to classify a dataset consisting of images of galaxies, calling back to my experience at school doing classifications for Galaxy Zoo. I also considered doing something with images of stars but felt galaxies were more visually interesting. Another topic I considered were images of asteroids but I struggled to find any usable image datasets for them.

AstroNN & Galaxy10 DECals

After settling on wanting to use a galaxy image dataset I spent some time looking for labelled galaxy image data. Galaxy Zoo originally got their data from the Sloan Digital Sky Survey (SDSS) so Initially I looked at getting the dataset directly from the SDSS. Unfortunately it is a huge dataset on the order of hundreds of gigabytes, so obviously that was off the table. Galaxy Zoo do have their own dataset available to download that contains category labels, however they are not attached to the images of the galaxies but instead to an object identifier within the SDSS data. So I would have to manually join the two datasets together. The scale of the datasets made this unachievable with my resources. I would likely need an entire separate harddrive just to store the two datasets.

Luckily I stumbled upon the AstroNN python library while looking for an alternative dataset on Kaggle. AstroNN is an open-source python library providing tools for applying deep learning techniques to astronomy. It includes tools for working with astronomical data formats such as APOGEE, Gaia and LAMOST data and performing tasks like spectral analysis and stellar parameter estimation. For this project however, I am less interested in these features, and instead I am wanting to just use the library as a convenient way to get access to AstroNN's 'Toy' Datasets.

These datasets are the Galaxy10 dataset, and the Galaxy10 DECals dataset. The original Galaxy10 dataset was created with an early Galaxy Zoo release that used the SDSS images mentioned previously, while Galaxy10 DECals is an updated dataset that uses data from a later Galaxy Zoo release that instead uses images from the Dark Energy Camera Legacy Survey (DECals). The Galaxy10 DECals images have much better resolution and image quality. The Galaxy10 DECals dataset is 2.54 GB, containing 17736 256x256 pixels coloured galaxy images separated in 10 classes.

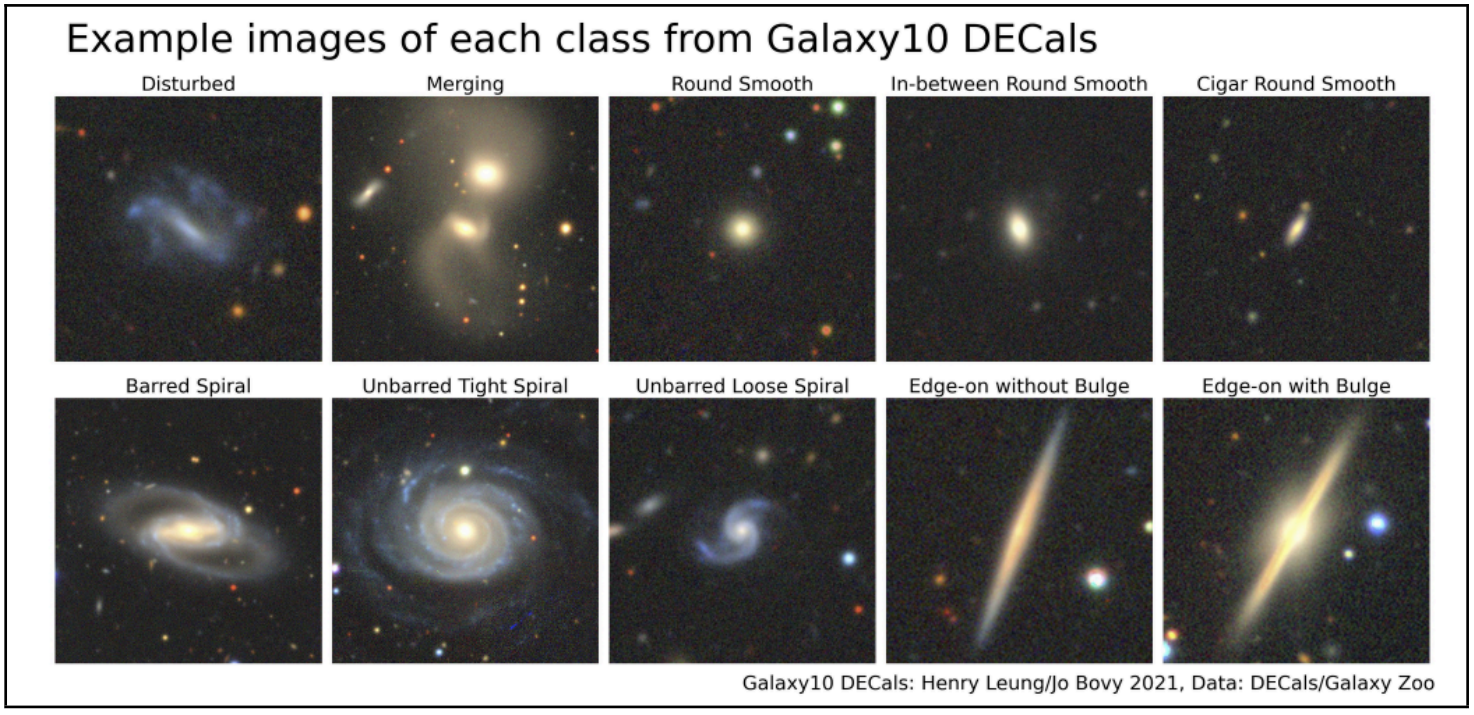


Figure 10: Example images of each class from Galaxy10 DECals dataset

The Galaxy10 DECals dataset is perfect for my needs. It has already joined the galaxy image data with the Galaxy Zoo labels, and has cleaned the data ensuring there are no missing labels or images in the dataset. However, this does make the dataset less representative than a real random selection of telescope image data from a sky survey, making any system trained on the data contain these biases and potentially less useful on other datasets. Additionally the dataset has already had some measure of downscaling applied to the galaxy images, this was done to both make the dataset easier to host and download online and also to make the dataset easier to manage and less demanding on computer hardware. This downscaling step may have been interesting to apply myself but otherwise this dataset is perfect for the needs of this project.

Setup

AstroNN was initially a bit of a stumbling block in this project. Initially I attempted to download the dataset directly from the astroNN website hoping to forgo using the astroNN library as I did not need the rest of the features, however I struggled to open the h5 filetype. After deciding to then use astroNN just for the purposes of downloading and opening the data, I also had a few issues getting the astroNN library working as it is not as well documented or frequently updated as I would have liked, leading to dependency and compatibility issues, as the latest astroNN release is not compatible with the latest TensorFlow release and not directly installable through Anaconda. But after a bit of trial and error I managed to find a way of installing astroNN through pip and then an older version of TensorFlow through Anaconda.

The next step was to import and open the dataset with Jupyter Notebooks to confirm everything was set up correctly. I also imported a variety of functions for later use.

```
import numpy as np
from astroNN.datasets import load_galaxy10
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.metrics import classification_report
from sklearn.utils.class_weight import compute_class_weight
from collections import Counter
import time
import json
```

```
# Obtain the Galaxy10_DECals dataset
images, labels = load_galaxy10()
```

```
C:\Users\emily\.astroNN\datasets\Galaxy10_DECals.h5 was found!
```

Figure 11: Code and output for code imports and the loading of the Galaxy10 DECals dataset

Fortunately the dataset was already in a convenient labelled format similar to the previous two I used making it easy to work with. My first test was to open it and view the data getting a feel for how it was distributed across the ten classes.

