PHY480 Semester II Project Report

Developing a Machine Learning Algorithm to Classify Detections in Large Astronomical Surveys

Matthew Kent Myers*

June 6, 2019

Abstract

With the ever increasing numbers of astronomical sources being observed due to advances in wide survey astronomy, it is becoming ever more apparent that reliable automated solutions to morphological source classification are required. We present a machine learning approach, in the form of a convolutional neural network (CNN), capable of using purely morphological properties to distinguish between galaxies and 'other' sources (stars, artefacts and pixel defects) with >98% accuracy within low resolution (1.4"/pixel) single band images taken with the Gravitational-wave Optical Transient Observer (GOTO). By matching detected sources to the Galaxy Zoo 2 catalogue and augmenting images of galaxies, a training sample of 120,000 pre-classified images was created and used to train the CNN. By only classifying sources as galaxies when the CNN was >95% sure the source was a galaxy and assuming all other sources were 'other', a CNN classifier was able to achieve a precision of 59%, recall of 90% and F1 Score = 0.713, when classifying data proportional to the unbalanced nature of that in the night sky. Furthermore, the model may be potentially performing better than calculated due to incorrectly labelled images, lowering precision calculations. While the precision is still fairly low the model can be used to find galaxies within unseen GOTO survey images, producing a reduced (∼4% of total) list of sources where >59% of these sources are galaxies. Finally, the same CNN architecture was retrained to reproduce Galaxy Zoo 2 classifications of featured and smooth galaxies, achieving > 80% accuracy, providing a proof of concept that galaxy subclassification can be achieved with low resolution data.

1 Introduction

The classification of astronomical objects, allowing for large catalogues of positions, numbers and properties of different types of sources is a vital tool for astronomers in understanding and studying the Universe in which we live. Morphology, crucial in understanding the structure and dynamics of astronomical sources, is one of the most fundamental metrics used by astronomers when classifying objects. Due to improvements in telescope technology as well as increasing numbers of observations, vast numbers of astronomical sources are being readily imaged. It is becoming ever more difficult to classify all these astronomical objects based upon their morphologies and it is no longer possible for professional astronomers to classify all sources manually, as was done in the past.

One approach to tackling the ever more increasing number of sources has been crowd sourced classification, most notably Galaxy Zoo [Lintott et al, 2008], allowing volunteers to classify images of galaxies via their morphology. Whilst Galaxy Zoo has been extremely successful in its aim of classifying sources in the night sky, it is still not capable of classifying the growing

*Email: Matthewkentmyers@gmail.com

number of imaged sources. Even with very large numbers of volunteers, it has still taken many months/years to achieve classifications of 900,000 galaxies within the Sloan Digital Sky Survey(SDSS) [York et al, 2000], due to the need for human classification. Furthermore, small teams of astronomers do not have the resources to conduct such large scale citizen classification projects. Another approach that removes the need for human classification altogether, are machine learning classifiers. Over the last few decades, significant advancements within machine learning have made it a more accessible, realistic and scalable approach to complicated computer-related tasks that are often too difficult to solve via hard coded algorithms. One of the main uses of machine learning has been the classification of images and has readily been applied to images of astronomical sources, successfully being used to discriminate between stars and galaxies as well as replicate morphological classifications made by Galaxy Zoo.

The focus of this project is to use machine learning to classify astronomical sources detected within images taken by the Gravitational-wave Optical Transient Observer (GOTO), into galaxies and stars purely based upon their morphological properties. GOTO is a set of 4 robotic telescopes located on La Palma, with the intended purpose of making follow up observations to gravitational wave observations. As such GOTO has a large field of view and is capable of observing 20^{2o} of the sky at once. However, it has a relatively low resolution, with each pixel equating to $\sim$1.4". This lower resolution presents many problems when attempting to distinguish between stars and galaxies based upon their morphological properties, as both types of sources often have similar shapes and sizes with only slightly different light profiles within GOTO images.

The following section (2) of this report contains a comprehensive overview of machine learning approaches to classification tasks, culminating in section 2.5, which briefly highlights some previous machine learning applications to classification tasks within astronomy. The remainder of the report is outlined as follows. In section 3 we describe the steps taken to convert large survey images into a set of labelled training images as well as how these images were manipulated from image form into one that can be used to train (or be classified by) a CNN. In section 4 we describe how the CNN was setup, with the architecture and training process outlined. In section 5 we outline the results of the CNN classifier as well as discuss the significant problem of unevenly sized test samples (such as sources in unseen GOTO images) and outline how the CNN can be improved. In section 6 we demonstrate a proof of concept CNN capable of making subclassifications of galaxies imaged within GOTO survey images. Finally, section 7 concludes and summarises the findings within the report.

2 Machine Learning

The following subsections form a comprehensive overview of machine learning approaches to classification tasks and outline the decision process that was made when choosing the best machine learning approach for the project. Section 2.1 describes the main types of machine learning, as well as their advantages and disadvantages with regard to the project. Section 2.2 outlines the principles of an artificial neural network, with section 2.3 outlining the more complex convolutional neural network. Section 2.4 discusses the training procedure that allows convolutional neural networks to classify images. Finally, section 2.5 reviews previous uses of machine learning within astronomy.

2.1 Best Machine Learning Approach to Classifying Galactic Images

There are many different machine learning approaches to classifying images. Appendix B - section 1.2, outlines further literature regarding the various types of machine learning - as well as their advantages and disadvantages when considering an image classification task.

Machine learning can broadly be classified into two groups: supervised and unsupervised learning. Supervised machine learning relies on a set of pre-classified images, called the training

data, from which the classifier 'learns' (Appendix B - section 1.2.1). Conversely, unsupervised machine learning classifiers do not require pre-classified training samples in order to learn from, but instead group data into groups with similar properties. For the purpose of this project, when deciding the optimal approach to classify images taken within a wide area survey, it was quickly decided that, due to time constraints, difficulties in validating training, as well as the inability to pick desired classification groups, that an unsupervised machine learning approach would not be pursued. This left various feature and non feature-based machine learning approaches. Within feature-based learning, the features that accurately describe the differences between image classes can be manually extracted from each image (Appendix B - section 1.2.2). These features are then used by the machine learning algorithm to classify sources, learning from training examples about how important each feature is and how it relates to each class of image. Feature-based machine learning approaches are often simpler and just as accurate as more complex machine learning approaches. However, when picking out features from an image, much of the image detail is lost, as the pixel data is reduced down to only a few parameters. Due to the low resolution of GOTO images already making it difficult to pick out features that distinguish stars from galaxies, we need to preserve as much of the image data as possible. It was therefore decided a feature-based approach was not appropriate.

With unsupervised and feature-based machine learning not being appropriate for the classification of images taken within GOTO images, the remaining approach to classification tasks is the artificial neural network - and its more advanced continuation - the convolutional neural network.

2.2 Artificial Neural Networks

First theorised in 1943, artificial neural networks (hereafter, ANNs) are built on the principle of neurons found in nature [Gershenson, 2003]. Each artificial neuron takes in several inputs, each with its own weight. The neuron then sums up these inputs and adds an additional bias input, describing how important that neuron is within the model. Finally, a chosen activation function is applied to decide if the neuron will 'fire' or not. This is shown schematically in figure 1 (credit to J. B. Ahire[1]) and mathematically in equation 1, where y is the output of the neuron, ϕ is the activation function, ω_i is the weight of input x_i and b is the bias of the neuron.

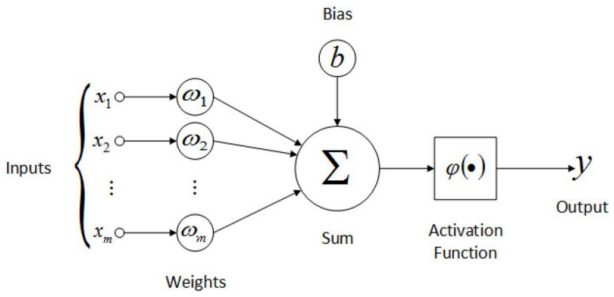


Figure 1: A schematic diagram of an artificial neuron, comprising one neuron within an artificial neural network.

$$y = \phi\Big(\sum_i \omega_i x_i + b\Big) \tag{1}$$

An ANN is comprised of many artificial neurons, ordered in layers, shown schematically in figure 2 (credit to P. Kasture[2]). The first layer within the network is the input layer. Depending on the input given to the ANN, it can be either a non feature or feature-based form of machine

[1] https://medium.com/@jayeshbahire/the-artificial-neural-networks-handbook-part-4-d2087d1f583e
[2] *http://blog.priyankakasture.me/neural-network-fundamentals/*

learning. Each input neuron of a feature-based ANN will contain a value corresponding to a feature that has been manually extracted. The ANN will then use these features to classify the image. However, more commonly and more relevant to the project is a non feature-based ANN, where each input neuron contains a normalised pixel value from the image, with there being as many input neurons as pixels in an image. Each neuron within the intermediate hidden layers takes an input from all neurons in the previous layer and applies the procedure described in figure 1 and equation 1. The output from the neuron is then passed to every neuron in the next layer and the procedure is repeated. The final layer of an ANN is called the output layer. The output layer typically has as many outputs as classifications, with each neuron corresponding to a class. The neuron with the highest value in the output layer corresponds to the class which the ANN has predicted the image belongs to.

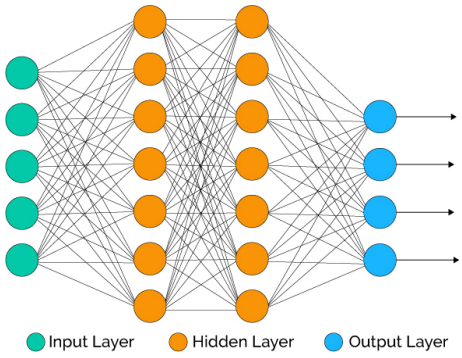


Figure 2: A schematic diagram of an artificial neural network comprising of an input layer, two hidden layers and an output layer. If there are several hidden layers a neural network is considered a deep neural network.

A continuation of the ANN is the convolutional neural network. Based upon the same model as an ANN, the convolutional neural network has rapidly became the dominant force within machine learning image classification, due to its unparalleled accuracy. Since we exploit convolutional neural networks in this project, we explore them in detail within the next section.

2.3 Convolutional Neural Networks and Deep Learning

Based upon the same principle as an ANN, a convolutional neural network (hereafter, CNN) is still comprised of layers of neurons, however, within a CNN each neuron can play a very different role. Each layer within a CNN is comprised of many neurons of the same type. The main different types of neurons (and therefore layers) are outlined below:

- **Input Layer:** Within a CNN the input is no longer a row of neurons, but one neuron which is a 2D 'image' of normalised pixel values (there can be 3 input neurons if colour images are being used - one for red, green and blue).
- **Convolutional Layer:** The convolutional layer(s) (giving the CNN its name) is the layer(s) that allows the CNN to pick out features from an image.

  Within a convolutional layer each neuron, instead of being a single value, is now an n×n matrix (called a filter or kernel) which is specifically designed to pick-out a certain feature from the image when scanned across the pixel values of the input neuron (the 2D pixel values). The kernel scans across the image, taking the dot product of the pixel values and the kernel, producing a value which is added to a new 'distorted' image with a certain feature highlighted (e.g edges, or circles) called a feature map. There are as many feature maps as neurons in a convolutional layer, each picking out a different feature as different kernels are used. Figure 3 (credit to E. Bonura[3]) shows the kernel taking the dot products of pixel values from the input layer of the CNN and adding them to a feature map.

[3]https://medium.com/@bonura.emanuele.sv/a-neural-network-in-11-lines-of-vba-4367a7219441

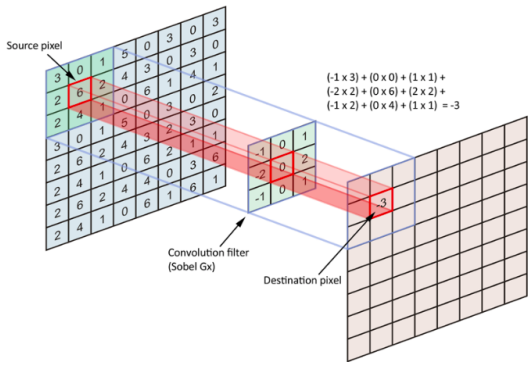


Figure 3: A schematic diagram of a convolutional layer within a CNN. The filter is run across the image, with the dot product being added to a feature map, which can be used within the next layer of the CNN.

- **Dense Layer:** Within a CNN, a regular layer of neurons, as described in section 2.2 is referred to as a dense layer. As each neuron is connected to all neurons in the previous layer, with each connection and neuron having a weight and bias, it is referred to as dense due to a large number of trainable parameters.

- **Activation Layer:** In contrast to an ANN, after each layer there is often an activation layer rather than a simple activation function within each neuron. Each activation neuron within an activation layer can apply a different activation across each feature map, or just one activation function can be applied to each feature map.

  One of the most commonly utilised activation function within CNNs (and routinely used within the CNN outlined within section 4) is the rectifier function. The rectifier function, often referred to as a ramp function, is described by equation 2 where x is the input to the activation neuron and y is the output. The rectifier function sets all negative neurons in the previous layer to zero and leaves positive neurons untouched.

$$y = \left\{ \frac{0 \; for \; x < 0}{x \; for \; x \geq 0} \right\} \tag{2}$$

- **Pooling Layer:** A pooling layer is typically applied after a convolutional layer and is used to reduce the dimensionality of a feature map, speeding up training time as well as improving accuracy. The most common type of pooling layer is max pooling, a relatively simple operation that takes the maximum value within defined windows applied across the whole image.

- **Dropout Layer:** A problem often experienced within supervised machine learning is overfitting, whereby the model learns how to differentiate between images specific to the training set, often at the detriment of finding general features. This results in a model with good training accuracy, however is incapable of classifying unseen images successfully. A dropout layer is used to prevent overfitting within a CNN. During every iteration of training, several neurons (chosen randomly) within a dropout layer are set to zero, effectively making them unusable by the CNN when classifying the training images. This process forces the CNN to classify images using the remaining neurons, preventing the CNN from becoming too reliant on certain neurons and forcing it to learn more robust features.

- **Flattening Layer:** A flattening layer simply flattens the data from a 2 dimensional image down to a 1 dimensional list of pixel values. These values can then be run through a regular ANN, or if the dimensionality has decreased enough, can simply be used to make classifications.

CNNs with many hidden layers and therefore the capability of picking out many features from images are called deep CNNs, or deep learning. Deep learning has proven to be able to pick out many abstract features and as such is the most accurate and complex way of classifying images using machine learning, outclassing all other forms of machine learning classifiers.

In the same fashion as an ANN, each connection between neurons in a CNN has its own weight and each neuron has its own bias. It is these weights and biases that are iteratively changed during training that allow a CNN to 'learn' what features are important with regard to distinguishing different image classes.

2.4 Training a Convolutional Neural Network

CNNs are a supervised machine learning approach to image classification; as such in order for it to accurately classify unseen images we must provide the CNN with pre-classified images for it to 'train' on. Each neuron and connection in a CNN has an associated bias and weight. By comparing the CNN's predictions to a set of pre-classified images, called the training sample, the weights and biases are slightly modified by an iterative process. Each neuron in a CNN can pick out different features and if a feature is good at describing the difference between two classes of images the bias to that neuron will be increased via training as it improves the accuracy. Furthermore, the following connection to subsequent neurons will become weighted more heavily. Over the course of many training iterations, all weights and biases will be slowly changed to achieve the best accuracy.

Due to how a CNN is trained it is prone to overfitting as mentioned earlier. Overfitting is when the machine learning classifier becomes very good at classifying the training image, however can no longer classify unseen data (i.e., it has not learnt the general properties that describe the classes, but has just learnt to classify the training images). To help avoid overfitting, another group of pre-classified images, that are not used to train the CNN, called the validation set, are used after each iteration of training to test the accuracy of the CNN. Figure 4 (credit to *elitedatascience.com*[4]) shows how the error is expected to change with training. As can be seen the error on the training set continuously decreases, however after the early stopping point the error of the validation set starts to increase as the CNN has started to overfit. The training of the CNN must be stopped before overfitting occurs to ensure maximum accuracy.

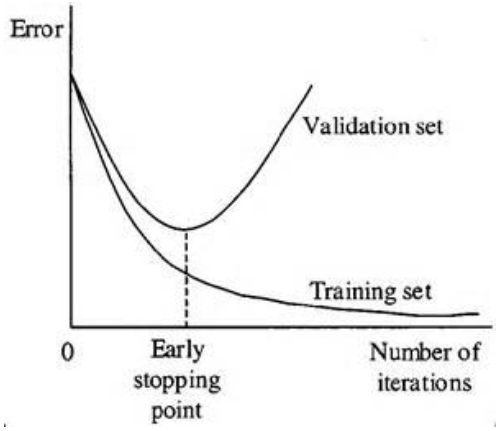


Figure 4: A graph showing how the error of the training and validation set of images when classified with a CNN is expected to change with the number of iterations of training

2.5 Machine Learning within Astronomy

With the exponential increase in computational power as well as advances in machine learning algorithms, machine learning is proving to be a very active and promising area of research

[4] *https://elitedatascience.com/overfitting-in-machine-learning*

within classification tasks, with astronomy being no exception. There is a long history of many machine learning approaches being used to classify images of astronomical objects, utilising many approaches. Whilst there are many examples of unsupervised (e.g. [Hocking et al, 2017] & [Schutter et al, 2015]) and feature-based approaches (e.g. [Calleja, 2004], [Gauthier et al, 2016] & [Huertas-Company et al, 2010]) being used to successfully classify images of astronomical objects (Appendix B - section 1.3.1 & 1.3.2 respectively), ANNs and CNNs have consistently provided the best results, outperforming any shallow machine learning approaches [LeCun et al, 2015] as well as all major feature-based approaches [Bazell & Aha, 2001].