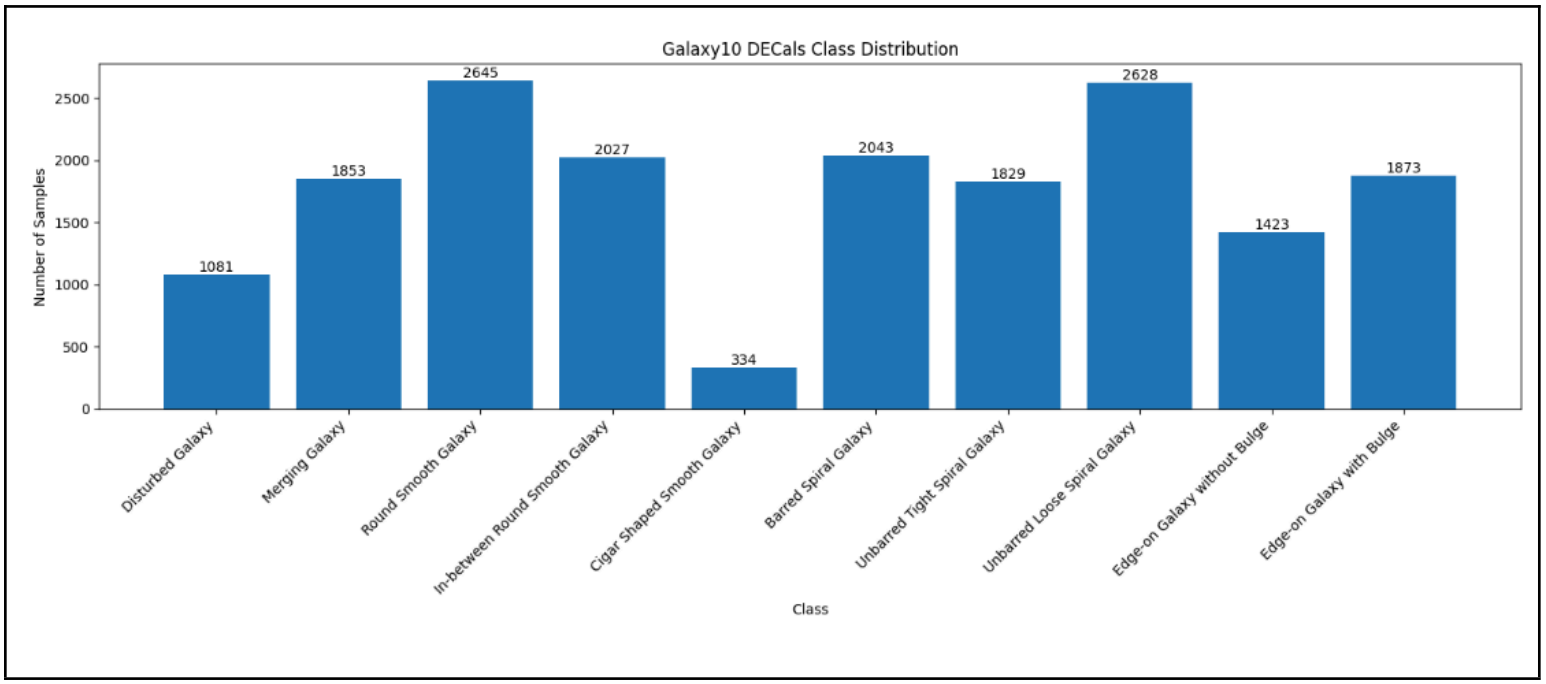


Figure 12: Class distribution of the Galaxy10 DECals dataset

I made a chart showing the distribution of the 10 classes just to double check that it matched the information given on the astronNN website. The dataset is rather unbalanced with the two most represented classes, Round Smooth and Unbarred Loose Spiral Galaxies, each having around 2600 samples. While the Cigar Shaped Smooth Galaxy class only has 334 samples. This class imbalance could cause issues for the models as they may struggle to accurately predict the minority classes.

I also decided to view a few samples from the dataset to make sure the dataset was properly shuffled before splitting into subsets for training.

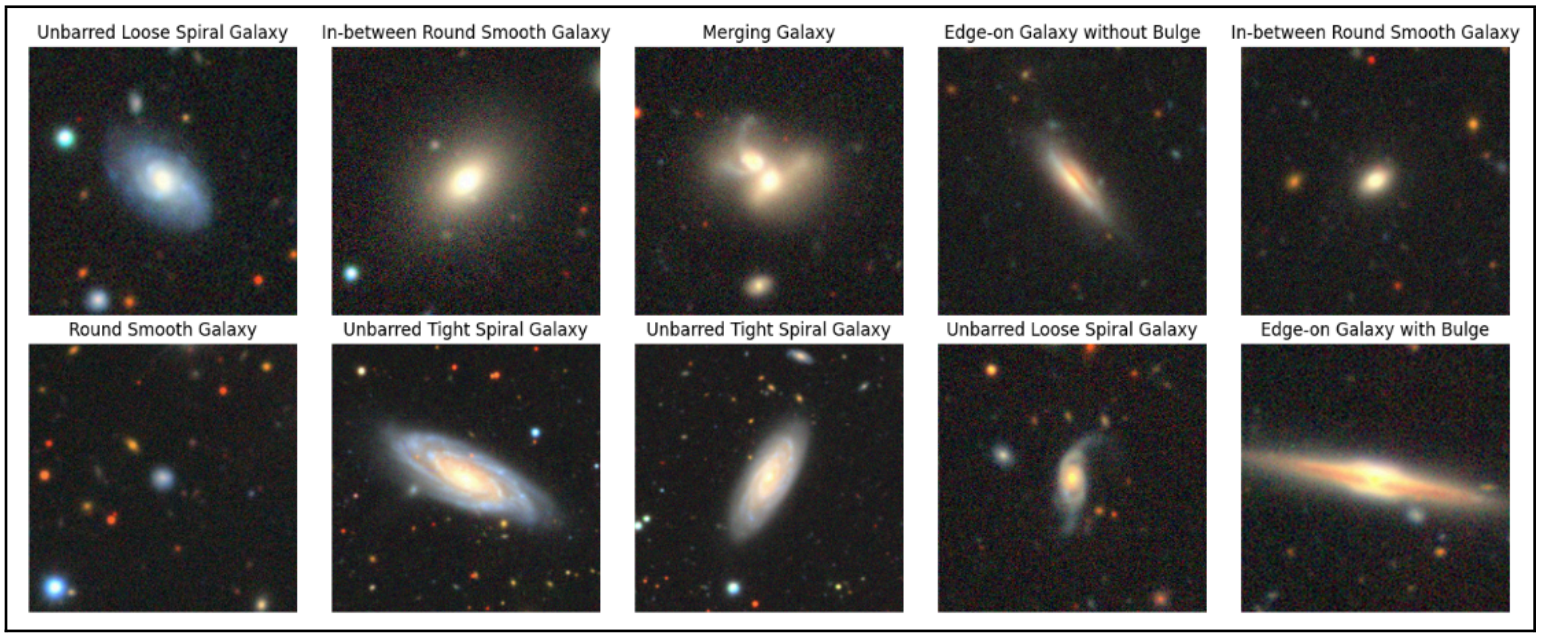


Figure 13: Random sample of images from the Galaxy10 DECals dataset

The Models

Now that I had the dataset imported and read for use, I checked that the training and test datasets had approximately the same distribution of classes. (A validation subset was also made as part of the training process for each model taking 20% of the training set.) I also added a catch for each model that looked to see if model performance had stopped improving, if after five epochs a model's validation accuracy had not improved, training would stop and the model would roll back to the best model weights it had found so far. I put this in as I was wary of how long these models might take to train on the CPU and wanted to ensure the models were not running for any longer than necessary.

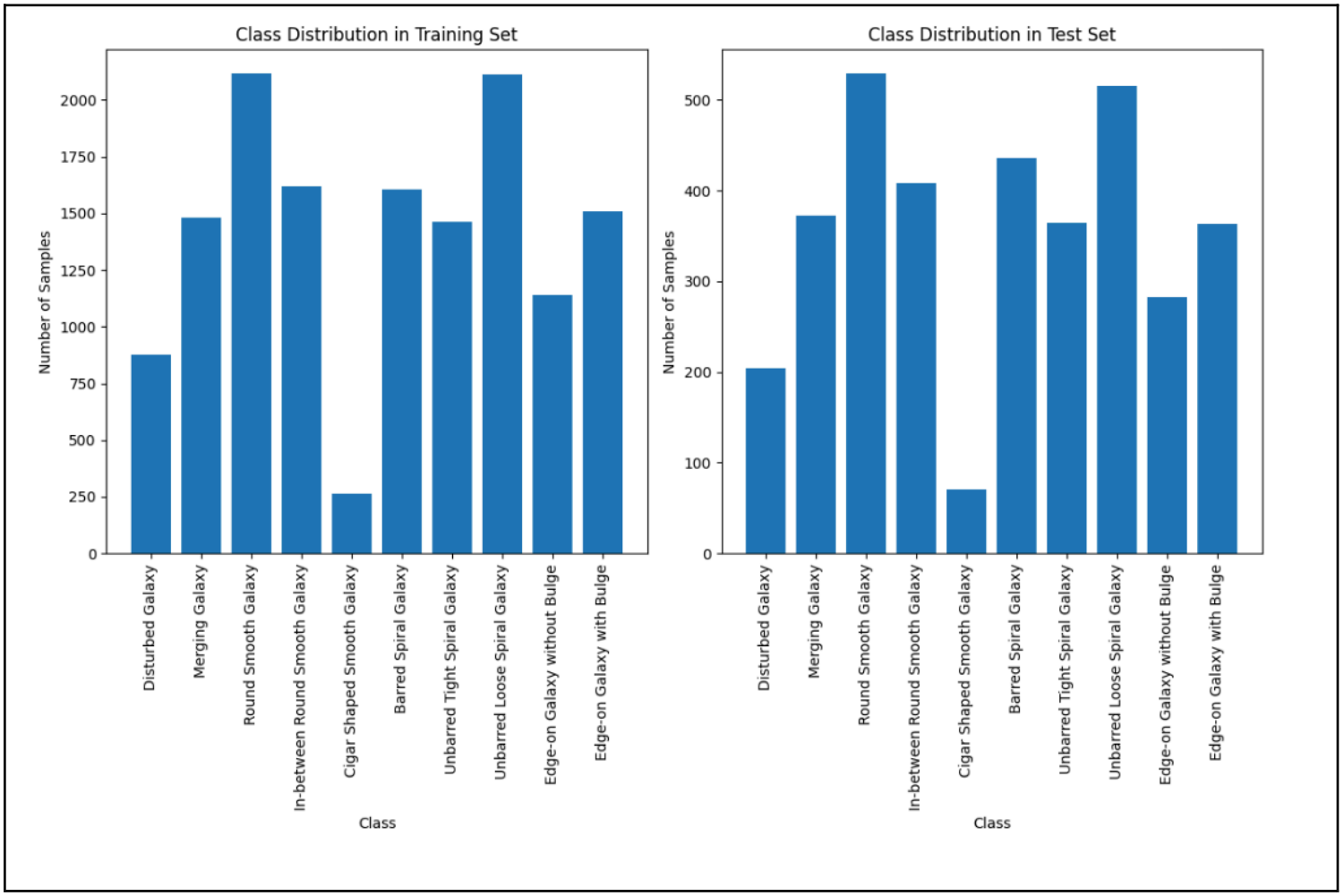


Figure 14: Class distribution for the Training and Test subsets.

Data Augmentation

Next to help mitigate any overfitting from the imbalanced dataset, I decided to create a few data augmentation steps to be applied to each model: first a random flip where each epoch every image has a 50% chance to be flipped horizontally, then a random rotation of +/- 20% of a full rotation, and finally a random translation of the image up or down and left or right of up to 20% of its height and width respectively.

While these steps won't fully mitigate any issues from the imbalanced nature of the dataset they should help mitigate overfitting in general. This is achieved by artificially increasing the number of samples in the dataset so each epoch the images the model is 'learning' are slightly different, forcing the model to learn the patterns in the data not just the exact images in the dataset.

Each model's architecture is illustrated in Appendix A

Now that everything was set up I made three models to train on the Galaxy10 DECals dataset.

Model 1

For Model 1 I chose to reuse the architecture from the model used previously with the CIFAR-10 dataset but with the added data augmentation steps. I chose to reuse the model as it makes for an interesting comparison so I can see how its performance compares on the 10 galaxy images versus the 10 CIFAR-10 classes.

It trained for 50 epochs without stopping early, taking 67 minutes to train, and achieving a final testing accuracy of 60%. As seen in the training accuracy and loss graphs, the training accuracy performance increased steadily throughout training and even after 50 epochs it only just looks to be levelling off. The validation performance is a little strange, it is always above the training performance and occasionally drops back down matching the training accuracy. I'm not sure what caused this behaviour however, so it might make for a future project to investigate further.

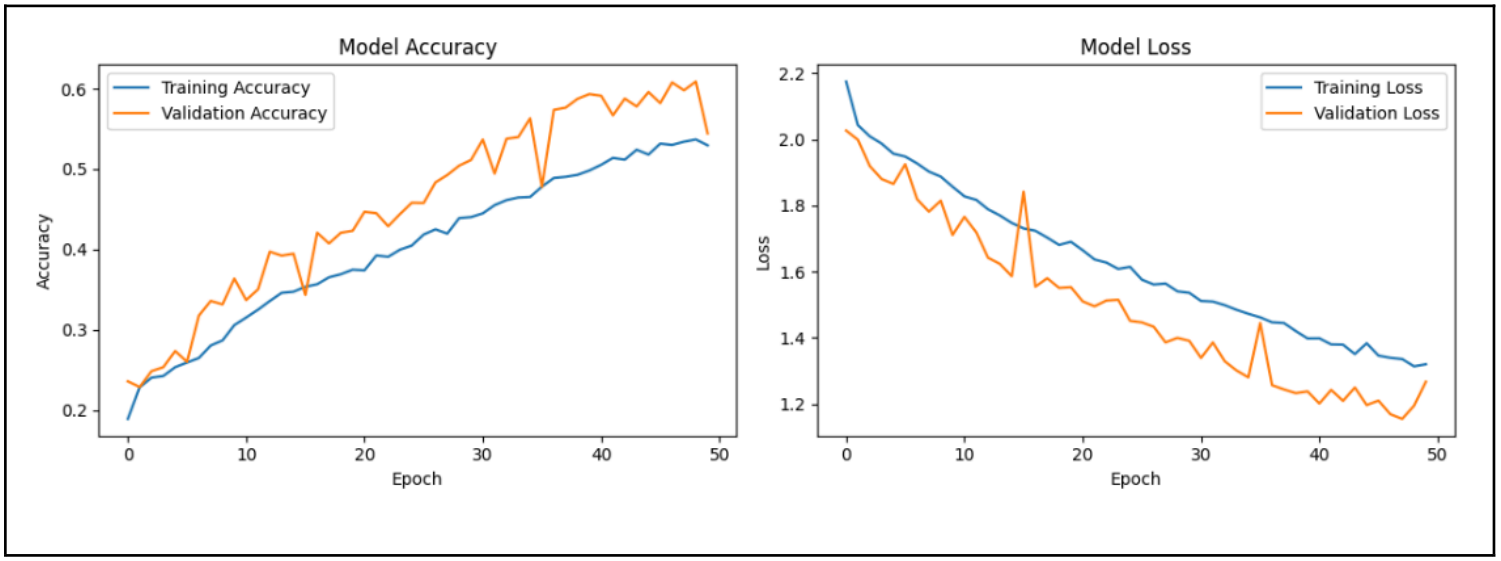


Figure 15: Model 1 Training Accuracy and Loss Graph

From the confusion matrix it is clear that this model struggles with some of the classes. It never predicted the Disturbed Galaxy and only predicted the Cigar Shaped smooth galaxy twice in the test set. It also struggled with the Barred Spiral Galaxies, instead predicting nearly half of them as Unbarred Loose Spiral Galaxies, and actually seems to overpredict Unbarred Loose Spiral Galaxies in a lot of cases. This seems to indicate that this model struggles on both the minority classes and with a couple of classes that are quite visually similar to each other.