Since 1992 ANNs have been able to achieve high accuracy when classifying images of galaxies (Appendix B - section 1.3.3), with Storrie-Lombardi et al [1992] achieving 65% accuracy classifying galaxies into 5 groups, increasing to 90% accuracy if considering the second best classification. Banerji et al [2010] achieved 90% accuracy classifying 1,000,000 images from the SDSS into point sources, early and late type galaxies, using an ANN trained on 75,000 sources. As outlined in section 2.3, CNNs have unmatched accuracy with regard to image classification, being able to tell the difference between high quality images of stars and galaxies with >99% accuracy [Kim & Brunner, 2016]. Dieleman et al [2015] were able to replicate Galaxy Zoo classifications for the 65,000 brightest objects within Galaxy Zoo with ∼99% accuracy for objects with high consensus within Galaxy Zoo. Whilst these high accuracies are for good quality images, high accuracies are still achieved using lower quality images (e.g Abraham et al [2018] & Aniyan & Thorat [2017]) who classified single band radio images of galaxies into 3 classes achieving up to 95% accuracy and developed a CNN capable of detecting bar structures within 9,346 images within the SDSS with 94% accuracy respectively.

Whilst there is no definite optimal way to approach a machine learning classification task, the consistently highest accuracy classifiers are CNNs. With the development of Python packages such as Tensorflow and Keras, there is no longer a complicated barrier to setting up and running a CNN (and other forms of deep learning), making full utilisation of CNNs (and deep learning) easier and more accessible than ever before. As such, a CNN was decided as the best machine learning approach for the classification of sources with GOTO survey images.

3 Data Collection and Processing

In this section we outline the steps taken to process the data from a large survey image taken with GOTO into a form classifiable by a CNN. The following section is split into the following parts: Section 3.1 describes the raw image data. Section 3.2 outlines the process of creating a training set of pre-classified cutout images of sources taken with GOTO, which are used to train and test the CNN. Finally, section 3.3 describes how the cutout images of sources were manipulated into a form capable of training (or being classified by) the CNN.

3.1 Data

The raw image data is delivered in the form of large single band field of view telescope observations, imaged with the GOTOs 4 robotic telescopes[5], located at Roque de Los Muchachos observatory on La Palma. Each image is $\sim 20^{2o}$ in size, with a resolution of 1.4"/pixel and contains thousands of sources. Different regions within GOTO images have differing levels of background noise. To mitigate this problem, efforts were taken to remove a 2D map of background noise as opposed to just removing a constant background level from the image during the source detection stage of image pre-processing (section 3.2.1). This resulted in a background subtracted image with a much more constant level of background noise.

[5] *https://goto-observatory.org/*

3.2 Creating the Training Set of Images

As a CNN is a supervised machine learning approach, it requires a set of pre-classified images - called the training set - in order to learn. This subsection covers the process of creating this training set of images. Initially a source detection algorithm was run across each GOTO image (section 3.2.1) to find all sources present; the detected sources were then compared to the Galaxy Zoo 2 catalogue in order to find a representative sample of galaxies within GOTO images (section 3.2.2). Finally the sources were cut out into individual postage stamp images (section 3.2.3) creating the training sets classified images.

3.2.1 Source Detection

The first step to generating the training set of images is running an accurate source detection algorithm across each image taken with GOTO. After much experimenting (Appendix B - section 2.2) `detect_sources`[6] by Photutils was found to be the most accurate and versatile source detection tool. Any group of >10 pixels brighter than 1.5× the background standard deviation at that location were detected as a source. Many images taken with GOTO have artefacts running across them, such as the one shown in figure 5. In order to try and stop these artefacts from being detected as sources we implemented a 2D Gaussian kernel[7] to the detection procedure. This kernel sweeps across the image whilst detecting, making the detection algorithm more likely to detect sources with shapes similar to that of the 2D Gaussian chosen. The implementation of the kernel not only reduced the number of artefacts being detected as sources (as they often do not have shapes like a 2D Gaussian), but also reduced the amount of double detections of larger/strange shaped sources, as it effectively smoothed them out. It should be noted that the largest sources in an image are still double detected and many artefacts still get detected as sources, which can routinely add 100+ detected sources to an image, as shown in figure 5 where each red circle is a detected source. It proved very difficult to accurately remove these detected artefacts without unintentionally removing actual detected sources from the images as the defects had many different shapes and often looked somewhat like stars. As these artefacts appear at different locations within different images they are unlikely to be pixel defects and are more likely to be shooting stars or satellites. As can be seen the line of pixels has been detected as many sources and as we are unable to accurately remove these detected artefacts, they have to be passed onto the machine learning algorithm, along with all other detected sources.

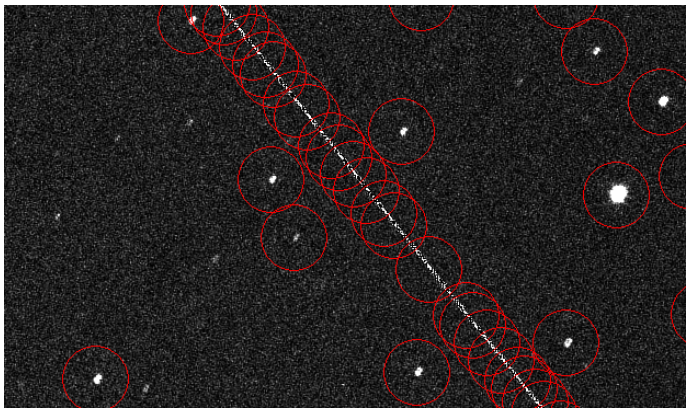


Figure 5: A large artefact within a GOTO image (most likely a satellite or shooting star). The red circles (detected sources) show that artefacts are routinely being detected as sources.

[6] *https://photutils.readthedocs.io/en/stable/api/photutils.detect_sources.html*
[7] *http://docs.astropy.org/en/stable/api/astropy.convolution.Gaussian2DKernel.html*

3.2.2 Matching to Galaxy Zoo

By allowing simple and easy galaxy classification, the Galaxy Zoo project has allowed over 200,000 volunteers to classify over 900,000 galaxies within the night sky, with accuracy similar to that of professional astronomers [Lintott et al, 2008]. Follow up projects such as Galaxy Zoo 2 [Willett et al, 2013] have produced a catalogue of positions as well as morphological properties, such as bars, bulges and strength of spiral arms of ∼300,000 of the brightest ∼900,000 galaxies in Galaxy Zoo 1. In order to find a representative sample of galaxies within GOTO images and thus create the training image sets of galaxy images, the galaxies present within GOTO images had to be identified. This was done by positionally matching all detected sources within a GOTO image to galaxies within the Galaxy Zoo 2 catalogue.

A total of 165 GOTO images covering 2640^{o2} of the same region of the night sky as the Galaxy Zoo 2 catalogue had the source detection algorithm outlined in section 3.2.1 and Appendix B (section 2.2) run on them. All sources detected within the images were saved as a catalogue with their right ascension and declination. Astropy `match_to_catalog_sky`[8] was used to positionally match the GOTO source catalogue to the Galaxy Zoo 2 catalogue. As GOTO is a low depth imaging telescope, faint/distant galaxies will not be visible/detectable. Galaxy Zoo 2 provided a smaller catalogue to compare with, speeding up matching and giving more information about the galaxies.

Initially, if a source detected within one of the GOTO images was located within 1" of a galaxy within the Galaxy Zoo 2 catalogue the source was deemed to be a galaxy. However, due to the relatively low resolution of GOTO, with a pixel scale of 1.4", comparing to within 1" leaves very little room for error when comparing the pixel position calculated from the Galaxy Zoo 2 catalogue. To allow the source detection algorithm some leniency, this was increased to within 5" or within ∼3 pixels of the galaxy location stated within the Galaxy Zoo 2 catalogue. Implementing this 5" leniency doubled the number of galaxies found within each image. As each source was matched to the Galaxy Zoo 2 catalogue its 'type' was noted. If the source was found in Galaxy Zoo 2 it was recorded as either a 'featured' or a 'smooth' galaxy, based upon its Galaxy Zoo 2 classification. If the source was found not to be in Galaxy Zoo 2 it was assumed to be 'other' - containing all sources that are not galaxies. This 'other' sample predominantly consists of stars, however also includes artefacts that have been detected as sources (section 3.2.1), galaxies that have somehow slipped through the matching process (see section 5.4.1) or other objects such as nebulae, etc. The number of sources detected within each of the 165 GOTO images are summarised in Table 1.

Type:	Featured Galaxies	Smooth Galaxies	Total Galaxies	Other Sources
Mean (i.e., per image):	27.5 ± 11.5	19.6 ± 9.3	47.2 ± 19.5	1,877.7 ± 542.1
Most (in one image):	70	55	129	3,134
Minimum (in one image):	6	3	13	1,020
Total (in all images):	4,486	3,183	7,669	307,824

Table 1: A summary of the types of sources detected within all of the 165 GOTO images used to create the training sets of images - the number of galaxies was found by comparing the detected sources to the Galaxy Zoo 2 catalogue.

Table 1 shows that there is a considerable variation in the number of sources per image. This variation is expected as if you are observing into or away from the Milky Way we would expect more or fewer stars and could potentially see no galaxies if observing into the plane of the Milky Way. Similarly, galaxies clump together in filaments/clusters, so depending on whether you are looking at a cluster or not, more or fewer galaxies would be expected. It should be noted that although we expect to see a variation in the number of stars and galaxies detected,

[8] *http://docs.astropy.org/en/stable/api/astropy.coordinates.SkyCoord.html*

this does not mean the detection and matching procedure is perfect. There are almost certainly galaxies being missed during the classification process, due to both the detection algorithm used on GOTO images as well as the matching procedure to the Galaxy Zoo 2 catalogue not being perfect - see section 5.4.1 for further discussion and ramifications.

3.2.3 Creating a Cutout Image of Every Source

The final stage of creating the training set of images was to cut out all sources detected within the survey images. This was done by creating a new 'fits' image for every source detected, cutting an area around each source centred on the central pixel position of that source. Initially an area of 50×50 pixels was cut out around each source as a 'postage stamp' image. It was found that all sources were smaller than 50×50 pixels and, due to the large size, there were often second or third sources present within the postage stamp image. In order to speed up the machine learning algorithm and to reduce the number of cutouts with multiple sources present, the postage stamps were decreased to a size of 30×30 pixels. <1% of detected sources in a survey image were larger than 30×30 pixels and are currently being ignored in favour of speeding up the machine learning classifier. These larger objects are both stars and galaxies, however, the features of these larger objects are much more clear and well defined, meaning a non machine learning classifier could potentially be used to classify these sources.

Once the source had been cut out it was saved into a file depending on its Galaxy Zoo 2 classification. If a source was not present within Galaxy Zoo 2, it was saved as an 'other' source. Table 1 shows the total number of sources from the 165 GOTO images covered Galaxy Zoo 2. This lead to the creation of 307,824 cutout images of 'other sources' aswell as 7,669 cutout images of galaxies taken with GOTO. These sets of cutout images must now be converted into a form capable of training (or being classified by) a CNN.

3.3 Preparing the Cutout Sources for Machine Learning.

With the source detection and cutout phase complete, the individual postage stamp images of sources were converted into a form that could be fed into a CNN. The following subsection outlines the steps taken to achieve this. Firstly, the data was normalised into a form that would allow a CNN to pick out features that described the images (section 3.3.1). Secondly, the data was augmented in order to increase the number of galaxies that could be used to train the machine learning algorithm (section 3.3.2). Finally, each image was transformed into a matrix and added to a tensor (a list of 2D matrices) capable of being fed into a machine learning algorithm (section 3.3.3).

3.3.1 Normalisation of the Data

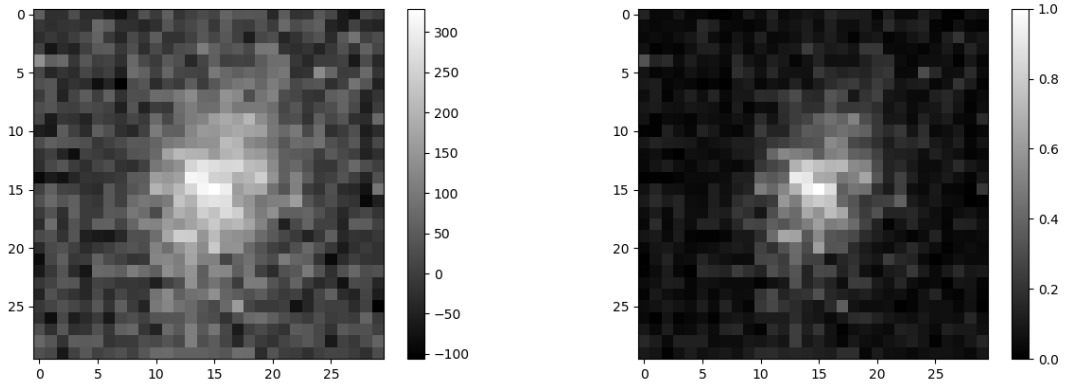
The normalisation of the data within machine learning is extremely important. As we are classifying sources based upon their morphologies (and not brightness or other properties) we must ensure other properties are not affecting the CNN's learning. To ensure the CNN is trained on morphology, the data was normalised such that every source appears to have the same brightness, allowing for comparison between, for example, two galaxies that may have similar morphologies, however vastly different brightness. To highlight the importance of normalisation, an initial ANN was unable to achieve greater than $\sim 65\%$ accuracy when discriminating between galaxies and 'other' sources. However, after re-normalising the exact same data, the same ANN was capable of achieving upwards of 90% accuracy, with even less training time.

Unfortunately, as with much of machine learning, the normalisation of data is very much trial and error, with no fixed way to achieve optimal results. For our data, we found that normalising each pixel in a manner described by equation 3 achieved optimal results, where $x_{normalised}$ is the new normalised pixel value, x_{pixel} is the raw pixel value, x_{min} is the smallest (or most negative) pixel value within the cutout image and x_{max} is the maximum pixel value.

This procedure normalised the pixel data between 0 and 1, in addition to squaring each pixel. This has the effect of brightening the already bright central pixels that form the galaxy or star, whilst making the darker surrounding pixels darker, effectively increasing the contrast of the image (whilst also keeping the data between 0 and 1). This allowed the algorithm to pick out features more easily. Most postage stamp cutouts had a mean close to zero, due to the majority of the background pixels having a value close to zero, however, each cutout did have a slightly different mean and standard deviation.

$$x_{normalised} = \left(\frac{x_{pixel} - x_{min}}{x_{max} - x_{min}} \right)^2 \tag{3}$$

An example of image normalisation is shown in figure 6, highlighting how the squaring process makes the central object appear brighter and reducing noise.



(a) Un-normalised Cutout

(b) Normalised Cutout

Figure 6: An example of a cutout image of a galaxy taken with the GOTO telescope on La Palma. a) shows the un-normalised raw image, b) shows the normalised image with pixel values ready to be fed into the machine learning algorithm

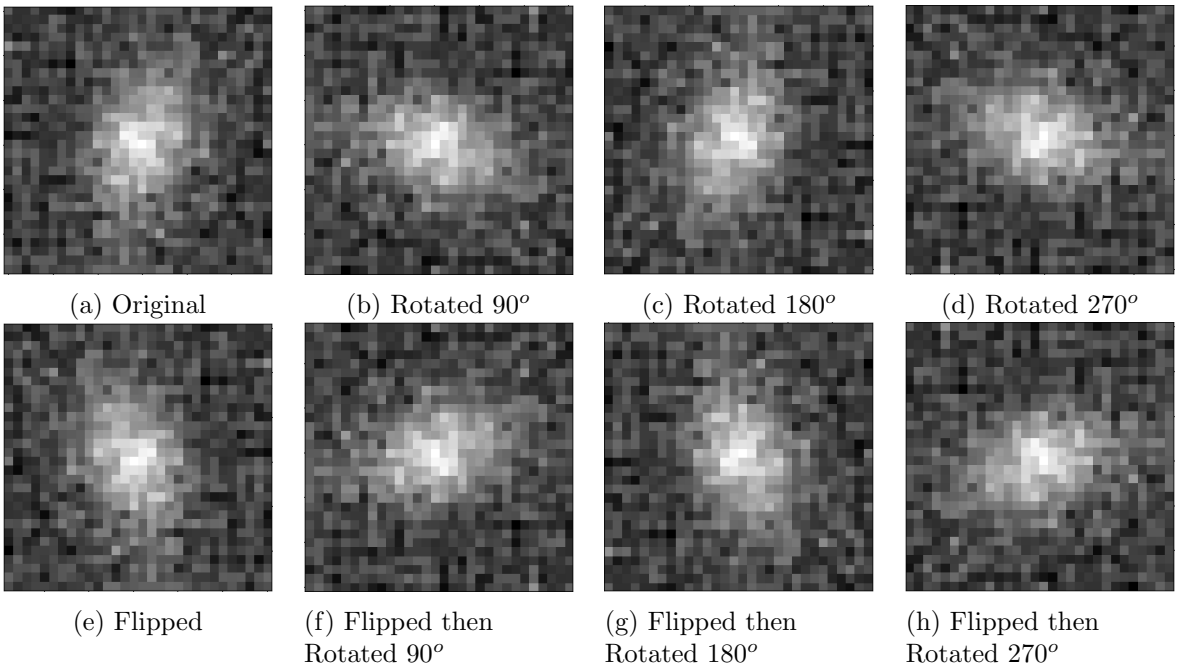
3.3.2 Augmentation of the Data

When training a machine learning classifier it is important to ensure roughly evenly sizes of training samples in order to get optimal results from the classifier. If uneven training samples are used the classifier will become very good at finding features that describe the dominant class, but will never learn the features that describe the smaller training sample. Furthermore, if uneven sample sizes are used when validating the learning of the algorithm it can often lead to inaccurate classifiers, as these calculations are affected by the number of samples passed through.