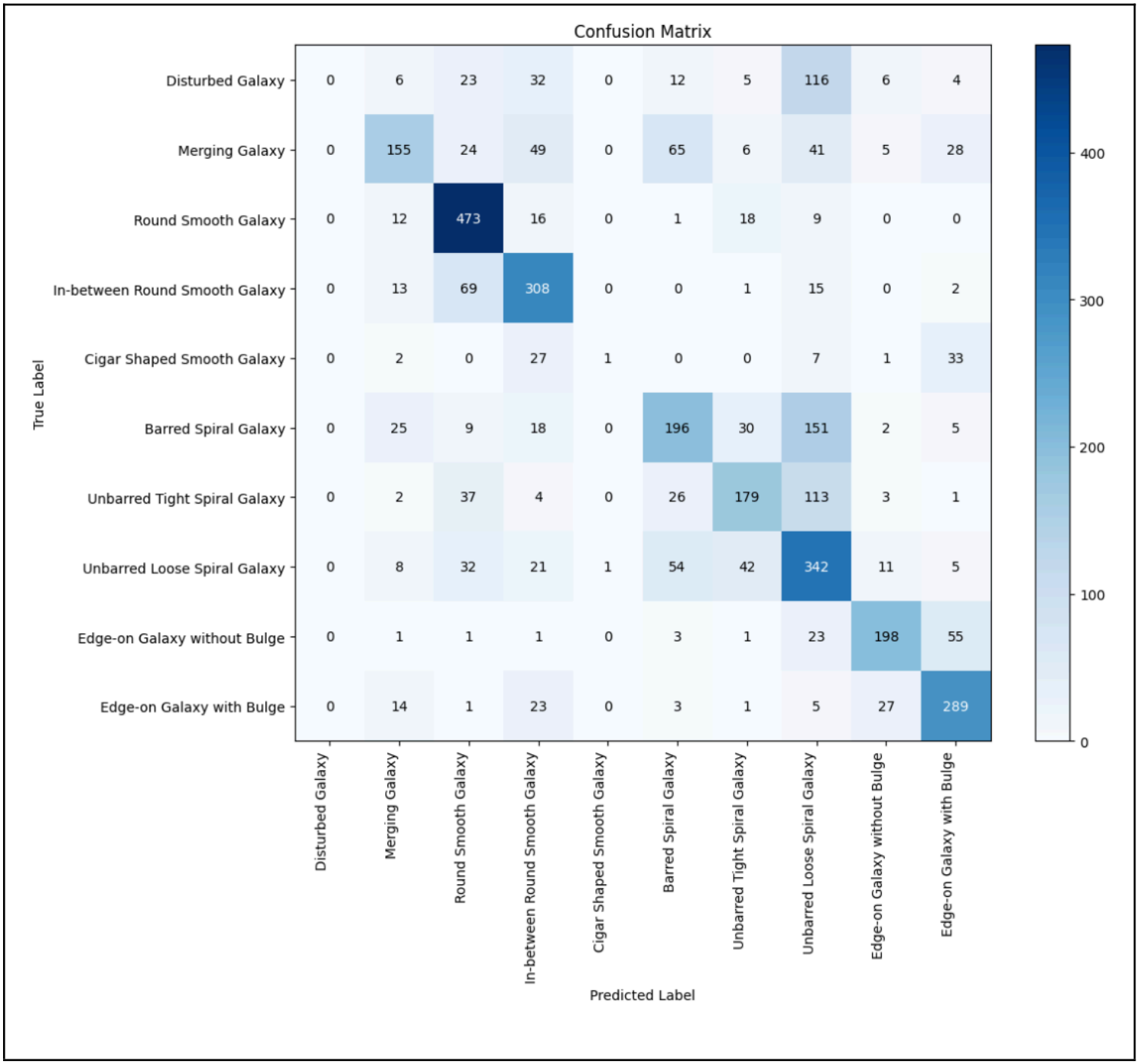


Figure 16: Model 1 Confusion Matrix

Model 2

For my second model I chose to build a new model based on one I developed previously for my TM358 machine learning and artificial intelligence coursework. The model was originally designed for categorising images of trees as seen from above. I chose to use this model as a base, as the Tree Image dataset it was designed for has some similarities to the galaxy dataset. Firstly both have 10 classes with a large class imbalance with each having distinct minority classes. Secondly the images in both datasets are similar in size. And thirdly there's a slight visual similarity between the data sets as they are mostly mostly one colour images of chaotic greens versus black starfields, with most images not having clear lines or edges.

This model is deeper, with 7 convolutional layers compared to 3 in the first model, uses a variety of kernel sizes which can help in capturing features at different scales, and the number of filters increases gradually.

Model 2 trained for 50 epochs without stopping early, taking 35 minutes to train, and achieving a final testing accuracy of 66%. As seen in the training accuracy and loss graphs, the model's performance increased quickly for the first 10 epochs before slowing down and remaining steady, only looking to level off towards the last few epochs.

Interestingly this model was quicker to train than model 1 and got better results, despite both having a similar number of parameters (3,649,546 vs 3,532,314). This increase in efficiency is likely due to the changes in architecture. Model 1 starts with a 3x3 convolution and 32 filters, while Model 2 starts with a 10x10 convolution and only 8 filters. This larger initial kernel in model 2 should capture relevant features more quickly, reducing the need for extensive processing in later layers. Model 2 also slowly increases in complexity starting with fewer filters and gradually increasing them, while model 1 jumps quickly to 64 filters. Model 2 uses a variety of kernel sizes, which might capture relevant features more efficiently than the consistent 3x3 kernels in model 1.

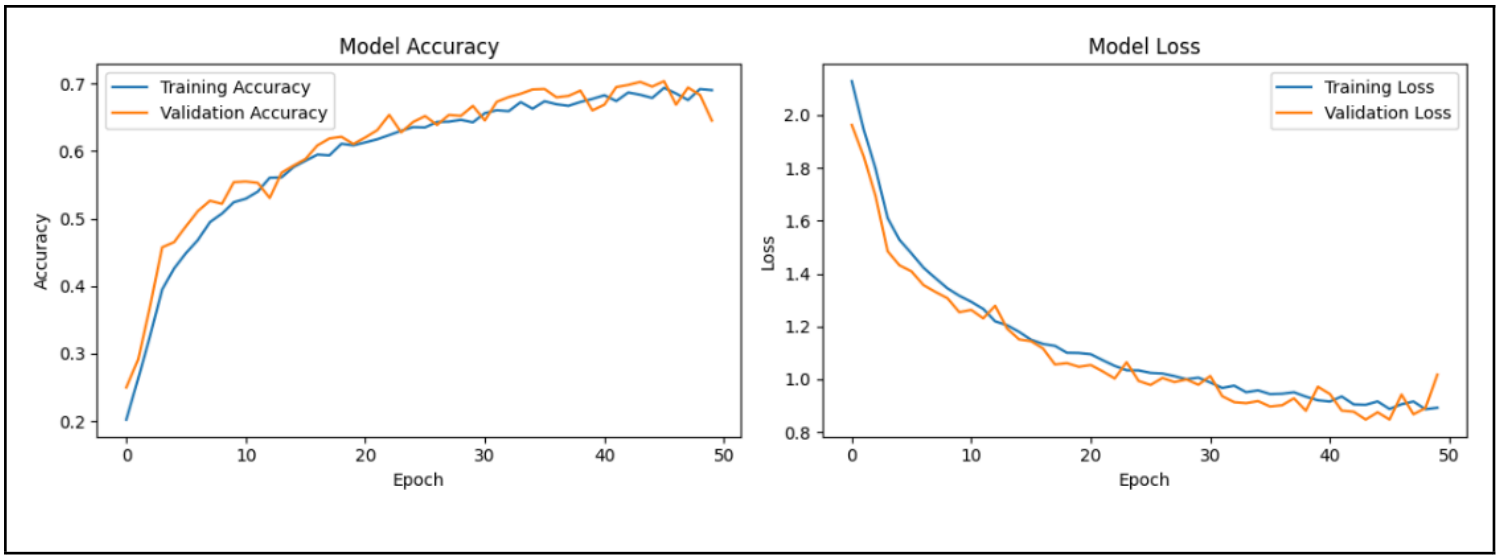


Figure 17: Model 2 Training Accuracy and Loss Graph

This model's confusion matrix shows slight improvements for some classes, particularly Cigar Shaped smooth galaxy now shows an improved performance. However there are some issues, the three spiral galaxy classes are often confused for one another, and merging and unbarred loose spiral galaxies are being over-predicted. This over prediction shows a bias towards these majority classes. Round Smooth Galaxies are still very accurately predicted

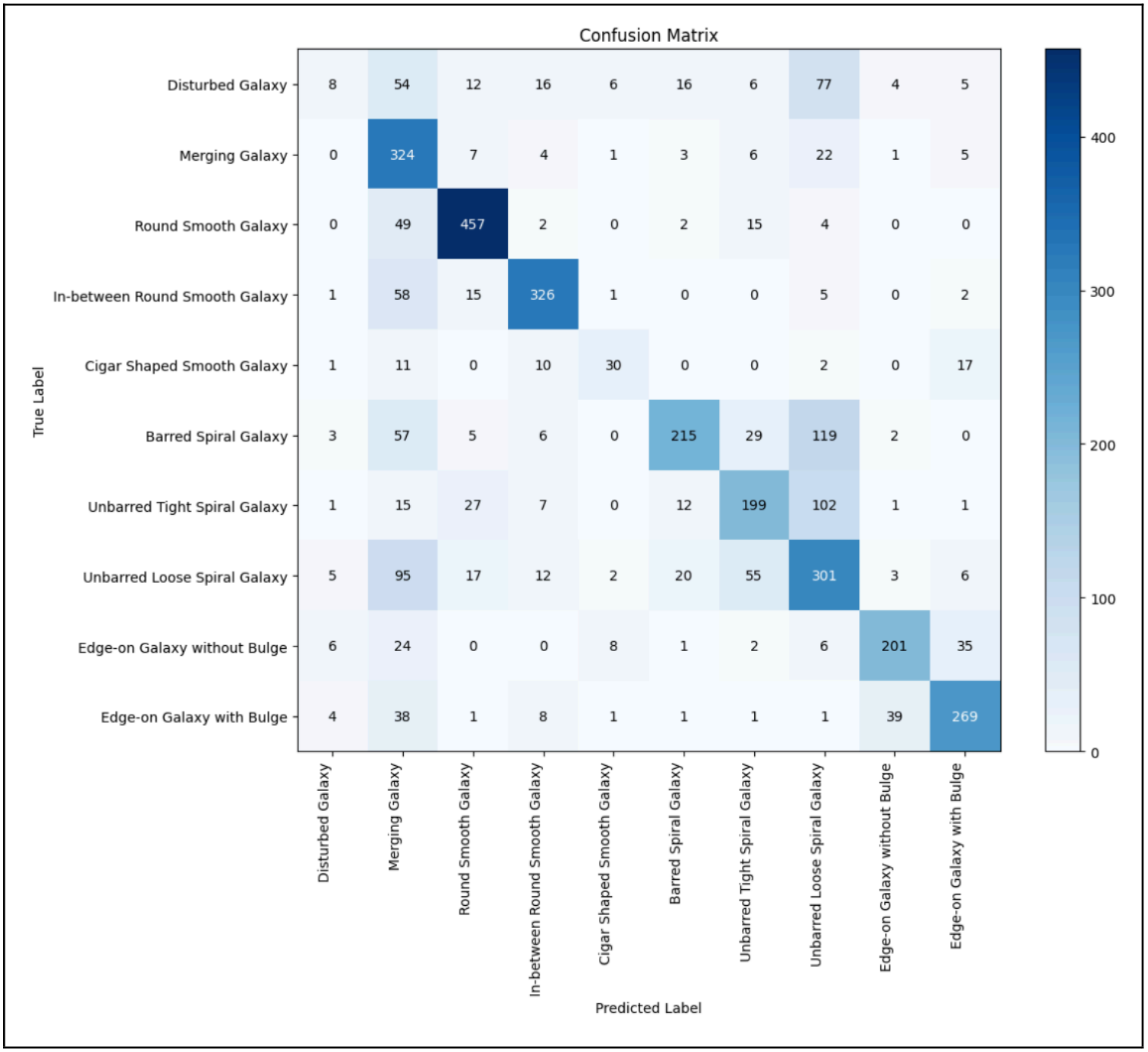


Figure 18: Model 2 Confusion Matrix

Model 3 & DenseNet

For Model 3 I wanted to try something new. Hui, W. et al. (2022) had used an architecture called DenseNet and gotten good results. I thought it would be a good final test to see if I could do the same.

Huang, G. et al. presented their DenseNet architecture in 2017 and it represents a novel approach that challenges the traditional linear architecture of CNNs. In DenseNet, each layer is directly connected to every other layer that follows it within a dense block, instead of information passing from one layer to the next in a linear fashion. In this way each layer not only learns from the layer immediately before it but also from every preceding layer in its block. This dense connectivity serves a dual purpose. First, it encourages feature reuse throughout the network, allowing later layers to build upon the foundational patterns recognized by earlier ones. Second, it promotes a smoother flow of gradients

during training, effectively combating the notorious vanishing gradient problem experienced by overly deep networks.

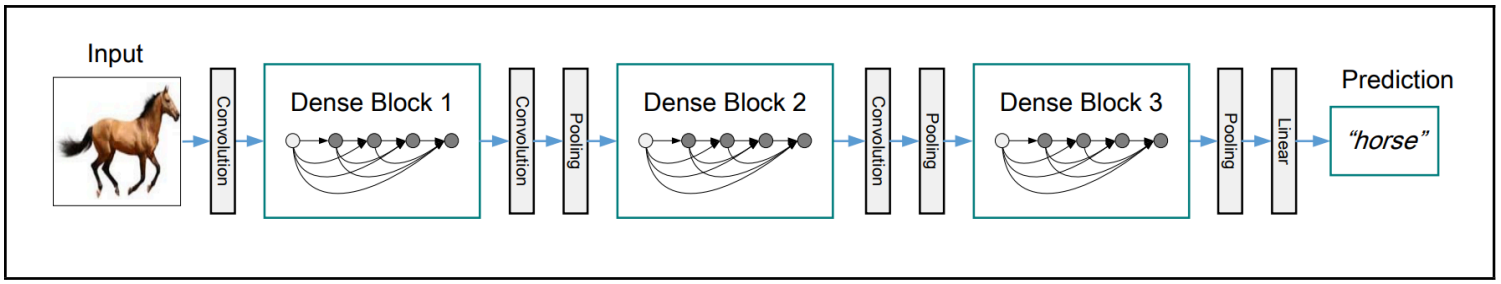


Figure 19: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling (Huang, G. et al. 2017)

Within TensorFlow there is a collection of deep learning models that are available as part of Keras Applications module. These models allow you to quickly use a prewritten architecture, then to add changes to it or build upon it further. There are a few versions of DenseNet available through Keras Applications each denoted by a number that states the number of layers. These models are available pretrained on large datasets like ImageNet but I chose to use the untrained version as the point of this exercise was to train the model on my machine to work on the Galaxy images. The previous largest model I had run on my pc had 10 layers, so I chose DenseNet121 for this exercise as it was the smallest one available. However, at 121 layers I was still unsure if this would run successfully.

To build the model 3, I added on the same three data augmentation steps used before, to the start of the DenseNet block and a couple of final dense layers afterwards to configure it for the 10 Galaxy classes.

The model trained for 38 epochs stopping early as its validation performance stopped improving, and achieving a final testing accuracy of 80%. Even though it stopped early this model still took a little over 9 and a half hours to train. Training accuracy performance increased steadily, but there is definitely something weird going on with the validation performance, with its performance rising and falling periodically before appearing to potentially level off. Repeated training of this model and changing up a few parameters might give some insight into what causes this phenomenon, but given the extreme training time that was not feasible for this project.