To achieve optimal results, we would ideally train the CNN on all sources imaged by GOTO. However, as can be seen in table 1, there is a large imbalance between the number of galaxies and 'other' sources (∼1:40) within a GOTO survey image. As evenly sized training samples are required, many 'other' sources had to be ignored and were not used when training the CNN. To allow as many 'other' sources as possible to be used in training, full utilisation of the galaxy images was needed and was achieved by augmenting the images of galaxies, vastly increasing the size of the galaxy sample.

In a similar fashion to Aniyan & Thorat [2017] who trained a CNN on 200 images augmented to over 36,000 images, the images within the galaxy training sample used to train the CNN were flipped horizontally and then both the original and flipped image were rotated 90^o, 180^o and 270^o. An example of this augmentation shown in figure 7, showing how each new image created is different from the original. This procedure increased the number of galaxy images by 8 times, from 7,669 images to 61,352 images and now allows us to feed more 'other' sources into the CNN

as well as help to improve rotational invariance as each new galaxy is just a rotation/flip, but treated as a new galaxy.



(a) Original (b) Rotated 90^o (c) Rotated 180^o (d) Rotated 270^o

(e) Flipped (f) Flipped then Rotated 90^o (g) Flipped then Rotated 180^o (h) Flipped then Rotated 270^o

Figure 7: An example galaxy image being augmented in order to increase the size of the training sample. Galaxy images were flipped, with both the original and flipped images then being rotated by 90^o, 180^o and 270^o with each being saved as a new image.

90^o, 180^o and 270^o rotations were chosen as right angle rotations of matrices are much easier to compute than intermediate angle rotations. Furthermore, intermediate angle rotations produced areas of unknown pixel value in the corners of the rotated image. This could be negated by either filling these corners with noise similar to that of the rest of the cutout, or producing larger (for example 50×50 pixel) cutouts, rotating them and then cutting the rotations down to 30×30 pixels in size. Neither of these solutions were picked as it was deemed unnecessary to increase the training size further and would take a lot of computational time, however, is something worth considering if more training samples are needed from a limited number of initial training images.

3.3.3 Converting to Tensor Form

To get the cutout images into a form which a CNN can learn from, the image cutouts were converted into tensor form (a long list of two dimensional matrices). 60,000 cutouts of both galaxies and 'other' sources were selected at random, converted in turn into a 30×30 matrix of pixel values (it was at this point the data was also normalised) and then added to a tensor. This created a tensor containing all cutout images, with dimensions (120000, 30, 30) and is called the features list. As the 30×30 matrices were being added to the tensor, the type of classification (galaxy=1 or 'other'=0) was added to a list called the labels list, producing a 120,000 element list of 0's and 1's ordered in the same way as the images in the features list (i.e., all galaxies in the features list are 1's in the label list). The machine learning algorithm can now work out what features of images, stored in the features list, correspond to their classification given by the labels list.

4 Creation and Training of the Convolutional Neural Network

In order to create a CNN capable of classifying images of sources taken with GOTO a sequential model was created via the TensorFlow[9] Python library. In order to utilise TensorFlow, Keras[10] was used to design and build the model. Keras allows for easy manipulation of number, types and sizes of layers outlined in section 2.3, within the TensorFlow library. The architecture of the CNN used to classify sources in this project is summarised in table 2 (`model.summary` produced by TensorFlow). The summary describes the layer types used, their order, output dimensions and the number of trainable parameters. The model is comprised of 6 convolutional layers, ranging in size from 70 to 250 neurons, each with a 3×3 kernel. Each convolutional layer was followed by a rectified linear unit ('ReLU') layer which deployed a rectifier activation function to all neurons. Towards the end of the model, the feature maps are pooled, with a dropout layer included to reduce overfitting. Finally three dense layers (similar to a regular neural network) are added and comprise the majority of trainable parameters. A final sigmoid activation function is made to decide the class of the image.

Layer Type	Output Dimensions	# Trainable Parameters
Input layer	(30, 30, 1)	0
2D Convolutional	(28, 28, 70)	700
ReLU Activation	(28, 28, 70)	0
2D Convolutional (2)	(26, 26, 110)	69,410
ReLU Activation	(26, 26, 110)	0
2D Convolutional (3)	(24, 24, 140)	138,740
ReLU Activation	(24, 24, 140)	0
2D Convolutional (4)	(22, 22, 180)	226,980
ReLU Activation	(22, 22, 180)	0
2D Convolutional (5)	(20, 20, 200)	324,200
ReLU Activation	(20, 20, 200)	0
2D Convolutional (6)	(18, 18, 250)	450,250
ReLU Activation	(18, 18, 250)	0
2D max pooling (2×2)	(9, 9, 250)	0
Dropout (25%)	(9, 9, 250)	0
Flatten	(20250)	0
Dense	(258)	5,224,758
ReLU Activation	(258)	0
Dense (2)	(128)	33,152
ReLU Activation	(128)	0
Dense (3)	(1)	129
Sigmoid Activation	(1)	0
Total trainable Parameters: 6,468,319		

Table 2: A summary of the CNNs architecture used to classify images of sources taken with GOTO. The model has a total of 6,468,319 trainable parameters, mostly from the large dense layer towards the end of the model.

After carrying out the steps outlined in section 3 a total of 60,000 pre-classified images of both galaxies (both featured and smooth galaxies - within Galaxy Zoo classifications) and 'other' objects, had been created. The data was then split into 3 groups to train, validate and test the CNN. From the 120,000 images 15% (9,000 of both galaxies and 'others') were removed completely from the training process and were never once seen by the CNN until the testing phase. During training, a further 15% of remaining images (7,650 of both galaxies and 'other'

[9] https://www.tensorflow.org/
[10] https://keras.io/

sources) were removed to be used to validate the learning and ensure no overfitting had occurred. This left a total of 72.25% (43,350 of both galaxies and 'other' sources) of the images being used to train the classifier. Training took place using a GeForce GTX 1050 Ti 4GB Graphics Card, utilising the GPU version[11] of TensorFlow, which sped up learning by several factors when compared to the CPU version. The train time per epoch (training iteration) for the CNN was ∼150 seconds. The code used to create and run the CNN will be uploaded to GitHub[12] in the near future, as well as the code used to create the training images outlined in section 3.

During training the accuracy of the CNN was monitored. Figure 8 shows the error (given by: N^o incorrect predictions / Total N^o of predictions) of classification on both the training and validation data as the model was trained for 50 iterations (∼2 hours of training). As can be seen, the error on classification of the training images continuously decreases with training, as expected. Surprisingly the error on the validation images decreases and then stays constant with training; this suggests that either 1) the dropout layer (see section 2.3) towards the end of the CNN has successfully removed the possibility of overfitting. 2) The model has not been trained for enough iterations to observe overfitting. 3) The images being used within training and validation are similar enough such that there are no small features within the training set that can be used to distinguish them from the validations set and could result in overfitting. It is most likely that the model has simply not been trained enough for overfitting to be observed as classifications are fairly complicated with such low detail images. It should also be noted that after ∼20 iterations of training there is no improvement to the accuracy of the validation.

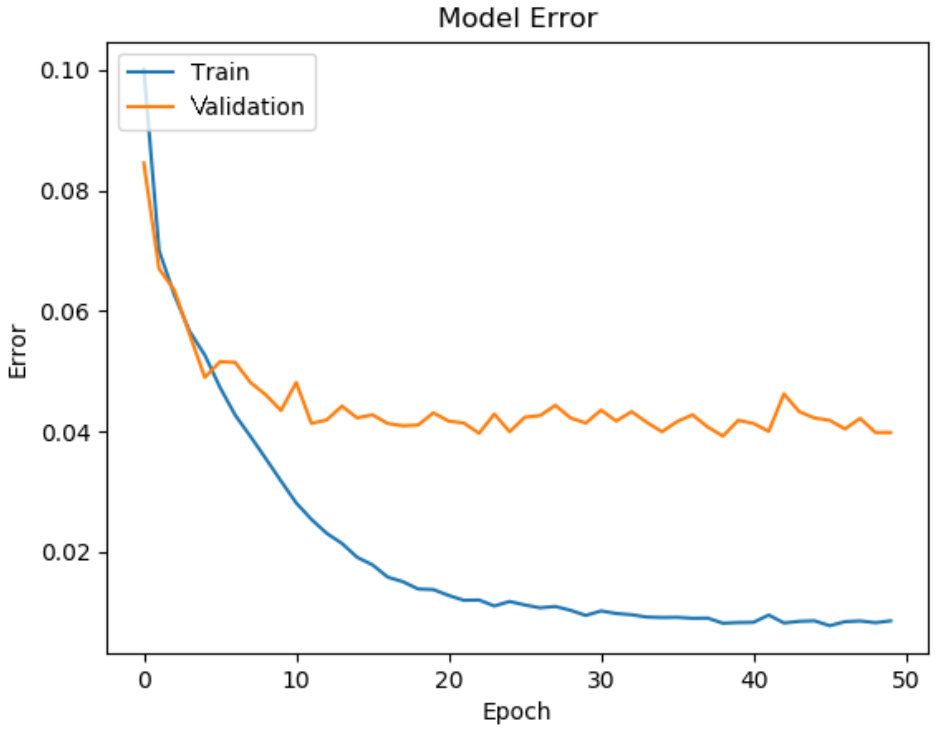


Figure 8: The evolution of the CNNs error over 50 epochs of training. The model was trained on 86,700 images (43,350 of both galaxies and 'other') and validated on 15,300 images (7,650 of both galaxies and 'other'), with each training epoch taking ∼150 seconds.

[11] *https://www.tensorflow.org/install/gpu*
[12] *https://github.com/Matthewkm*

5 Results and Discussion

The following section shows the results obtained by CNN when classifying unseen test data, as well as discusses the significant problem of unevenly sized test samples, such as the numbers of galaxies and 'other' sources within GOTO images. Section 5.1 discusses the metrics used to quantify the success of the CNN taking into account the uneven numbers of sources expected when classifying unseen GOTO images. Section 5.2 shows the results of the classifier on evenly sized test data created during the training phase (section 4). Section 5.3 demonstrates the problems encountered when unevenly sized testing samples are used and how this leads to poor precision when classifying unseen large GOTO survey images. Section 5.4 shows examples of correctly and incorrectly classified sources and discusses potentially incorrectly labelled sources. Finally, section 5.5 discusses several ways to improve the CNN.

5.1 Quantifying the Success of the Classifier

Due to the uneven numbers of sources within GOTO images, metrics that take the differently sized data samples into account have been used to quantify the success of CNN at distinguishing between sources in such images. The following metrics have been chosen: precision, recall and F1 score. Within these metrics, true/false denotes a correct/incorrect prediction, and positive/negative is a galaxy/'other'.

- Accuracy (equation 4): describes how accurate the model is at classifying images, however, doesn't take into account differently sized test samples.

$$Accuracy = \frac{Number\ of\ Correct\ Prediction}{Total\ Number\ of\ Predictions} \tag{4}$$

- Precision (equation 5): describes how precise a model is, looking at how many predictions of the positive class (galaxy) are correct, compared to the total number of positive guesses (galaxy predictions). (i.e., the proportion of galaxy predictions that were actually galaxies).

$$Precision = \frac{Number\ of\ True\ Positives}{Total\ Number\ of\ Positives} \tag{5}$$

- Recall (equation 6): describes how many positively predicted results are actually correct. (i.e., the proportion of galaxies that were predicted correctly)

$$Recall = \frac{Number\ of\ True\ Positives}{(Number\ of\ True\ Positives) + (Number\ of\ False\ Negatives)} \tag{6}$$

- F1 score (equation 7): a combination of both precision and recall, taking both false positives and false negatives into account

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{7}$$

Precision, recall and F1 score are much more useful metrics than accuracy if there is an uneven class distribution, where positives and negatives cannot be treated with equal weight, as is the case with images of the night sky, where there are many more stars than galaxies.

5.2 Classifying the Test Data

After the model had been trained for 50 epochs, it was used to classify the 18,000 unseen test images. Table 3 shows the results of the CNN when classifying these equally sized samples of test images.

		8,937	9,063	Total
True Label	Other	8,621 (0.96)	379 (0.04)	9,000
	Galaxy	316 (0.03)	8,684 (0.97)	9,000
		Other	Galaxy	
		CNN Predicted Label		

Table 3: A confusion matrix showing how 18,000 testing images were classified by the CNN that had been trained and validated on 86,700 and 15,300 images respectively. The numbers in brackets represent the normalised number of images classified when comparing to the total number of images in that class (the accuracy).

When considering evenly sized testing samples, the model achieves >96% accuracy when distinguishing between images of galaxies and 'other' objects imaged with a low resolution telescope. Whilst this accuracy is high, it does not translate to the model being useful at classifying uneven samples of data. Due to the nature of the data, there are many more 'other' sources present than galaxies within a GOTO survey image. This is an issue as errors will be magnified on the larger ('other') sample, when compared to the smaller sample. i.e., the error on 'other' sources, although being the same size as that for galaxies, results in a much larger number of sources. Due to the fact that accuracy calculations do not take unevenly sized data samples into account, error and accuracy are not good metrics to be used to quantify the success of a model in situations with uneven data samples.

5.3 Testing the CNN on Realistically Sized Data Samples

To truly test the success of the CNN on uneven data samples, like the unseen GOTO survey images it will be used to classify, a test set of images containing sample sizes proportional to values in table 1 (40:1 - 'other':galaxies) was produced. This test set of images, containing 188 galaxies and 7512 'other' sources, was then classified by the CNN. Table 4 shows the results of these images being run through the classifier, where the classifier is rounding the output neuron to either 1 (a galaxy) or 0 ('other'), in order to classify the image.

		7,196	504	Total
True Label	Other	7,189	323	7,512
	Galaxy	7	181	188
		Other	Galaxy	
		CNN Predicted Label		

Table 4: A confusion matrix showing how unseen test sources (7,512 'other' and 188 galaxies), weighted as example GOTO survey images (40:1 - 'others':galaxies) were classified by the CNN that had been trained and validated on 86,700 and 15,300 images respectively. The output neuron was rounded to 1 or 0 to classify the source as either a galaxy or 'other'.

Despite the classifier having high accuracy (>96%), with only 7 of the 181 galaxy images being incorrectly predicted (equates to <2 incorrect galaxy classifications per survey image), due to the number of 'other' sources outnumbering galaxies by ∼40×, the number of incorrectly

classified 'other' sources is greater than that of the total number of galaxies within an image. This produces very low precision when classifying galaxies within a survey image, shown in table 6

In an attempt to reduce this issue, a further classification on the same images was made. However, rather than the model rounding the output neuron to either 1 or 0 (galaxy or 'other') the CNN would now classify any source with an output neuron below 0.95 as a star. Meaning the only classifications of galaxies would be those where the CNN was >95% sure the image was that of a galaxy. Table 5 shows the results of the neural network treating sources with an output neuron of below 0.95 as 'other'. The metrics outlined in section 5.1 have been calculated using numbers in tables 3, 4 and 5 and are shown in table 6.

		7,408	292	Total
True Label	Other	7,391	121	7,512
	Galaxy	17	171	188
		Other	Galaxy	

CNN Predicted Label

Table 5: A confusion matrix showing how unseen test sources (7,512 'other' and 188 galaxies) weighted as example large images (40:1 - 'others'[stars]:galaxies) were classified by the CNN that had been trained and validated on 86,700 and 15,300 images respectively. Only sources with >95% probability of being a galaxy were classified as galaxies, with the rest being classified as 'other'.

Metric	Evenly Sized Data	Proportional Data (rounded)	Proportional Data (>95%)
Accuracy	0.961	0.957	0.982
Precision	0.958	0.359	0.586
Recall	0.964	0.963	0.910
F1 Score	0.956	0.524	0.713

Table 6: The accuracy, precision, recall and F1 score calculated for 3 testing samples of images. The first column calculated using values within table 3 comprised of 18,000 test samples of equal sizes (9,000 galaxies and 'others'). The second column calculated with values from table 4, with test samples of sizes proportional to that expected in a GOTO sky survey image, with classifications made by rounding the output neuron to 1(galaxy) or 0('other'). The third column of metrics are calculated using values in table 5, with test samples proportional to that expected in a GOTO sky survey image, however only sources predicted to have a >95% chance of being a galaxy by the CNN were classified as galaxies, with all remaining sources being classified as 'other'.

When considering even training samples the CNN achieves high accuracy, precision, recall and F1 score, however these results are almost redundant due to the unevenly sized data present within sky survey images.

When considering results calculated using sample sizes proportional to the number of sources present within GOTO sky survey images, the accuracy of the classifier does not change, as accuracy equations do not take into account the size of the samples. However, the precision of the classifier decreases dramatically from 95% to 36%. This decrease in precision is expected as the number of incorrectly classified 'other' sources outnumbers the total number of galaxies within an image. This is a significant issue; if the CNN was used to produce a list of sources that it deemed to be galaxies within a GOTO image only ∼36% of the sources would actually

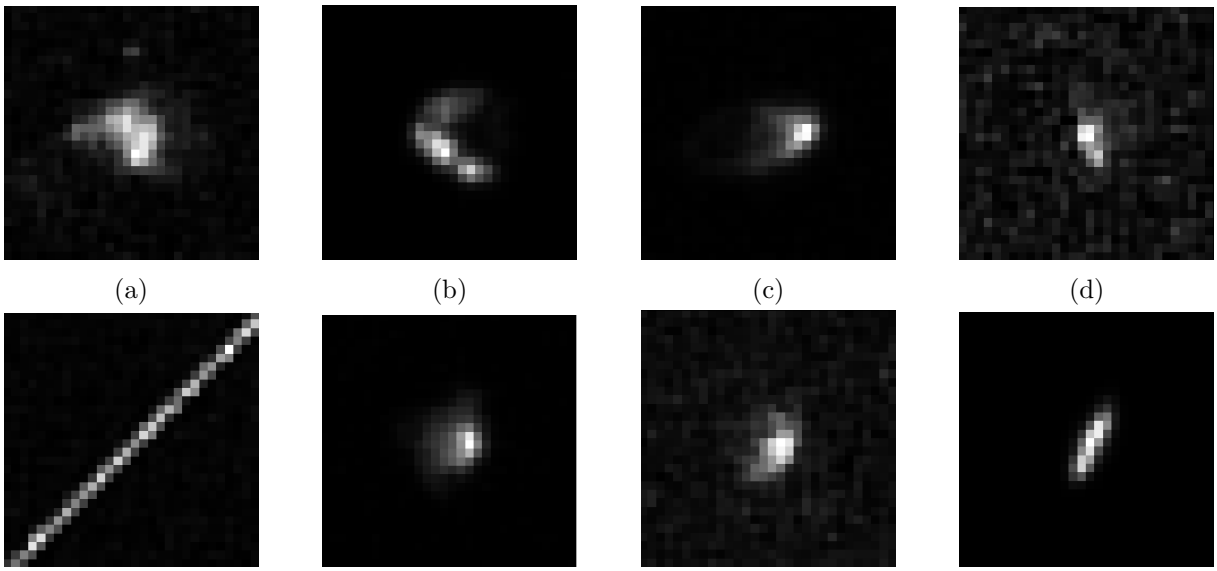
be galaxies. This fact highlights how the CNN is much less successful when considering data in proportion to sky survey images, than when compared to the testing phase of learning (section 5.2) where the model is tested on equally sized samples. Uneven sample sizes are a problem within all aspects of machine learning and it is often near impossible to negate the problem. This low precision results in a fairly low F1 score of 0.71. Recall remains high, as most of the galaxies have been found, with only 7 of the 181 galaxies being missed.

Using the CNN and only classifying sources with >95% chance of being galaxies as galaxies improves the model by a considerable margin. The recall drops marginally from 96% to 91% as more galaxies are missed ($\sim$4-5 of the $\sim$ 47 galaxies per image), which is smaller than expected and shows that <5% of galaxy images were deemed to have a probability prediction between 50% and 95% of being a galaxy. As such, this drop in recall is small when considering the other improvement to the model. The precision of the classifier increases significantly from 36% to 59%, as many more 'other' sources are now being classified correctly, decreasing the number of incorrectly classified 'other' sources, which now no longer outnumber correctly classified galaxies (table 5). The F1 score has increased from 0.52 to 0.71, a substantial improvement. The model can now produce a list of galaxies within a sky survey image, where $\sim$ 59% of sources are actually galaxies and only misses 4-5 (8-10%) of galaxies per survey image. Whilst this result is a definite improvement, the uneven sample sizes cause a large problem with achieving good precision and an improved model/data is needed to achieve a more precise CNN.

5.4 Examples of Classified Sources

Whilst a fairly modest precision and F1 Score have been achieved, when considering the low resolution and image distortion within the data, it is understandable as to why the CNN struggles to classify some sources. Figures 9 and 10 show several correct classifications made by the CNN for 'other' sources and galaxies respectively. 'Other' sources shown within figure 9 have fairly high levels of distortion or strange morphologies, such as figure 9a, 9b and 9c. As these sources are most likely stars, it is unclear as to why their morphologies are so distorted. Figure 9e shows an example of a line artefact shown in section 3.2.1, which the CNN has correctly identified as an 'other' source. Figure 10 shows examples of galaxies that have been correctly classified as galaxies by the CNN. Whilst the morphologies of these galaxies changes fairly dramatically from image to image (e.g. 10c and 10d) most of these galaxies look like representative galaxies, hence the CNN has had no issues correctly classifying them.

'Other' Sources Classified Correctly

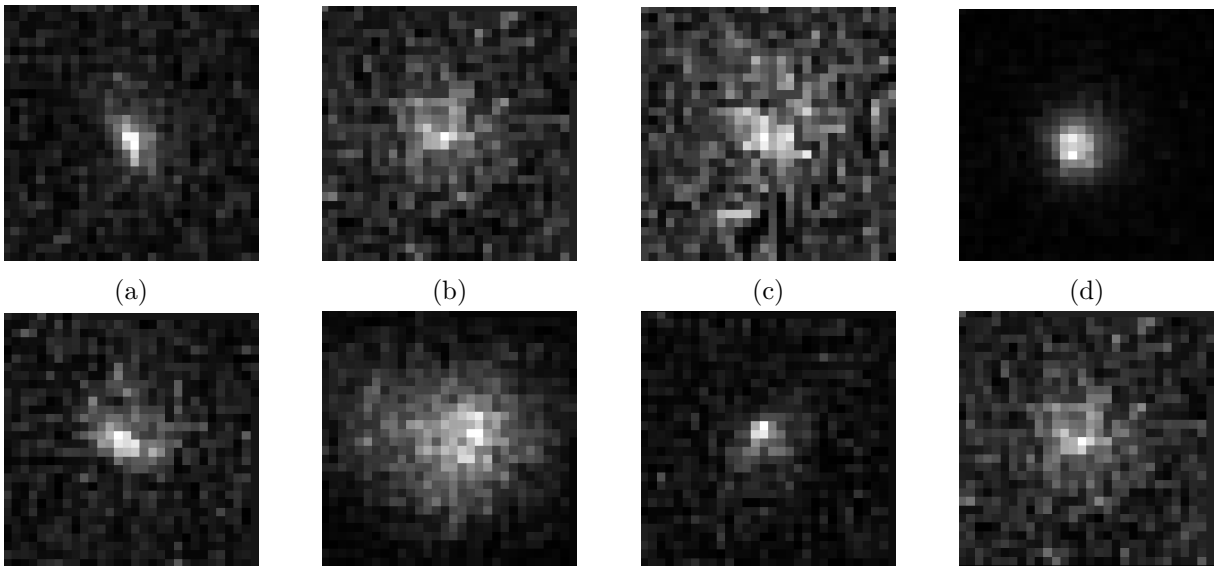


(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 9: A selection of sources labelled as 'other' that were correctly classified as 'other' sources by the CNN.

Galaxies Classified Correctly



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 10: A selection of sources labelled as galaxies that were correctly classified as galaxies by the CNN.

Figures 11 and 12 show several examples of sources that have been incorrectly classified by the CNN. The incorrectly classified sources are generally much more complex and often look neither like stars nor galaxies, to the point where human classifiers would struggle to accurately classify these sources (e.g. figures 11b, 11f, 12a, 12f and 12g). Many sources incorrectly classified as 'other' (figure 12) often appear to be very small or faint galaxies (e.g. figures 12b, 12c and 12f) and as such the classifier has incorrectly classified them. However, figures 12d and 12h both look like representative galaxies so it is unclear why the CNN has incorrectly classified these sources. One potential explanation is that incorrectly labelled sources may be impeding the CNN's ability to pick out features accurately.

Labelled 'Other', Classified Incorrectly as Galaxies

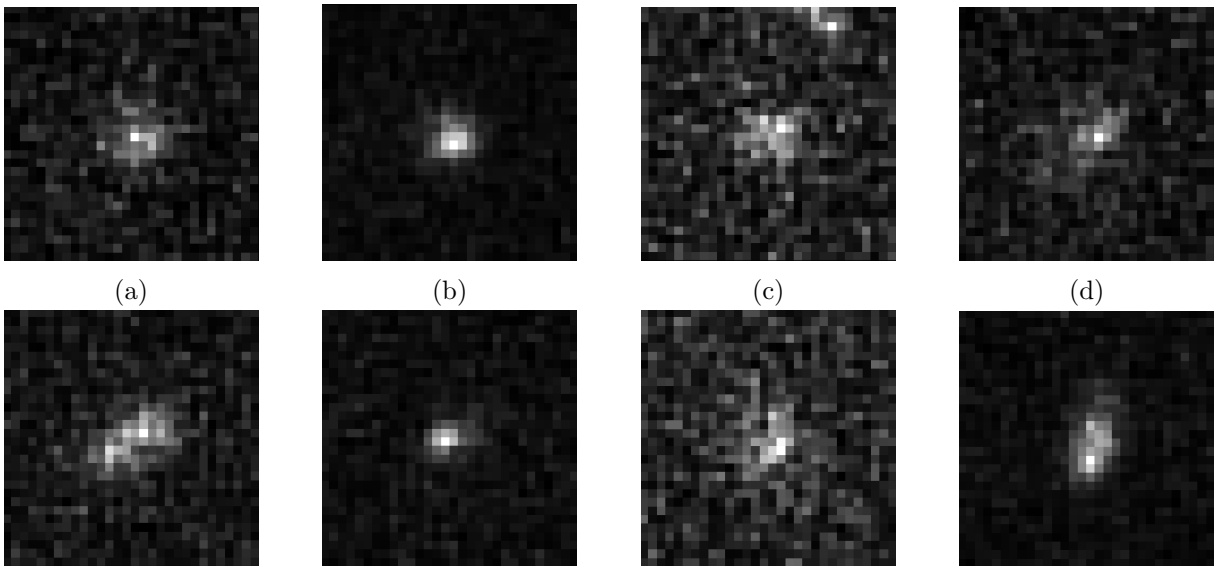


Figure 11: A selection of sources labelled as 'other' that were incorrectly classified as galaxies by the CNN.

Labelled Galaxy, Classified Incorrectly as 'Other'

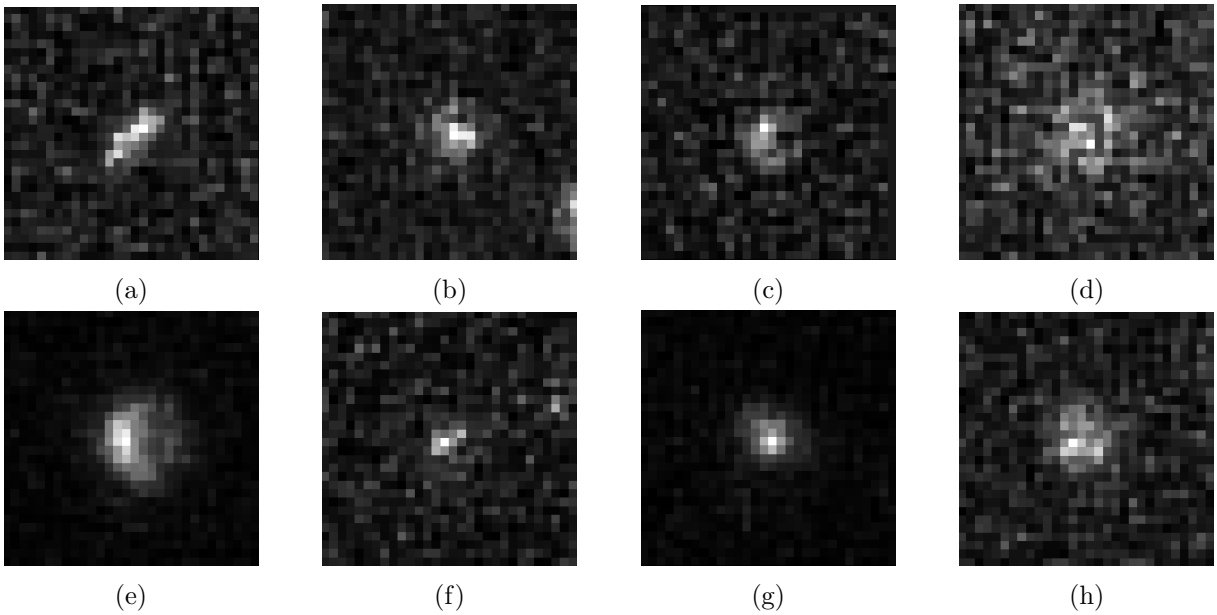


Figure 12: A selection of sources labelled as galaxies that were incorrectly classified as 'other' sources by the CNN.

5.4.1 Incorrectly Labelled Sources

A major point of concern is the fact that some sources classified by the CNN may have been incorrectly labelled as the opposing class. Clear examples of this are shown in figures 11a, 11c and 11g which all look like fairly clear galaxies, however, have been labelled as 'other' sources. This shows that some sources may have been incorrectly labelled as 'other' sources during the matching process outlined in section 3.2.2. There are several reasons why sources may have been incorrectly labelled. It is potentially possible that there are galaxies that were detected within the source detection phase (section 3.2.1) of creating the training sets of images which are not contained within the Galaxy Zoo 2 catalogue as they have been missed during the preselection phase of Galaxy Zoo. Galaxy Zoo uses galaxies identified within the SDSS when selecting galaxies to pass to volunteer classifiers. The SDSS utilises band images, shape profiles and spectral data amongst other features to flag whether a sources is a galaxy [York et al, 2000], so it is unlikely a source will be incorrectly classified as a star and missed from the SDSS classification. It is also possible that as Galaxy Zoo 2 only contains 300,000 galaxy classifications (whilst the SDSS contains over 50million identified galaxy) Galaxy Zoo 2 is simply not complete enough to contain all galaxies detected within the GOTO survey images. However, it is most likely that source positions calculated within the source detection phase do not closely match (within 5") the positions of galaxies stated within Galaxy Zoo 2, resulting in galaxies detected within GOTO images not being flagged as galaxies and automatically being classed as 'other' sources. Or vice versa; 'other' sources detected in GOTO images may have positions close (within 5") to galaxy positions stated within Galaxy Zoo 2, resulting in 'other' sources being incorrectly labelled as galaxies.

It appears the CNN has been able to correctly classify galaxies labelled as 'other' sources as galaxies. However, these incorrect labels will have a significant impact when calculating precision, recall and F1 score, as many of the incorrectly labelled 'other' sources may in fact be galaxies. It is unclear how many sources have incorrect labels, however, it is quite possible that the CNN may be performing better than calculated in section 5.3.

5.5 Improving the Convolutional Neural Network

Whilst the fact that unevenly sized data decreases the precision and therefore F1 score of any machine learning classifier, it is still possible to improve the results of the CNN. Improving the accuracy of the CNN will reduce the number of incorrect 'other' source classifications that dilute the correct galaxy predictions. However, due to the significantly uneven data samples (1:40 - galaxy:'other') very high accuracies are required to achieve good precision and F1 scores. There are three ways to improve the accuracy outlined in this subsection: 1) improved CNN architecture (section 5.5.1), 2) cleaning/improving the data entering the CNN (section 5.5.2) and 3) reducing the amount of data needed to be classified with the CNN (section 5.5.3).

5.5.1 More Complex Architecture

Although a fairly complex CNN has been created with over 6 million trainable parameters, it is still relatively simplistic compared to many advanced cutting edge image classifiers. Due to limitations in computing power, more complex classifiers could not be produced or trained. More complex architecture with more training carried out on more powerful hardware would almost certainly improve the accuracy of classifications. Furthermore, with regard to most of machine learning, CNN results are very trial and error - with results often coming from a 'black box' where small tweaks to certain parameters can make large differences to results. It may be possible to improve the CNN by simply tweaking and optimising these parameters.