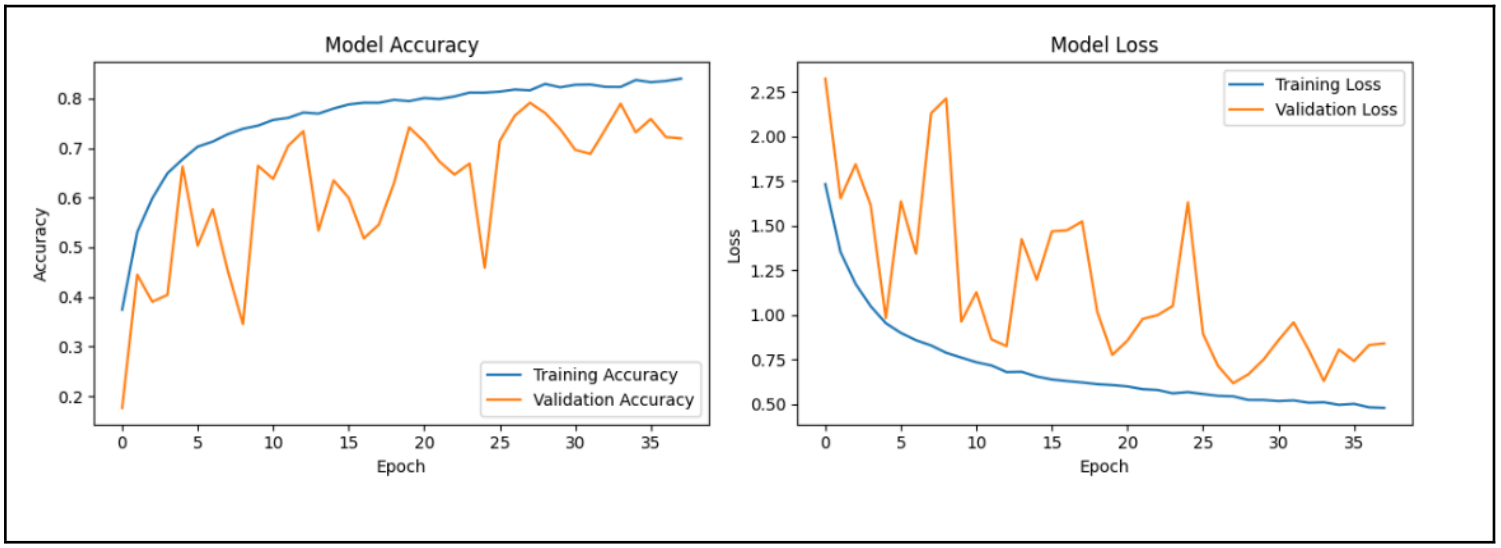


Figure 20: Model 3 Training Accuracy and Loss Graph

The confusion matrix for this model shows a definite improvement over the previous two models. It still struggles with the disturbed galaxy class, though it does at least get 52 true positive instances of this class whereas the first models got 0 and 8. It still predicts the unbarred loose spiral galaxy but not as much as the previous models.

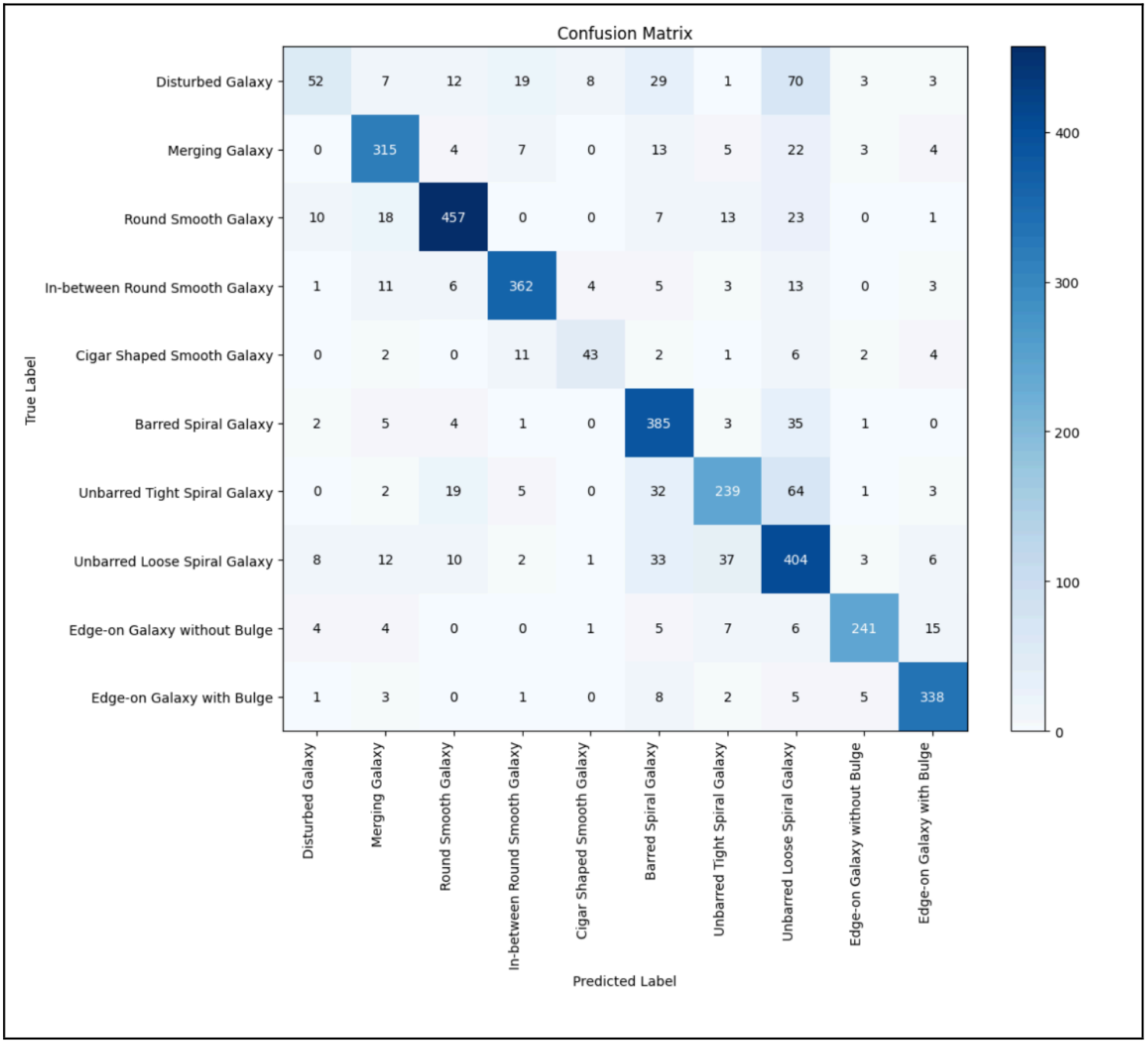


Figure 21: Model 3 Confusion Matrix

Results

Below are the results from all three models. I have recorded: the overall Average Accuracy for each model, each model's Precision, Recall and F1-score for each Galaxy Type, and both the Macro and Weighted Averages for each of Precision, Recall and F1-Score.

Each of the metrics describes a different measure of performance. Precision measures the accuracy of positive predictions for each galaxy type, indicating how often the model is correct when it identifies a specific class. Recall, on the other hand, indicates the model's ability to detect all instances of a particular galaxy type, showing how comprehensive its identification is for each class. The F1-score, being the mean of precision and recall, provides a balanced measure of the model's performance for each galaxy type, particularly useful given the uneven distribution of classes in the dataset.

To assess overall performance, I calculated both macro and weighted averages. The macro average, an unweighted mean across all galaxy types, offers insight into the model's performance treating all classes equally, irrespective of their frequency in the dataset. This metric is particularly valuable for understanding how the models handle rare galaxy types. The weighted average instead takes into account the frequency of each galaxy type, providing a performance measure that reflects the natural distribution of galaxies within the dataset. A significant discrepancy between these averages would indicate that the model performs disproportionately better on more common galaxy types.