However, as highlighted in section 5.4, it is quite probable that the limiting factor of the CNN is the bad quality data, as well as the fact that some data is labelled incorrectly and many artefacts are being passed onto the CNN. It is possible that even with more complex

CNN architecture no significant improvements to accuracy would be obtained, as it is simply impossible to pick out features that differentiate the images.

5.5.2 'Cleaning' of Data

Artefacts:

Potentially one of the most substantial ways to improve the accuracy and precision of the CNN is to clean the data before it enters the classifier. Many sources within the 'other' samples were passed onto the machine learning algorithm that were clearly not stars, but some form of artefact (e.g. figure 9e). These artefacts (section 3.2.1) such as asteroids, satellites and potentially pixel defects, often looked very different from one another (but with no clear way to distinguish them via a hard coded algorithm). As such, the machine learning algorithm will have difficulty training to classify sources that look different from one another as just one 'other' classification. Removing artefacts detected as sources could significantly improve the accuracy and precision of the CNN.

Improvements to the source detection algorithm could potentially increase the quality of the data. Not only would this reduce the number of artefacts entering the CNN, but would also reduce the number of large sources being detected as more than one source, or detection of faint sources, which may not be classifiable by the CNN.

One potential way to ensure only stars and galaxies were passed onto the CNN could be to positionally match all sources detected within a GOTO survey image to the SDSS. All sources that did not match with already observed sources would be classified as 'artefacts'. The CNN could then be trained on three classes: galaxies, stars and artefacts. However this still leaves the problem that many artefacts do not look similar to each other, so the CNN may still struggle to pick out features that describe these artefacts.

Cutout Images with Multiple Sources Present:

Another potential issue is the fact that some sources are so close to one another in the sky that they appear as a secondary source in the cutout of another source; this could potentially confuse the CNN as this cutout looks very different from other sources within its class.

The CNN should learn fairly quickly that pixels on the outskirts of the image do not contain much information about the source located in the centre of the image. As such, the weights and biases of neurons examining pixels towards of the edge of the cutout, where any secondary source in the image will be located, will be lower and should not greatly affect the classification. As a result, a secondary source within a cutout may not significantly affect the accuracy or precision of the classifier. An example of a secondary source present within an image not affecting classification is shown in figure 13, where a large source towards the top right of the postage stamp image did not affect the classification process; the CNN still correctly classified the central source as a galaxy.

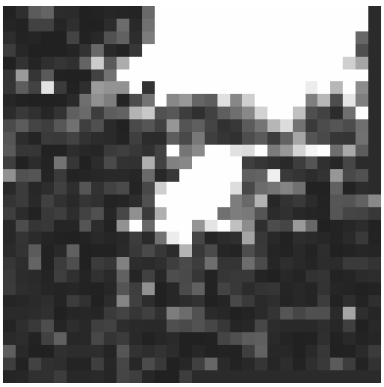
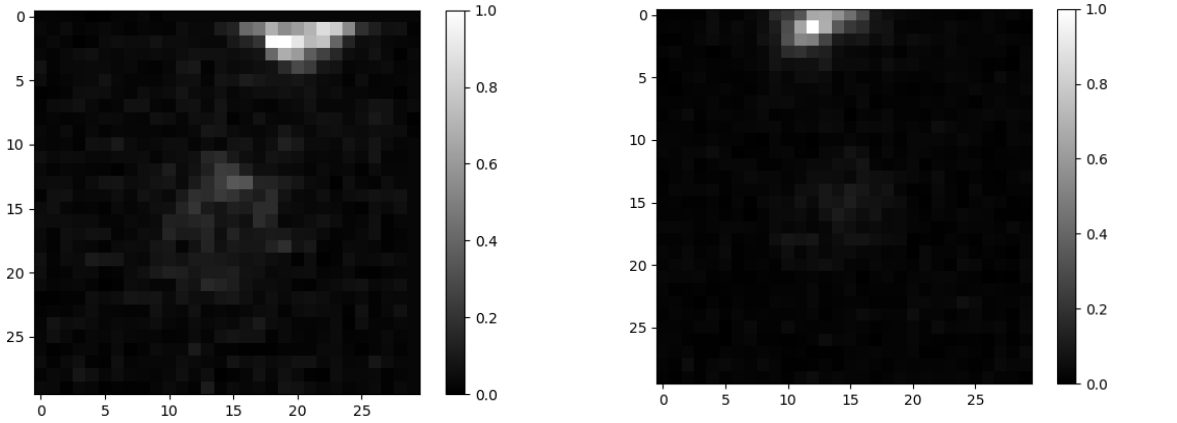


Figure 13: An cutout image taken with GOTO of a galaxy with a second source present in the top right. The CNN identified correctly that the source at the centre of the image is a galaxy.

However, a further and much more significant issue with regard to secondary sources present within cutout images is if the secondary source is brighter than the central one. If this is the case, the central source is incorrectly normalised due to the brighter pixels present in the secondary source. Figure 14 shows two normalised images of different faint galaxies with bright secondary sources present within the cutouts. As a result of the brightest pixel being within the secondary source, the central source has not been normalised such that its pixel values are between 0 and 1 (section 3.3.1). As cutout images of faint sources with brighter secondary sources present have not been normalised correctly, the CNN will struggle to classify such images, as the central source's features will not match other correctly normalised images within the class. Despite being incorrectly normalised, the CNN was still able to correctly classify the cutout shown in figure 14a. However, it was unable to classify the cutout shown in figure 14b correctly. This may be due to the central galaxy of figure 14a being relatively brighter, allowing the CNN to pick out enough features to make a correct classification.



(a) Correctly classified as a galaxy by the CNN

(b) Incorrectly classified as an 'other' source by the CNN

Figure 14: Two normalised (section 3.3.1) cutout images of galaxies with secondary brighter sources present. Due to the secondary sources, the image has not been normalised correctly, with the central galaxies being almost too faint to see.

5.5.3 Data Reduction

The primary reason for the CNN's bad precision is due to the massive imbalance between the number of galaxy and 'other' sources within GOTO survey images. If a hard coded algorithm could be utilised to remove the majority of 'other' sources that were definitely stars before the machine learning phase, the numbers of galaxy and 'other' sources having to be classified by the machine learning algorithm would be comparable in size. This would vastly increase the precision of the CNN as incorrectly classified 'other' sources would not outnumber the accurate galaxy predictions. Secondly, during the training phase, such an algorithm would: 1) create more equal numbers of training images of galaxies and 'other' sources, 2) speed up classification of unseen sources as the majority of sources could be removed and fewer sources would have to enter the CNN, and 3) potentially increase accuracy as the CNN would be trained on galaxies and 'other' sources that looked similar to galaxies, allowing it to learn to find small nuanced differences and not just learn to tell the difference between mostly well defined stars/defects and galaxies

Attempts to create a hard coded algorithm to remove definite stars as well as artefacts were made (appendix B - section 2.3). `Source_properties` was used to to extract many properties of each source detected including ellipticity, size and brightness. These properties were used to try and remove obvious stars from the detected sources. However, the naive assumption that galaxies within GOTO images could be differentiated from stars using only a few basic parameters turned out to be incorrect. For example, figure 15 shows sources detected within

a GOTO image whereby green/yellow circles indicate a featured/smooth galaxy (according to Galaxy Zoo 2) while red circles indicate a star (or other). As can be seen, galaxies appear very similar to stars and are often the same size and shape as stars, with only slightly different light profiles. With no clear way to distinguish between galaxies and 'other' sources using extracted source properties, the hard coded algorithm proved unsuccessful.

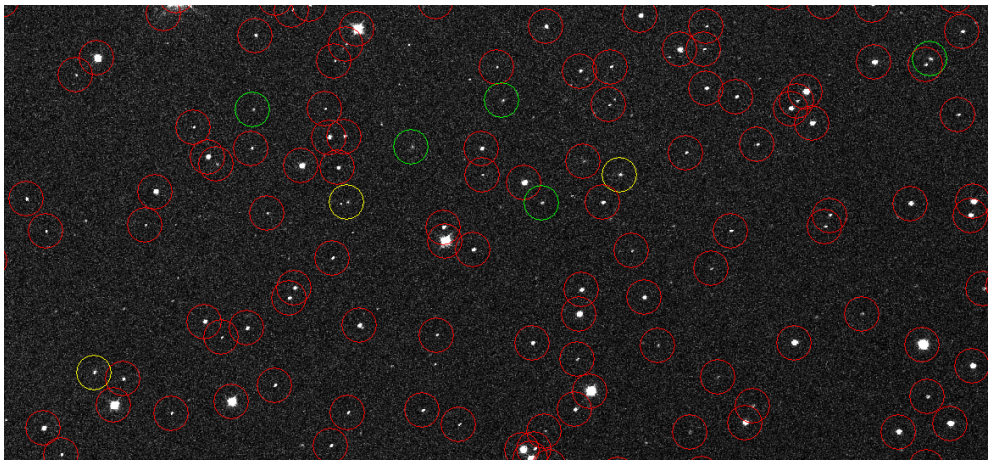


Figure 15: A section of a survey image taken with the GOTO telescopes on La Palma. Sources with green/yellow circles around them are featured/smooth galaxies respectively according to Galaxy Zoo classification. Sources with red circles around them are other sources, predominately stars, but potentially also artefacts, such as shooting stars, satellites or pixel defects.

6 Subclassification of Galaxies - Proof of Concept

So far we have only considered using a CNN to differentiate between galaxies and 'other' sources within GOTO survey images. However, within astronomical source classification an area of particular interest has been the further subclassification of galaxies into Hubble's 'tuning fork' classification scheme [Hart et al, 1971] allowing for study into the creation and evolution of galaxies. As a proof of concept, the same CNN architecture outlined in section 4 was re-trained to reproduce Galaxy Zoo 2 classifications of galaxies imaged with GOTO, classifying them as either featured and smooth galaxies, similar to Hubble's 'tuning fork' classification scheme.

Using the same method outlined in section 3.2, training sets of pre-classified galaxy cutout images were created using classifications within Galaxy Zoo 2. There were a total of 4,537 featured and 3,234 smooth galaxies present within the 165 GOTO images covered by Galaxy Zoo 2. These cutout images of galaxies were augmented (section 3.3.2) to form training sets of 25,000 images of both featured and smooth galaxies. Figure 16 shows the CNN being trained for 50 iterations, achieving maximum accuracy after ∼30 epochs of training, however, the validation error still gradually decreases as training continues.

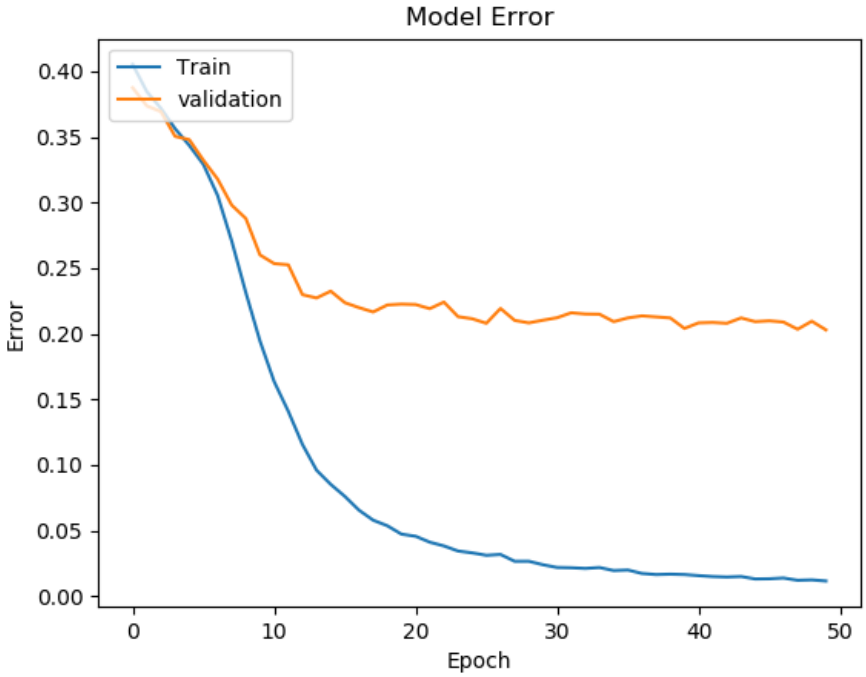


Figure 16: The evolution of the CNN's error over 50 epochs of training. The model was trained on 36,125 images and validated on 6,375 images, with each training epoch taking 60 seconds.

The CNN was tested on 3,750 unseen images of both featured and smooth galaxies (as classified by Galazy Zoo 2). Table 7 shows the results of the CNN. The accuracy of CNN is ∼81% and as equally sized test samples have been used the precision and recall are high, at 81% and 80% respectively. This equates to an F1 score of 0.851, which is lower than that achieved for equally sized test samples of galaxy and 'other' source predictions shown in table 6. This is due to the much more complicated nature of distinguishing between featured and smooth galaxies, especially with low resolution data, as opposed to making galaxies and 'other' source classifications. However, as roughly equal numbers of featured and smooth galaxies are present within GOTO images, the precision and F1 score are higher than that of galaxy and 'other' source predictions when test samples proportional to unseen GOTO images are used.

		3,804	3,696	Total
True Label	Featured	3,046	704	3,750
	Smooth	758	2,992	3,750
		Featured	Smooth	
		CNN Predicted Label		

Table 7: A confusion matrix showing how unseen test sources (3,750 images of both featured and smooth galaxies) were classified by the CNN that had been trained and validated on 36,125 and 6,375 images respectively.

This CNN is only a proof of concept classifier as there is currently no way to accurately find all galaxies within an unseen GOTO image, with the best results only having ∼58% precision. In order for this classifier to be used a much more precise classifier is needed to detect all galaxies within a GOTO image, with very few 'other' sources being incorrectly classified as galaxies. However, this classifier does show that if high precision could be reached it is also possible to make galaxy subclassifications (into featured and smooth) with upwards of 80% accuracy with low resolution images of galaxies.

7 Conclusion

To summarise, in this report, a CNN was created to differentiate between galaxies and 'other' sources (stars, artefacts and defects) purely using morphological properties within low resolution (1.4"/pixel) survey images taken with GOTO telescopes. The CNN was trained, validated and tested (split 72.25%/12.75%/15% respectively) on 60,000 images of both galaxies and 'other' sources, created by matching detected sources to the Galaxy Zoo 2 catalogue and augmenting images of galaxies to increase the training sample as well as improve rotational invariance. The CNN achieved high accuracies (>96%) when classifying evenly sized test samples. However, due to the extremely uneven numbers of galaxies and 'other' sources within GOTO survey images (∼1:40) the precision and F1 score of the CNN were low (35% and 0.586 respectively) when classifying data proportional to GOTO survey images. In order to increase the precision of the classifier, only sources with >95% probability predictions of being galaxies by the CNN were classified as galaxies, with all other sources being classified as 'other'. This in turn reduced the number of incorrectly classified 'other' sources by ∼2.7 times, increasing the precision of the CNN to 59%, while only mildly reducing the recall from 96% to 90% as surprisingly few extra galaxies were classified incorrectly. This in turn increased the F1 score to 0.713. However, the CNN may be performing better than calculated due to inaccurately labelled sources affecting precision and recall calculations. Even with this fairly low precision the CNN is capable of significantly reducing the number of sources that are potentially galaxies within a GOTO survey image from 1,900 sources containing ∼47 galaxies, to a list of 73 sources (96% reduction) containing ∼42 galaxies, whilst only missing ∼4-5 (∼8-10%) galaxies per image.

Whilst CNNs have routinely shown unparalleled accuracy within classification tasks, due to the uneven nature of data within sky survey images, it is very difficult to produce high precision classifiers, as any error on the dominant class will outweigh the smaller class. The primary way theorised to improve the CNN outlined in the report is to create a hard coded algorithm to reduce the data by removing definite stars and defects, creating more evenly sized samples as well as allowing the CNN to learn how to differentiate between more complex objects more effectively. Additionally, improving the quality of data (i.e., removing artefacts, faint or distorted sources) being fed into the CNN could significantly improve the success of the classifier. An additional issue noted is that some training and test sample images may be incorrectly labelled. Whilst the CNN seems to be capable of classifying these incorrectly labelled sources correctly, these sources will affect the training of the CNN, due to their features not matching their labelled class. Ensuring no sources are incorrectly labelled during the matching phase (section 3.2.2) of creating the training set of images may significantly improve the CNN. Finally, improving the CNN's architecture may also produce a more precise classifier, however, the low quality of data and incorrectly labelled sources are the most likely factors limiting the success of the CNN.

Finally, as a proof of concept the CNN has been shown to be capable of reproducing classifications of featured and smooth galaxies, as described in Galaxy Zoo 2 classifications, with 81% accuracy. However, without a better precision classifier to find galaxies present within unseen GOTO images, this secondary classifier cannot be used, but does show that such subclassification of galaxies is possible with low resolution data. We aim to make the code used to create and run the CNN, as well as the code used to create the training sets of images outlined in section 3 and 6 publicly available via GitHub[13] in the near future.