Accuracy represents the overall correct classification rate across all galaxy types. It provides a quick understanding of each model's performance.

Together, these metrics offer a nuanced view of the models' capabilities. They identify specific strengths and weaknesses in classifying different galaxy types, the impact of class imbalance, and assess overall performance improvements across the three models.

The results of the test sets reveal a clear progression in performance from Model 1 to Model 3, with Model 3 demonstrating superior capabilities across all metrics and galaxy types. The overall accuracy of the models shows a steady improvement, increasing from 0.6 for Model 1 to 0.66 for Model 2, and reaching 0.8 for Model 3.

	Precision			**Recall**			**F1-Score**		
Galaxy Type	**Model 1**	**Model 2**	**Model 3**	**Model 1**	**Model 2**	**Model 3**	**Model 1**	**Model 2**	**Model 3**
Disturbed	0.00	0.28	0.67	0.00	0.04	0.25	0.00	0.07	0.37
Merging	0.65	0.45	0.83	0.42	0.87	0.84	0.51	0.59	0.84
Round Smooth	0.71	0.84	0.89	0.89	0.86	0.86	0.79	0.85	0.88
In-Between Round Smooth	0.62	0.83	0.89	0.75	0.80	0.89	0.68	0.82	0.89
Cigar Shaped	0.50	0.61	0.75	0.01	0.42	0.61	0.03	0.50	0.67
Barred Spiral	0.54	0.80	0.74	0.45	0.49	0.88	0.49	0.61	0.81
Unbarred Tight Spiral	0.63	0.64	0.77	0.49	0.55	0.65	0.55	0.59	0.71
Unbarred Loose Spiral	0.42	0.47	0.62	0.66	0.58	0.78	0.51	0.52	0.69
Edge-on Without Bulge	0.78	0.80	0.93	0.70	0.71	0.85	0.74	0.75	0.89
Edge-on with Bulge	0.68	0.79	0.90	0.80	0.74	0.93	0.74	0.77	0.91

	Precision			**Recall**			**F1-Score**		
	Model 1	**Model 2**	**Model 3**	**Model 1**	**Model 2**	**Model 3**	**Model 1**	**Model 2**	**Model 3**
Macro Average	0.55	0.65	0.80	0.52	0.61	0.76	0.50	0.61	0.77
Weighted Average	0.58	0.67	0.80	0.60	0.66	0.80	0.58	0.65	0.79

	Model 1	**Model 2**	**Model 3**
Accuracy	0.60	0.66	0.80
Training Time	67 minutes	35 Minutes	9.5 Hours

Notably, Model 3 exhibits significant improvements in classifying challenging galaxy types. For instance, while Model 1 completely failed to identify Disturbed Galaxies, Model 3 achieved an F1-score of 0.37 for this class. Similarly, the classification of Cigar Shaped Galaxies improved dramatically, with the F1-score rising from a mere 0.03 in Model 1 to a respectable 0.67 in Model 3.

All three models performed consistently well in classifying Round Smooth Galaxies and Edge-on Galaxies, with Model 3 achieving the highest scores for these types. This suggests that these galaxy types may have more distinct features that are easier for the models to identify.

A key strength of Model 3 is its ability to balance precision and recall across different galaxy classes. This balance indicates a more robust overall performance, with the model being able to both accurately identify galaxies of a specific type and capture a high proportion of galaxies belonging to each class.

The comparison of macro and weighted averages provides insight into how well the models handle class imbalance. The narrowing gap between these averages from Model 1 to Model 3 suggests that Model 3 is better equipped to deal with the varying frequencies of different galaxy types in the dataset.

Specific improvements are seen in several galaxy types. For example, the recall for Barred Spiral Galaxies increased dramatically from 0.45 in Model 1 to 0.88 in Model 3. The precision in classifying Unbarred Loose Spiral Galaxies also saw a significant boost, rising from 0.42 in Model 1 to 0.62 in Model 3.

Model 3 demonstrates more consistent performance across all galaxy types, with most F1-scores exceeding 0.7. This consistency is a crucial factor in the model's overall superior performance.

In Hui, W. et al. (2022)’s paper, they got a test accuracy of 0.8864 with their DenseNet121 model after training for 30 epochs. They however had extra preprocessing steps and optimised their architecture to better take account of the dataset itself.

The progression from Model 1 to Model 3 demonstrates clear improvements in overall accuracy, handling of challenging classes, and balanced performance across different galaxy types. The enhanced performance of Model 3, which utilises a DenseNet121 architecture, stands in contrast to the simpler CNN models used in Models 1 and 2, which were originally designed for non galaxy datasets. This significant improvement suggests that the more complex DenseNet121 architecture is better suited to capture the intricate features that differentiate galaxy classifications even when none of the datasets have been designed specifically for galaxy images. The ability of DenseNet121 to reuse features through dense connectivity likely contributes to its superior performance in this specific task of galaxy classification. When compared to Hui, W. et al.'s results I am happy that my results are approaching theirs’, with limited resources and without an expertise in the dataset allowing for further preprocessing, I was still able to get interesting results.

Part 6: Conclusion & Review of Project Work

In this project I set out to build a Convolutional Neural Network on my home pc and use it to classify images of galaxies. I wanted to show that increases in technology don't necessarily mean a removal of citizen science opportunities but can instead help people get involved and engaged in doing their own projects. In this I feel I have been successful.

I collected the necessary tools and software to be able to build and run CNNs learning about what software is available to assist and organise this process. I was able to find and use a dataset that was built thanks to previous citizen science efforts from the Zooniverse community; efforts that now are deemed inefficient as the sheer amount of data being generated by modern telescopes generates more images than could ever be categorised by humans. I was able to demonstrate that using the data that had previously been classified by hand could now be used to build a machine learning system to continue to classify new images into the future. And I was able to do this using consumer hardware and freely available software, demonstrating another way for individuals to engage with scientific projects.

If I were to continue this project I have a few ideas for future work. I would like to once again try to get the GPU setup working as that's something I had to move on from after a couple of failed attempts in order to avoid wasting further time. I'd like to do more tests with different CNN architectures to try and build a more optimised network for classifying galaxies including researching best practices in network design. Finally I would like to try collecting and building my own dataset to use as that is another area I have never explored.

Part 7: Review and Reflection

Review of project management

Going into this project I had no idea what I wanted to do, only that I wanted to do something using an astrophysics dataset as when I first started my university journey I studied physics and felt that finishing with something in that vein would act as a nice bookend to the whole experience. But coming up with an actual project idea I found to be extremely difficult as I was not really sure what was appropriate especially as I did not have much in the way of resources or experience beyond what I had done previously with the OU. During a Zoom call with my Tutor we came up with the idea of building a CNN classifier and detailing the process, as it was a topic I was enjoying in my other module. Otherwise, I have worked mostly independently, just checking in with my tutor when I needed some reassurance and guidance.

As my project initially focused on just building a CNN without any real world use case, I felt it needed some kind of framing device especially as it did not have any human participants. I came up with the idea of exploring the citizen science angle after realising the Galaxy image dataset used labels from a project I actually got the opportunity to help with during my school years. This allowed my project to touch on ethical and social issues

Initially I had a loose schedule month by month of project activities that was then refined into a Gantt chart for TMA02. However, ongoing illness meant that I was not always able to keep up with the more rigid schedule and the stress of constantly falling behind, led me to abandon the schedule and instead switch to a living document by TMA03. This document outlined what needed to be done at the start of each section of the report and I could look through it each day, see what needed to be done and focus on what I had the energy to do that day, slowly building up the project, jumping between sections rather than finish one bit then moving on. This document has over time evolved into this project report. I found this more free form approach removed some stress as even if sections were not finished I could still see the progress build up. Without such issues I may have been able to stick to a more formal plan, but being able to adapt and be flexible massively helped me get through this project successfully.