References

Abraham S. et al, **2018**, "Detection of Bars in Galaxies using a Deep Convolutional Neural Network" arXiv:1711.04573v2

Aniyan A. K. & Thorat K., **2017**, "Classifying Radio Galaxies With CNN", arXiv:1705.03413v1

[13] https://github.com/Matthewkm

Banerji M. et al, **2010**, "Galaxy Zoo: Reproducing Galaxy Morphologies via Machine Learning", arXiv:0908.2033v2

Bazell D. & Aha D. W., **2001**, "Ensembles Of Classifiers For Morphological Galaxy Classification", Apj 548 : 219-223

Calleja J. & Fuentes O., **2004**, "Automated Classification of Galaxy Images", LNCS, volume 3215

Dieleman S. et al, **2015**, "Rotation-invariant CNNs for galaxy morphology prediction" Royal Astronomical Society, Volume 450, Issue 2

Gauthier A. et al, **2016**, "Galaxy Morphology Classification", Stanford University

Gershenson C., **2003**, "Artificial Neural Networks for Beginners", University of Sussex

GOTO "The Gravitational-Wave Optical Transient Observer" 2018, web page: *https://goto-observatory.org/*

Hart R. & Berendzen R., **1971**, "Hubble's Classification Of Non-Galactic Nebulae, 1922-1926", Boston University

Hocking A. et al, **2017** "An automatic taxonomy of galaxy morphology using unsupervised machine learning", arXiv:1709.05834v1

Huertas-Company M. et al, **2010** "Revisiting the Hubble sequence in the SDSS DR7 spectroscopic sample: a publicly available Bayesian automated classification", AA 525, A157

Kim E. J. & Brunner R. J., **2016**, "Star-galaxy Classification Using Deep CNNs" arXiv:1608.04369v2

Raddick M. J. et al, **2009**, "Galaxy Zoo: Exploring the Motivations of Citizen Science Volunteers"

Rojas R., **1996**, "Neural Networks", Springer

LeCun Y. et al, **2015**, "Deep learning" Nature volume 521, pages 436–444

Lintott C. J., **2008** "Galaxy Zoo: Morphologies derived from visual inspection of galaxies from the Sloan Digital Sky Survey" arXiv:0804.4483v1

Schutter A. & Shamir L., **2015**, "Galaxy morphology — An unsupervised machine learning approach", Astronomy and Computing 12 (2015) 60–66

Storrie-Lombardi M. C. et al, **1992**, "Morphological Classification of galaxies by Artificial Neural Networks", Monthly Notices of the Royal Astronomical Society, Volume 259, Issue 1

Willett K. W. et al, **2013**, "Galaxy Zoo 2: detailed morphological classifications for 304,122 galaxies from the Sloan Digital Sky Survey"

York D. G. et al, **2000**, "The Sloan Digital Sky Survey : Technical Summary" The Astronomical Journal, 120 : 1579-1587.

Appendices

A Semester One Project Plan

This section outlines the intended objectives over the course of the second semester. All dates are estimates. We intended to practice implementing CNNs throughout the whole semester, hopefully making it easier to implement one with data from GOTO when the time comes.

A.1 Source Detection & Reduction: December 2018 - January 2019

Continued improvements to the source detection algorithms as well as trying to perfect the data reduction process are needed. This is to ensure they are working as intended and no galaxies are being missed and automatically classified as stars, such as small circular galaxies.

A.2 Create a Training Set of Images: December 2018 - March 2019

As a CNN is a supervised learning algorithm a training set of pre-classified images is needed to train the classifier. In order to create the training set, each of the reduced sources (typically 1000-2000 sources per image) RA and DEC will be compared to the Galaxy Zoo database. All sources that appear in the Galaxy Zoo database will be known to be a certain type of galaxy. As the Galaxy Zoo is an extensive database, all remaining reduced sources in an image can be assumed to be false positive detections by the 'dumb' algorithm. If no galaxies are detected in a specific image (as they don't appear in Galaxy Zoo), it is assumed this area of sky is not covered by the Galaxy Zoo database, as each image should contain many galaxies, and the image will be ignored.

All reduced sources in an image can now be saved as a 'postage stamp' image into their respective groups, creating a training set of images of early and late galaxies as well as false positives. As each CCD has different deformations several classifiers will need to be made, each trained on images taken with that CCD. It may also be required, depending on the distortion of the images, to have multiple CNNs for different regions of an image, although on less distorted images hopefully only one CCN will be required. At this point there will be many more false positive images compared to early/late type galaxy images. In order to prevent overfitting equal numbers of images are preferred. Each galaxy image can be rotated and flipped creating a 'new' image until equal samples are created. This will also help the classifiers to become rotationally invariant.

A.3 CNN classifier with GOTO Data: March 2019 - June 2019

Once the training sets of images have been created they can be used to train the CNN. The data will be split into 3 groups to train, validate and test the classifier, as discussed in appendix B - section 1.2.1. Hopefully by this point in the semester we will have experience implementing CNN classifiers. However, there will undoubtedly be problems, potentially due to the sheer amount of computational power needed to train a CNN on so many images.

B Semester One Report

PHY480 1st Semester Report - Developing a Machine Learning Algorithm to Classify Detections in Large Astronomical Surveys.

150194163

December 13, 2018

Abstract

Due to advances in wide-field telescope technology, telescopes such as the Gravitational-wave Optical Transient Observer (GOTO) are capable of taking images with tens of thousands of sources present. Classifying all sources based upon their morphologies in such images has proved challenging. Many approaches have been tried, with some, such as the Galaxy Zoo project utilising citizen classification. However, these approaches still struggle to keep up with the ever increasing amount of incoming data, due to the need for human classification. One solution proposed in this report is to utilise a machine learning classifier to classify sources within images taken with GOTO, hoping to achieve accuracies comparable to that of human classifiers. A convolution neural network (CNN) was decided as the best classifier to use after a comprehensive review of previous literature. A source detection algorithm was carried out on images produced by GOTO to find the properties of all sources present within the image. An initial 'dumb' algorithm was then implemented to find all potential galaxy sources based upon their properties after correcting for distortions towards the edges of images, reducing the number of sources passed onto the CNN. Finally a plan is outlined to cross reference sources taken with GOTO and the Galaxy Zoo database to create a training set of images allowing for a CNN to be trained to classify sources into spiral and elliptical galaxies as well as false positive detections by the 'dumb' algorithm.

1 Literature Review

1.1 Introduction

The morphology of extrasolar objects, such as stars or galaxies, is a crucial metric that is among the first used by astronomers when classifying different objects throughout the Universe. Different objects appear very differently in the night sky with small objects such as stars appearing as point sources, whilst extended objects such as galaxies can often be observed to have shapes and structures. Early astronomers classified objects by eye or with small telescopes, being able to distinguish between stars and larger objects we now know to be nebulae and galaxies. As telescope technology has advanced it is now possible to observe more detailed morphological properties of larger extrasolar objects, allowing different objects to be classified based on similar properties. In order for an object to be classified based upon morphological properties it must be spatially resolvable. The vast majority of sources in the night sky are point sources (stars and very distant galaxies) and thus all look the same. Without the use of colour imaging or spectroscopy they can not be distinguished from one another. Larger objects such as galaxies and nebulae are generally spatially resolvable allowing morphological properties to be a useful metric in the subclassification of larger objects in the night sky.

With advances in wide-field telescope technology, large and detailed images of the night sky can be taken, each containing thousands of astronomical sources. The Gravitational wave Optical Transient Observatory(GOTO) [GOTO, 2018] is a set of robotic telescopes capable of imaging approximately 20 square degrees of the night sky at once. Set up to detect any light given off by gravitational wave events, the telescope can produce detailed images of the night sky containing tens of thousands of sources. Classifying all sources in these images based upon morphological properties is useful for astronomers as having large amounts of data about positions, type and relative numbers of different types of objects throughout the Universe aids study in the structure and composition of the Universe. It also allows large catalogues of similar types of objects to be produced, enhancing the study of individual types of object. With such vast numbers of sources it is not possible for astronomers to manually classify them, as was done in the past [deVaucouleurs et al, 1991] and new methods of classification are needed.

Whilst it is fairly easy to distinguish between point sources and more extended sources, further delineation between different types of objects is key to understanding the structure of the Universe. In particular, the classification of galaxies based upon their morphology has been an area of great scientific interest. The study started in 1925 when Edwin Hubble introduced his 'tuning fork' classification scheme [Hart et al, 1971]. With it he attempted to group the continuous shapes of galaxies into three main categories: elliptical, spiral, or barred spiral, as well as an irregular shaped category. This classification scheme proved to be successful, with variations still used to this date. More recent classification methods have built upon the 'tuning fork' approach (e.g Abraham et al [2000]), with others [Conselice, 2006] incorporating properties such as size, colour and mass of the galaxy into the classification scheme, allowing for a more rigorous classification of galaxies compared to that of just visual inspection. These classifications allow distinctions between early (elliptical) and late (spiral) type galaxies [Lintott, 2008] allowing for the study of the formation and evolution of galaxies over time. Furthermore, morphology can give us insight into the internal structures of galaxies as well as the effects of the dark matter present around them [Conselice, 2006].

Classification of different types of galaxies based on morphology has proven to be difficult to carry out on large scales. Over the past few years, various new methods of classifying large numbers of galaxy images based on their morphology have been tried. Galaxy Zoo is a project that explored crowd source classification, where over 150,000 volunteers have classified over 50million astronomical objects[1]. Galaxy Zoo has been vitally important in the large scale classification of astronomical data, with classifications found to be as reliable as classifications by professional astronomers [Lintott, 2008]. However, due to the enormously large amount of data now produced by modern telescopes, these projects are still unable to categorise sources at a quick enough rate to keep up with the incoming data.

It is clear there is a bottle neck due to the requirement of people having to manually inspect images of galaxies in order to classify them. To get over this bottle neck computers can be used to carry out such a process. Advancements in machine learning (ML) algorithms, especially convolutional neural networks (section 1.2.3), allow computers to classify images of galaxies at speeds far exceeding human capability, with accuracies of >90% [Aniyan & Thorat, 2017]. The rate at which galaxies can be classified using ML techniques is already faster that that of crowd source classification and will only increase due to optimisation of algorithms and advances in computational power. From previous methods explored ML algorithms will most likely hold the key to the mass classification of galaxies based on morphology, due to quick classification, high accuracy and scalability.

The aim of this project is to test if a ML algorithm is capable of classifying sources taken with the GOTO telescope into stars and galaxies, with accuracies comparable to that of manual classification. If this is achieved, we hope to investigate the possibility of using a ML algorithm to classify galaxies into the various classes described by Hubble's 'tuning fork' classification scheme.

1.2 Using Machine Learning to Classify Images

ML is essentially the ability for a computer to learn based upon previous outcomes to make accurate predictions about a new piece of data. The advantage of ML is that the machine can learn to solve problems that are often too difficult to hard code, opening up a brand new approach to solving complex mathematical problems. There are four main approaches of ML in regards to image classification, the details of which are outlined below.

The first distinction between approaches is in relation to how the ML algorithm learns, with two distinct approaches possible: supervised and unsupervised learning (section 1.2.1). The second distinction between approaches is in relation to how the ML algorithm gets information about the image. Feature based learning (section 1.2.2) involves extracting properties about an image, feeding them in raw data form to the ML algorithm. Conversely, other approaches such as neural networks can look at the image directly, analyse it using filters and then classify it; this approach is called non-feature based learning (section 1.2.3).

[1] *https://www.zooniverse.org/projects/zookeeper/galaxy-zoo/about/results*

1.2.1 Supervised and Unsupervised Machine Learning

There are two learning approaches utilised by ML algorithms: supervised and unsupervised learning [Donalek, 2011]. Supervised ML relies on a set of pre-classified data that is then split into 3 groups. The first of these groups is called the training data as this is what the algorithm uses to learn from. The algorithm is then validated on the second group of classified data to estimate errors and fine tune the algorithm. Finally, the trained algorithm is used to classify the third group of data (test data) that it has never seen before. The test data is used to assess the performance of the classifier. On the other hand, unsupervised ML does not use any pre-classified data. Instead, it takes a large sample of data and clusters it into groups of images with similar properties.

One of the pitfalls of supervised learning classifiers are their proneness to over fitting. The longer the ML algorithm is trained on test data (number of iterations of training) the better it becomes at classifying the test data. However, it has only learnt how to classify the test data and is now unable to classify new unseen data. Part of the validation data set is to find the ideal training time for the ML classifier. This is shown in figure 1 (credit to *elitedatascience.com*[2]) which shows that the error of the training set decreases as the number of training iterations increases, whereas the error of the validation set reaches a minimum before the error starts to increase again. The ideal point to stop training is when the error on classifying the validation data is at a minimum.

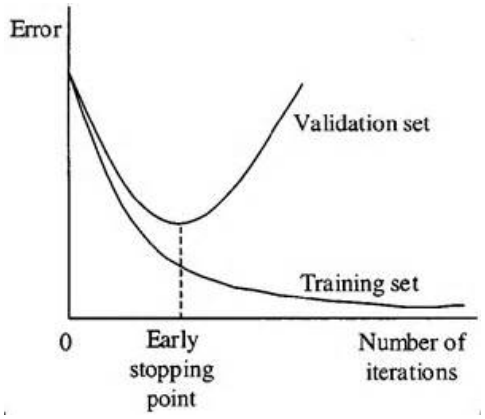


Figure 1: Shows how the error in classifying training and validation data sets change as the number of training iterations increase.

There are many advantages and disadvantages of using both learning methods. Whilst unsupervised learning doesn't require a large training set of pre-classified images, which is often hard to obtain, it is much harder to validate unsupervised learning methods. Several methods do exist [Halkidi et al, 2001], however are too in-depth and complicated to be implemented fully within the time frame of a 4th-year project, making it an unsuitable approach. In addition, unsupervised learning clusters images based upon similar properties, meaning unsupervised ML algorithms may group the data into undesired groups. This is not an issue with supervised learning as the training data is already classified into desired groups. For these reasons, only supervised methods are being considered in the remainder of the report[3].

1.2.2 Feature Based Learning Approaches

One way for a ML algorithm to classify an image is based upon its features. The features picked must describe the differences between types of images to be classified (e.g types of galaxy). Picking good features is the most important part of making feature learning successful [Blum et al, 1996]. These features are either picked by hand and extracted from the image, or another method of feature extraction is used. One common method of feature extraction is principal component analysis [Maitra et al, 2008], a multivariate statistical method that produces a linear combination of variables that summarise the image's data in a few parameters. Once features have been extracted they are often each given a weighting.

[2] *https://elitedatascience.com/overfitting-in-machine-learning*

[3] It should be noted that many of the methods outlined in section 1.2.2 and 1.2.3 have unsupervised variations, however are not mentioned

These weightings are varied as the ML algorithm learns from the training images, resulting in the best classification of new images based upon their features.

Several algorithms used to classify images involve 'plotting' each image's features into N-dimensional space, called a hyperplane, where N is the number of features. Each type of image should have features that group together. A ML algorithm can then find lines that separate these clumps most effectively, such that a new image can be classified depending on where it is placed on the hyperplane. Two examples of ML algorithms that utilise hyperplanes are K-nearest neighbour (KNN) [Peterson, 2009] and Support Vector Machine (SVM) [Hearst, 1998]. Whilst both KNN and a regular SVM are only able to classify linearly separable data, kernels can be applied within SVM to project the data into a higher dimension allowing for non-linear classification [Hearst, 1998].

Some approaches to classification assume each feature is equally weighted. One such example is a Naive Bayes classifier [Rish, 2001], utilising Bayes' statistical theorem to produce the probability of an image belonging to a certain class. Another approach is a random forest classifier, comprised of many decision tree classifiers [Safavian et al, 1991], produced by 'bootstrapping' the training data. A new image is passed through these bootstrapped decision tree classifiers, giving the probability of a new image belonging to a certain class as well as errors [Liaw, 2002].

1.2.3 Non Feature Based Learning Approaches

Whilst feature based algorithms are usually comparatively simple and can perform classifying tasks effectively, it is often difficult to pick out features that accurately explain images. Furthermore, a lot of data about the image is lost via feature extraction as the image is reduced to only a few key parameters. Whilst not a problem for detailed images, lower quality images can not be classified accurately by a few key parameters. To retain as much information about the image as possible non feature based ML algorithms can be used to classify images, where the raw image is the input.

- **Artificial Neural Network Image Classifiers:**
  Vaguely mimicking neurons in a brain, an artificial neural network (ANN) [Rojas, 1996] consists of several layers of neurons (that store a number) shown in figure 2 (credit to Kasture P[4]) as circles. Each neuron is 'connected' to all neurons in the previous and next layer. Commonly, each neuron in the input layer will correspond to the normalised pixel value of each pixel in the image (in a long list). However, these input neurons can also be features similar to those in feature based approaches. Each layer after the input layer is hidden except for the final output layer, which is comprised of one neuron for every class. The neuron with the highest value in the output layer corresponds to the class that the image belongs to.