Review of personal development

What went well?

I managed to get the convolutional neural networks to run, and I enjoyed that I was able to find and use a dataset with pictures of galaxies, as they are a topic I'm interested in as a fan of all things space related. I got to learn how to set up my own machine learning environment and gain insights into the technologies and software used, a part of the process that was all done for me with the open universities cloud coding environment. I particularly enjoyed the coding of the networks, getting to use the skills I gained from previous modules, I enjoyed the process of setting them up, watching them run and analyse the results. And now I have the tools set up I can continue to use them beyond my studies applying them to other topics that interest me.

What didn’t go so well?

I initially planned to use datasets from Kaggle. But to save time after falling behind on my original plan, I decided to instead focus on datasets that I had prior experience with for the early stages of the project. This enabled me to get onto the galaxy stage more quickly, especially as that tied into both the citizen science and galaxy zoo sections of the project, and was a dataset I was more personally interested in.

Additionally I was disappointed that I did not manage to get my GPU to work with tensorflow. It would have been satisfying to be able to utilise the additional power that would have brought to the machine learning aspect of this project.

What would I do differently if I did the project again?

If I were to repeat the project I might not choose to focus on the citizen science angle or the setup process with Anaconda. Instead I would probably expand the coding element of the project, picking a single dataset and focus on working to optimise the network. This would give the project more of a singular focus instead of trying to combine several different ideas. But that's only now that I have the knowledge and tools to be able to set that up thanks to this project. I do however, quite like how I was able to combine a social issue aspect to a project where I learn new skills and apply knowledge I learnt during my time with the Open University.

Have I reached an acceptable standard and have I met the learning outcomes for this project?

The more freeform and self guided nature of the project has always intimidated me, I have much preferred the more structured nature of other modules working towards a predefined goal. Having to set my own goals has been a challenge to me as often I am unsure what would be acceptable or what level of detail is required.

However, on reflection I feel I have been able to meet the learning outcomes of this project module: I set myself the goal of learning how to set up and build a CNN to learn to classify images of galaxies, expanding on work within my area of interest in machine learning. I was able to connect this goal to wider social and ethical issues of applying new technologies to this task. I was able to gather and analyse literature sources relating to the project. And I was able to describe my experiences, thoughts, and analyses, in a clear structured format, connecting several ideas together into a single narrative.

References

Wang, P. *et al.* (2020) 'Comparative analysis of image classification algorithms based on traditional machine learning and deep learning', *Pattern Recognition Letters*, Vol.141, pp.61-67. Available at: https://doi.org/10.1016/j.patrec.2020.07.042

Lv, Q. *et al.* (2022) 'Deep Learning Model of Image Classification Using Machine Learning', *Advances in Multimedia*, vol. 2022, 12 pages. Available at: https://doi.org/10.1155/2022/3351256

Banerji, M. *et al*. (2010) 'Galaxy Zoo: reproducing galaxy morphologies via machine learning', *Monthly Notices of the Royal Astronomical Society*, vol. 406(1), pp. 342–353. Available at: https://doi.org/10.1111/j.1365-2966.2010.16713.x

Cheng, T-Y. *et al*. (2021) 'Galaxy morphological classification catalogue of the Dark Energy Survey Year 3 data with convolutional neural networks', *Monthly Notices of the Royal Astronomical Society*, vol. 507(3), pp. 4425–4444. Available at: https://doi.org/10.1093/mnras/stab2142

Hui, W. *et al*. (2022) 'Galaxy Morphology Classification with DenseNet', *Journal of Physics: Conference Series*, vol. 2402, 11 pages. Available at: https://doi.org/10.1088/1742-6596/2402/1/012009

Huang, G. *et al*. (2017) 'Densely Connected Convolutional Networks', *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* Honolulu, HI, USA, 21-26 July 2017 Available at: https://doi.org/10.1109/CVPR.2017.243

Hecker, S. Haklay, M. Bowser, A. Makuch, Z. Vogel, J. Bonn, A. (ed) (2018) Citizen Science: Innovation in Open Science, Society and Policy. London: UCL Press. Available at: https://doi.org/10.14324/111.9781787352339

Woodcock, J. *et al*. (2017) 'Crowdsourcing Citizen Science: Exploring the Tensions Between Paid Professionals and Users', *Journal of Peer Production,* (10). Available at: http://peerproduction.net/issues/issue-10-peer-production-and-work/peer-reviewed-papers/crowdsourcing-citizen-science-exploring-the-tensions-between-paid-professionals-and-users/

McGourty, C. (2007) 'Scientists seek galaxy hunt help', BBC News, 11 July. Available at: http://news.bbc.co.uk/1/hi/sci/tech/6289474.stm

Feilden, T. (2013) 'Citizen science is the new black', BBC News, 15 October. Available at: https://www.bbc.co.uk/news/science-environment-24532312

Gray, R. (2017) 'Galaxy Zoo: Citizen science trailblazer marks tenth birthday', BBC News, 11 July. Available at: https://www.bbc.co.uk/news/science-environment-40558759

Keel, W. *et al*. (2016) 'Fading AGN Candidates: AGN Histories and Outflow Signatures', *The Astrophysical Journal*, vol. 835 19 pages. Available at: https://doi.org/10.3847/1538-4357/835/2/256

Anderson, D.P. *et al.* (2002) 'SETI@home: an experiment in public-resource computing', *Communications of the ACM*, vol. 45(11), pp. 56-61. Available at: https://doi.org/10.1145/581571.581573

Zooniverse (2010) *Types of Galaxies*. Available at: https://blog.galaxyzoo.org/2010/05/12/types-of-galaxies/ (Accessed: 1 July 2024).

Zooniverse (2024) *Galaxy Zoo*. Available at: https://www.zooniverse.org/projects/zookeeper/galaxy-zoo/ (Accessed: 1 July 2024).

Datasets

Tensorflow datasets

This project utilised datasets from the TensorFlow Datasets collection.

TensorFlow Datasets. (2023) *A collection of ready-to-use datasets.* Available at: https://www.tensorflow.org/datasets

LeCun, Y., Cortes, C. and Burges, C. (2010). MNIST handwritten digit database. Available at: http://yann.lecun.com/exdb/mnist/

Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images.

AstroNN

This project utilised a dataset from AstroNN

AstroNN (2023) Welcome to astroNN's documentation!. Available at: https://astronn.readthedocs.io

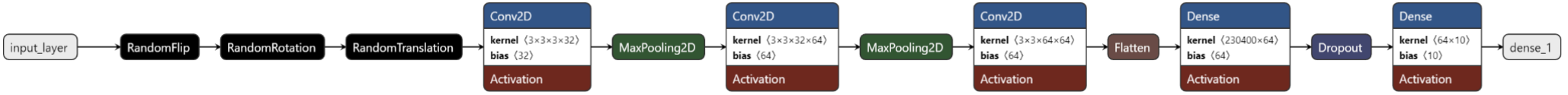
Galaxy10 dataset classification labels come from Galaxy Zoo https://www.zooniverse.org/projects/zookeeper/galaxy-zoo/

Galaxy10 dataset images come from DESI Legacy Imaging Surveys https://www.legacysurvey.org/

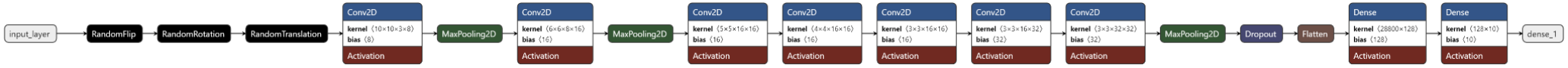
Appendix A: Model Diagrams

In order to better visualise the CNN model architecture for the three models used for the Galaxy10 DECals dataset phase of this project, I used Netron to create pictures showing their architecture. Netron is a free online tool that creates visualisations of deep learning and machine learning models, including CNNs like those used in this project. DenseNet121 shows up as a single element of the visualisation showing how it connects to the rest of the model architecture.

Model 1



Model 2



Model 3 (DenseNet)