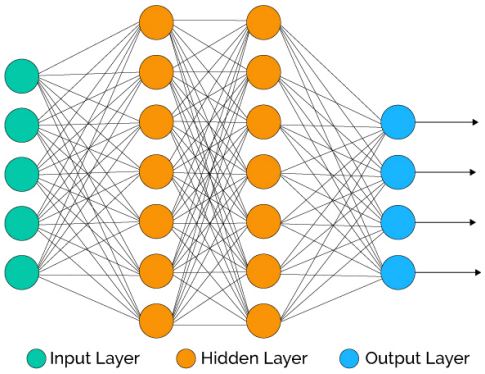


Figure 2: A schematic drawing of neurons (circles) in layers that comprise an artificial neural network.

Each neuron in the hidden layers is a function that takes the value of all neurons from the previous layer and outputs a number between zero and one. This is done by applying equation 1, where y is the output neuron value, x_i is the value of neuron i form the previous layer, W_i is the weighting of

[4] *http://blog.priyankakasture.me/neural-network-fundamentals/*

neuron i , b is the bias of the neuron and f is a Rectifier (normalising) function.

$$y = f(\sum_i W_i x_i + b) \tag{1}$$

Depending on these weights and biases, each layer can pick out different features from the image; some layers may pick out straight edges, while others may pick out circles or curves. ANNs that pick out lots of (often abstract) features from an image have lots of hidden layers and are known as deep neural networks or deep learning. Algorithms such as back propagation [Rojas, 1996, Chapter 7] allow the ANN to be 'trained' by slightly modifying the weights and biases of neurons as training images are compared to the outcome of the ANN. Training is often a long process as each connection has a weight and each neuron has a bias, resulting in hundreds of thousands of values that need to be fine tuned before the ANN can accurately classify new images.

- **Convolutional Neural Network Image Classifiers:**
  Arising within the last decade, convolutional neural networks (CNNs) are variations of ANNs [Kim et al, 2017]. CNNs are still comprised of layers of neurons, but they act differently in that they accept a 2D array of pixel values into the neural network as apposed to a long list. Furthermore, each neuron is now an n×n matrix (called a filter or kernel) that, when multiplied with each pixel value in the image, will find a certain feature (e.g edges), producing an array of pixel values called a feature map. A Rectifier (normalisation) layer is then applied to the feature map for optimised effect. Many convolutional layers with rectifier layers are applied to the image, producing many rectified feature maps, each picking out a certain feature.

  Pooling layers are then applied to reduce the dimensions of the feature maps, now essentially having one bright pixel if a certain feature is present. Once the data is reduced the pooled layers are turned into a list of pixel data, which is then classified using a standard deep neural network. Because of these convolutional layers picking out features from an image, CNNs are the dominant method used in automatic image classification.

1.3 Previous uses in Astronomy

Due to advancements in algorithms and computational power, ML is a very active area of research especially with regard to image recognition and classification, with CNNs only being developed within the last decade. Within astronomy the vast majority of approaches utilised so far involve supervised learning, with relatively few attempting unsupervised learning. Below is an outline of previous research relating to classification of galactic images.

1.3.1 Unsupervised Learning Approaches

Schutter et al [2015] & Hocking et al [2017] used unsupervised ML approaches to classify galactic images based upon morphology, showing that unsupervised learning algorithms are capable of distinguishing between early and late type galaxies with no pre-directed training. Both methods were very involved, requiring an in-depth understanding of ML beyond what would be capable of being reproduced within a 4th-year project. Schutter et al [2015] also noted that the use of colour images greatly enhanced the accuracy of the algorithm, suggesting supervised learning algorithms may be better at classifying single band images, such as the images taken with GOTO.

1.3.2 Feature Based Learning Approaches

Many feature based learning approaches have been used to classify images, with many studies comparing the various methods. Calleja [2004] & Gauthier et al [2016] used principal component analysis to extract between 8 and 125 components from images of galaxies. When comparing all major feature based methods both found a random forest classifier was the most accurate at classifying images, finding 92% accuracy with 2 groups [Calleja, 2004] , 65% accuracy with 5 groups [Gauthier et al, 2016] and 50% accuracy with 7 groups [Calleja, 2004]. Huertas-Company et al [2008] outlined hand picked features to be used in the classification of galaxies including concentration (ratio of light within inner aperture), asymmetry, smoothness and gini coefficient (based upon Lorentz curve) of the source. The data was found to be non-linearly separable, so a SVM with a RBF kernel was used to separate the data. Later

on Huertas-Company et al [2010] used the previously outlined framework trained on the Sloan Digital Sky Surveys (SDSS) to classify 700,000 high red shift galaxies into 4 types, finding good agreement with results from Galaxy Zoo.

1.3.3 Deep Neural Networks

Within the last few years, deep learning has become the dominant force within ML image classification. Deep learning allows for multiple layers of abstract and detailed feature analysis, achieving accuracies much greater than that of other shallow learning approaches [LeCun et al, 2015]. Regular deep learning ANNs have been shown to match and often out-perform feature based approaches, with Bazell & Aha [2001] finding than an ANN out-performed Naive Bayes and decision tree when classifying spiral and elliptical galaxies. However, they noted that the Naive Bayes classifier was the easiest to implement. Banerji et al [2010] used an ANN with 12 hand-picked input parameters trained on 75,000 images to classify over 1,000,000 images from the SDSS into early, late and point sources, achieving greater than 90% accuracy for all 3 classes.

Due to their ability to extract detailed features from images, CNNs have unmatched accuracies within image classification. Using high detailed images, Kim & Brunner [2016] achieved 99% accuracy differentiating between stars and galaxies with a CNN, however did mention there may be over fitting issues. Dieleman et al [2015] were able to work out properties of the 65,000 brightest galaxies within the SDSS achieving almost 99% accuracy for images with high consensus within the Galaxy Zoo project. Whilst these accuracies are impressive, they are only so high as the image quality is good, or clear classifications can be made(star/galaxy comparison). CNNs can still achieve high accuracies when classifying more low detailed images. Aniyan & Thorat [2017] classified radio images (single band) of extended sources into 3 groups with between 75% and 95% accuracy. The CNN was trained on only 200 images which were then rotated and flipped to increase the training set as well as to help the classifier to become rotationally invariant. The classifier was able to produce the probably of a source being a certain type of object within 0.17 seconds, much faster than human classification. Abraham et al [2018] achieved 94% accuracy detecting bar structures within 9346 galaxy images from the SDSS, noting the CNN was capable of being scaled to handle large volumes of data. Finally Khalifa et al [2017] proposed a CNN architecture with 8 hidden layers that classifies galaxy images into early, late and irregular. Under preliminary tests, trained on 1,356 images, it was able to classify with over 97% accuracy.

An additional feature of CNNs that has been utilised by astronomers is the ability to transfer learning from already well established classifiers. When there is a limited number of training sources, CNNs can be trained to detect features from millions of images of everyday objects. This greatly improves their ability to pick out features from the training sources, giving more accurate classifications [Pan, 2010]. Ackermann et al [2018] found that utilising transfer learning from the IMAGENET[5] database increased the accuracy of classifying images of galaxy mergers by 15%, also noting this method was significantly better than any other method for detecting galaxy mergers.

There is no best way to approach a ML task. If good features can be picked from an image, feature based algorithms are often comparatively quicker, simpler and easier to understand than deep learning approaches that are often complex black box approaches. However, deep learning algorithms, especially CNNs, have shown unparalleled accuracies with regard to image classification due to their ability to pick out detailed and abstract features from an image. CNNs also bypass feature extraction (the most complicated and intricate step involved in feature based ML) often making them easier to implement than feature based approaches.

2 Progress on Project

The main goal of this first semester was to review literature about ML techniques and to decide what approach would be most suited for this classification task. The review of major techniques has been covered in section 1. Out of the ML approaches discussed the most effective and accurate at image classification is the CNN. Thanks to transfer learning and many ML Phyton libraries, such as TensorFlow[6] and scikit-learn[7], they are just as easy to implement as any other ML algorithms, with the added

[5] *http://www.image-net.org/*
[6] *https://www.tensorflow.org/*
[7] *https://scikit-learn.org/stable/*

advantage of not having to extract features from the images. For these reasons a CNN approach will most likely be taken to classify sources within GOTO images.

The following section of the report outlines the steps taken to transform the data from a raw form to one that can be classified using a ML algorithm; more specifically, a CNN. Section 2.1 discusses the raw data, section 2.2 discusses the source detection algorithms used to find sources, section 2.3 discusses reducing the number of sources entering the ML algorithm and section 2.4 discusses preparing images to be fed into a CNN.

2.1 The Data

The data is in the form of fits file images taken by the robotic telescope at the GOTO observatory on La Palma. Each image is around 70Mb and contains ∼6000-12000+ sources. The telescope has 4 different CCD chips and although errors have been amended now, there were problems causing CCD1 and 3 to produce defective images with large distortions to sources. Because of this, images taken with these CCDs have not been used. Furthermore, some of the original pictures taken with CCD2 have moderate deformations; an image taken with CCD2 is shown in figure 3 with distortion towards the right on the image (highlighted by the red box), stretching all the sources. CCD4 produces the best quality images with only small amounts of deformation, making CCD4 the easiest to analyse as little correction was required.

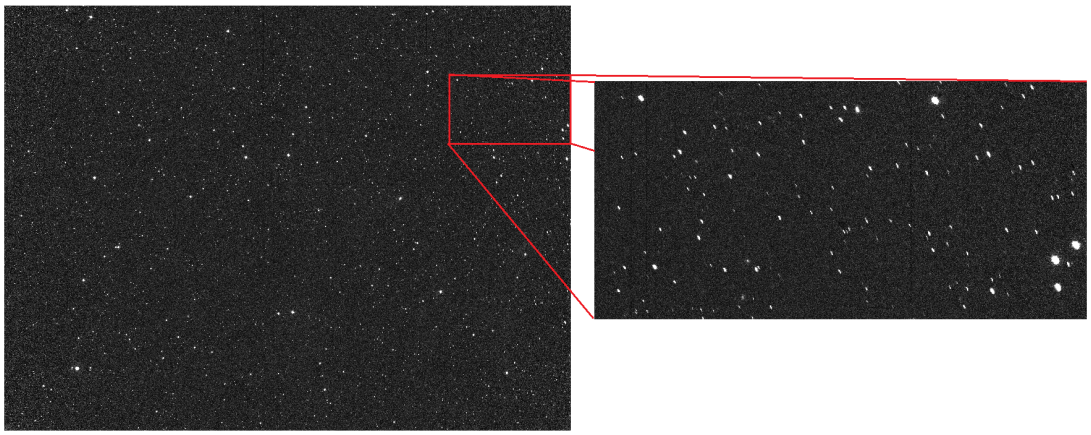


Figure 3: An image taken with the GOTO telescope using CCD2. There is clear distortion towards the right side of the image, while the left side remains normal.

The Astropy Python package was used to process and manipulate the fits images. This involved first downloading the Astropy package, which required a C compiler, and then learning how to use the package. Information about the image, such as right ascension and declination, was stored in the image headers and had to be read using the Astropy package. SAOImageDS9[8] was used to easily view images and sources and to check that various aspects of the code were working as intended. This was primarily done via the creation of regions files, allowing detected sources to be highlighted, as shown later in section 2.3.1 (figure 8 & 9).

2.2 Source Detection

To produce an image set compatible with a ML algorithm, each source in the large image had to be identified and cut out as its own 'postage stamp' image. These images could then be fed into the ML algorithm to be classified. To find all individual sources in the image several source detection Python packages were tested. Originally `DAOStarFinder`[9] and `find_peaks`[10] by Photutils were used. However, under visual inspection it was clear these two methods struggled to pick up all sources accurately, often

[8] *http://ds9.si.edu/site/Home.html*
[9] *https://photutils.readthedocs.io/en/stable/api/photutils.DAOStarFinder.html*
[10] *https://photutils.readthedocs.io/en/stable/api/photutils.detection.find_peaks.html*

missing fairly obvious objects, while detecting very faint objects which may have been background noise. Finally `detect_sources`[11] by Photutils was used to detect sources in the images. It was chosen for three main reasons:

1. It had the most detailed input parameters, allowing the most flexible and accurate source detection from any method tried.

2. It has a second complimentary function called `deblend_sources`[12] which allows sources in an image to be deblended from one another; this greatly helped reduce multiple detections of larger sources as well as allowing for nearby objects to be picked up as two separate sources. Examples of deblending overlapping sources are shown in figure 4. As can be seen, sources that are often obscuring each other can be deblended with fairly good accuracy.

3. Properties about the deblended sources could be calculated using Photutils `source_properties`[13] function, returning over 50 properties about each source, including semimajor/minor axis, orientations and size of the source. These properties are the ones used to create the ellipses around sources shown in figure 4.

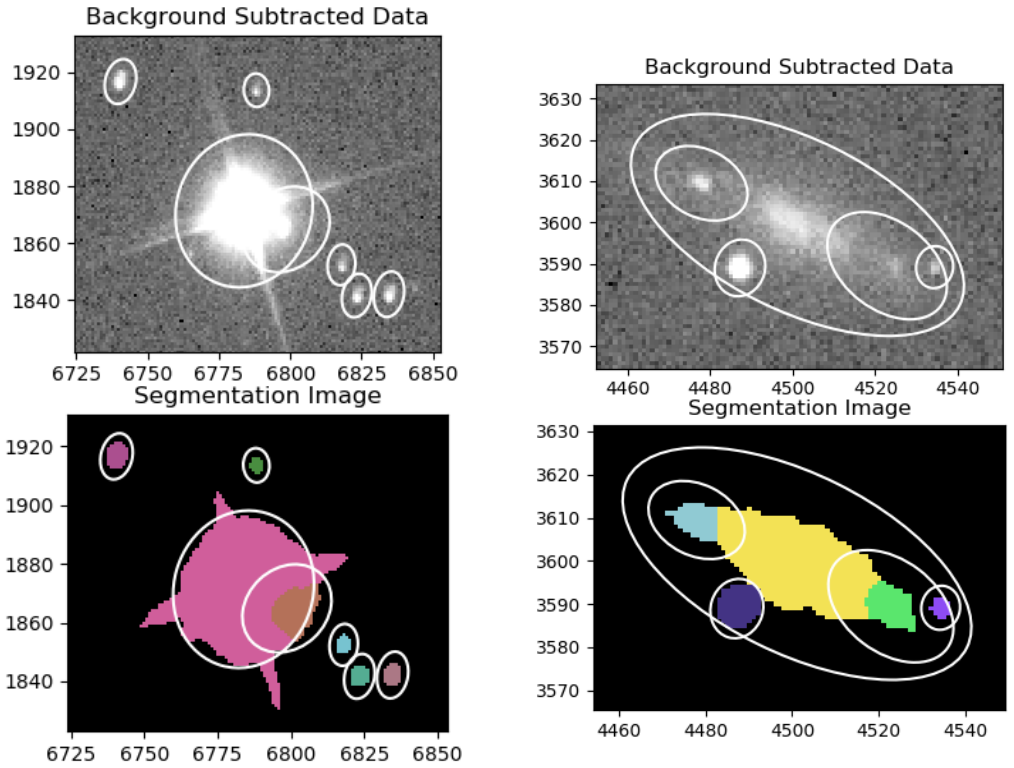


Figure 4: Example of Source deblending using Photutils `deblend_sources`, with the top images showing the background subtracted data and the lower images showing the deblended sources in different colours.

It should be noted that the detection algorithm used is not perfect and still misses some faint sources and occasionally double detects very elongated sources such as the ones found on the right of figure 3. Furthermore, for most of the semester the `deblend_sources` code was returning errors for unknown reasons and was unable to deblend sources correctly. Luckily, others had similar problems and had posted various solutions online. After making minor changes to the source code of `deblend_sources` it became

[11] *https://photutils.readthedocs.io/en/stable/api/photutils.detect_sources.html*
[12] *https://photutils.readthedocs.io/en/stable/api/photutils.deblend_sources.html*
[13] *https://photutils.readthedocs.io/en/stable/api/photutils.segmentation.source_properties.html*

a much more reliable way of accurately detecting sources.

The GOTO data processing pipeline already has a built-in source detection routine, returning various properties of detected sources. The results of this source detection seem to be comparable to that of `detect_sources`, `deblend_sources` & `source_properties`. At the moment, we are unsure which source detection routine we wish to pursue. The decision will depend on how accurately the various different parameters returned by both routines explain the sources within an image.

2.3 Reducing the Data

As ML is a computationally demanding process, it is important to attempt to minimise the number of sources that are being classified by a ML algorithm. As the vast majority of sources in an image are stars, running all sources in an image through a ML algorithm to find a few galaxies would not only take an unnecessarily long time but could potentially result in over fitting as ML algorithms operate best with ∼50/50 classification training samples. A 'dumb' algorithm was proposed to quickly detect all sources that are definitely stars within an image and classify them as such. This would leave behind ∼10-20% of sources that are not clear stars, but are more complicated objects such as galaxies or distorted objects causing false detections by the 'dumb' algorithm. All sources detected by the 'dumb' algorithm can then be classified using a CNN classifier to find the galaxies, and remove false detections.

Properties extracted from the `source_properties` function can be used to quickly tell if a source is a star or something more compacted. Stars appear as small point sources, so have small pixel areas and low ellipticity, whilst galaxies generally appear much larger and may have higher ellipticity. Stars also appear sharper, with galaxies fading more gradually. As shown in section 2.1 some images produced by GOTO are distorted, with objects on the outskirts of the images appearing more elliptical than they actually are in the night sky. This made removing definite stars from the image much more challenging.

2.3.1 The 'Dumb' Algorithm

To reduce the data two parameters were considered: the ellipticity and pixel area of each source. In non distorted areas of images the properties from the source detection algorithm, namely `source_properties`, accurately describe the sources and could be used to remove definite stars with no corrections needed. However, objects on the outskirts of the images appear to be more elliptical than they actually are in the night sky. The extent of this distortion is shown in figure 5, where the ellipticity(e) of sources in figure 3 are plotted against their position. As can be seen, sources on the right side of the image are very elliptical (0.6<e<1), whilst on the left side of the image they are much more circular.

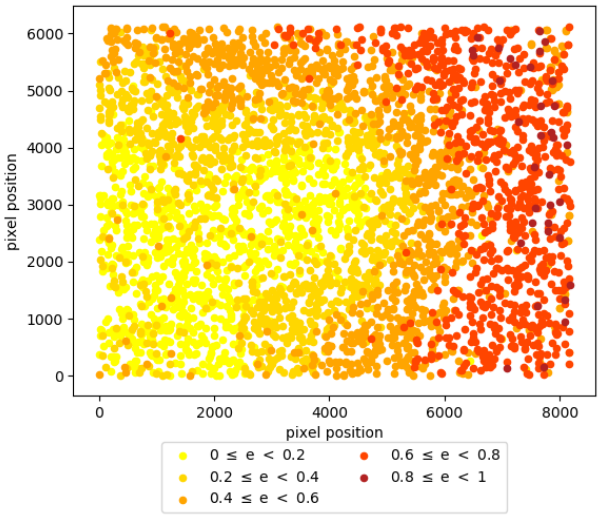


Figure 5: A plot of the ellipticity(e) of each source in figure 3 against their position.

To account for the deformations, allowing a 'dumb' algorithm to distinguish definite stars in distorted

regions of images, a 2D polynomial equation was fit to the ellipticity and orientation of each of the sources to find the theoretical average ellipticity and orientation at each position in the image. This was done using a function fitter called `Polynomial2D`[14] by Astropy. Each image analysed was fit with its own set of polynomial equations, such as the one shown in figure 6, a fit to the ellipticity in figure 5.

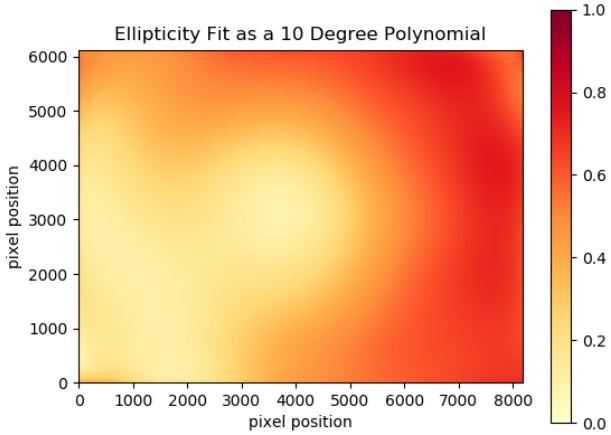


Figure 6: A plot of a 2D polynomial equation fit to ellipticity of sources shown in figure 5.

To remove definite stars in a computationally efficient manner, the pixel area of each source was considered first. After fitting a function to the pixel area of each source, an example shown in figure 7, it was decided that deformations in the images did not greatly impact the pixel area of sources across the image. Therefore, the average pixel area of sources across the image can be treated as ∼constant. It was first assumed that all small objects, below the mean pixel area, were definite stars or small pixel defects and were deemed not galaxies. Secondly, it was assumed that all large sources, above 200 pixels^2, were potential galaxies.

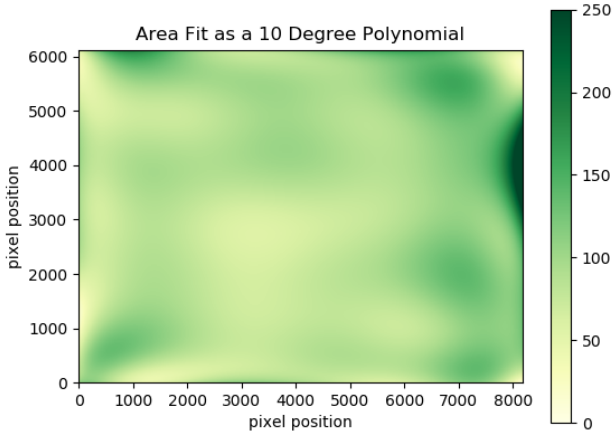


Figure 7: A plot of a 2D polynomial equation fit to pixel areas(pixels^2) of sources shown in figure 3

The ellipticity and, if needed, the orientation of the remaining sources was then considered to calculate

[14] *http://docs.astropy.org/en/stable/api/astropy.modeling.polynomial.Polynomial2D.html*

the 'normalised ellipticity' of each source, a number that describes how elliptical a source is taking into account the deformation in the image. Each source was considered individually, if the modelled ellipticity at its pixel location was >0.25 it was considered to be in a high deformation region, otherwise it was deemed to be in a low deformation region. In regions of low deformation the orientation of sources is essentially random. The orientation is therefore not factored into the normalised ellipticity calculation, which is just the ellipticity of the source divided by the modelled ellipticity at that pixel location. In regions of high deformation, all sources are highly elliptical and all have the same orientation. The actual orientation and ellipticity of a source in such a region is compared to the modelled orientation and ellipticity at that point and is used to calculate the normalised ellipticity taking into account its orientation. The normalised ellipticity of sources in an image clusters around 1. If a source had a normalised ellipticity of less than 0.75 or more than 1.25 it was considered a potential galaxy.

Unfortunately, there are still problems with the 'dumb' algorithm. The current solution still ignores small circular galaxies. Whilst uncommon in the images, there does need to be a solution where such galaxies can be detected. Furthermore, due to the way the `source_properties` function outputs the orientation of sources there is a sudden jump from $-\pi/2$ to $+\pi/2$ in some parts of the images. The polynomial equation fit to the orientation struggles to account the sudden jump as it is only designed for smooth transitions in data, meaning some small areas of the images don't have accurately modelled orientations.

An example of the 'dumb' algorithm detecting potential galaxies in an undistorted region of an image is shown in figure 8. As can be seen the detection algorithm is operating fairly well, detecting large and elliptical sources. Figure 9 shows a bad region of an image. The 'dumb' algorithm is still capable of picking out sources that are potential galaxies and is currently capable of reducing the number of sources entering the ML algorithm by >80%.

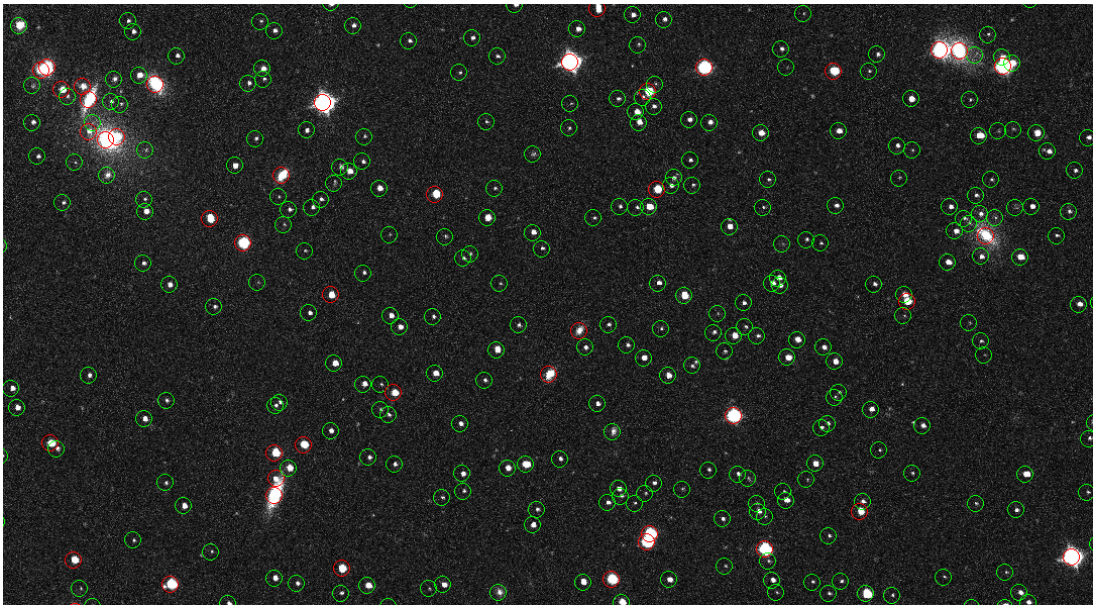


Figure 8: An undistorted region of sky imaged with GOTO's CCD2. Sources with a green circle around them are deemed definite stars by the 'dumb' algorithm, whilst sources with red circles around them are potential galaxy detections.

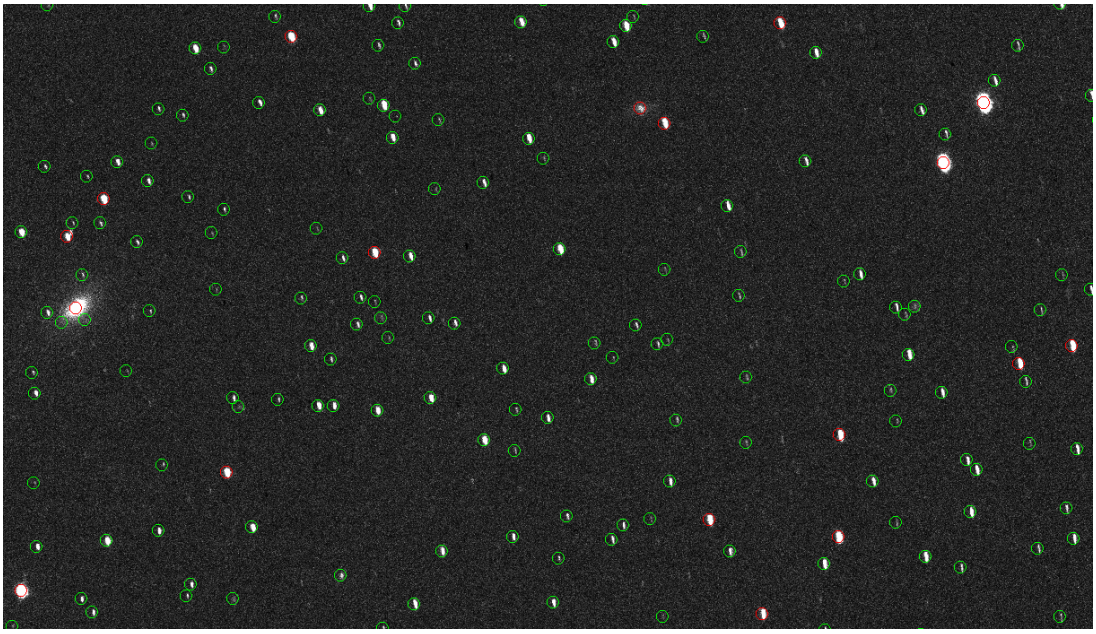


Figure 9: A distorted region of sky imaged with GOTO's CCD2. Sources with a green circle around them are deemed definite stars by the 'dumb' algorithm, whilst sources with red circles around them are potential galaxy detections.

The 2D polynomial equation used to fit the data could have varying degrees. Figure 11 in the appendix shows that beyond a polynomial equation fit of ∼10 degrees, there is not much change in fit as the number of degrees is increased. However, as the number of degrees increases the speed at which the code ran increased significantly. As the 'dumb' algorithm is only a 'quick and dirty' way of classifying definite stars the small increased levels of accuracy do not outweigh the quicker run time of lower degree polynomials of ~ 10.

2.4 Cutting out Sources from the Images

In order to be classified by a CNN each potential source has been made into a 50x50 (although this may be reduced to speed up classification) pixel 'postage stamp' image with the object to be classified at the centre of the image. To cut out the images `Cutout2D`[15] from the Astropy package was used. At this moment in time, it is still unclear if having other sources present in the 'postage stamp' images affects results of the CNN. If this is the case, all pixels except for ones that comprise the source will be set to 0.

As 50x50 images were needed, any source closer than 30 pixels to the edge of the image was discarded as a complete cutout could not be made. Luckily, this is not a problem as images taken with GOTO have an overlap, meaning sources close to the edge of one image are present in another image, where they are not so close to the edge. Therefore removing sources close to the edge of images has not resulted in any sources being ignored. Two examples of these 'postage stamp' images are shown in in figure 10. The image on the left is fairly clearly a galaxy, however other images such as on the one on the right are a little more complicated and often have more than one source present in the image.

While cutting out the image `WCS`[16] (World Coordinate System) part of the Astropy package read information from the header of the image and took the pixel value of the source to calculate its RA and Dec. This RA and Dec was then stored in the header of each of the cutout images, so each cutout source could be placed back onto the image or read into SAOImageDS9 via the creation of a region file. This information will be vital in creating a training set for the ML algorithm, discussed in section 3.2.

[15] *http://docs.astropy.org/en/stable/api/astropy.nddata.utils.Cutout2D.html*
[16] *http://docs.astropy.org/en/stable/api/astropy.wcs.WCS.html*

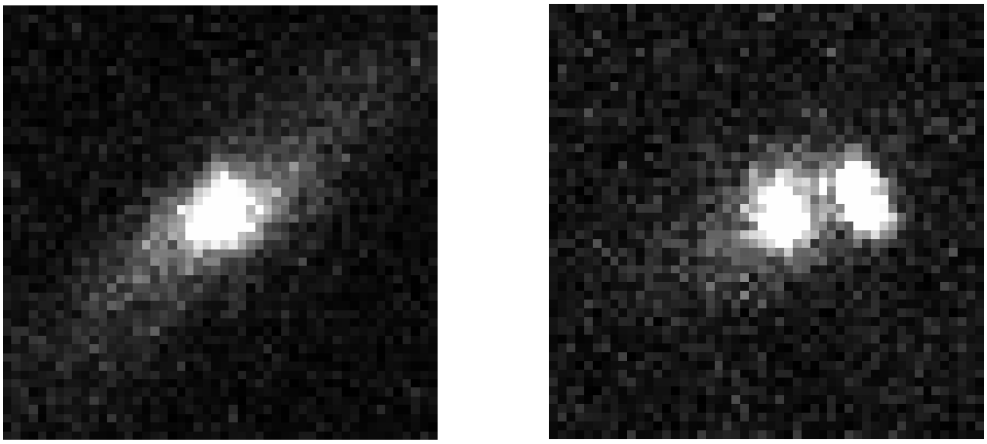


Figure 10: Two 50x50 pixel 'postage stamp' images cutout from a large GOTO image of sources detected as potential galaxies by the initial 'dumb' algorithm.

3 Project Plan

This section outlines the intended objectives over the course of the second semester. All dates are estimates. We intended to practice implementing CNNs throughout the whole semester, hopefully making it easier to implement one with data from GOTO when the time comes.

3.1 Source Detection & Reduction: December 2018 - January 2019

Continued improvements to the source detection algorithms as well as trying to perfect the data reduction process are needed. This is to ensure they are working as intended and no galaxies are being missed and automatically classified as stars, such as small circular galaxies.

3.2 Create a Training Set of Images: December 2018 - March 2019

As a CNN is a supervised learning algorithm a training set of pre-classified images is needed to train the classifier. In order to create the training set, each of the reduced sources (typically 1000-2000 sources per image) right ascension and declination will be compared to the Galaxy Zoo database. All sources that appear in the Galaxy Zoo database will be known to be a certain type of galaxy. As the Galaxy Zoo is an extensive database, all remaining reduced sources in an image can be assumed to be false positive detections by the 'dumb' algorithm. If no galaxies are detected in a specific image (as they don't appear in Galaxy Zoo), it is assumed this area of sky is not covered by the Galaxy Zoo database, as each image should contain many galaxies, and the image will be ignored.

All reduced sources in an image can now be saved as a 'postage stamp' image into their respective groups, creating a training set of images of early and late galaxies as well as false positives. As each CCD has different deformations several classifiers will need to be made, each trained on images taken with that CCD. It may also be required, depending on the distortion of the images, to have multiple CNN for different regions of an image, although on less distorted images hopefully only one CCN will be required. At this point there will be many more false positive images compared to early/late type galaxy images. In order to prevent over fitting equal numbers of images are preferred. Each galaxy image can be rotated and flipped creating a 'new' image until equal samples are created. This will also help the classifiers to become rotationally invariant.

3.3 CNN classifier with GOTO Data: March 2019 - June 2019

Once the training sets of images have been created they can be used to train the CNN. The data will be split into 3 groups to train, validate and test the classifier, as discussed in section 1.2.1. Hopefully by this point in the semester we will have experience implementing CNN classifiers. However, there will undoubtedly be problems, potentially due to the sheer amount of computational power needed to train a CNN on so many images.

References

Abraham R. G. & Merrifield M. R., 2000, "EXPLORATIONS IN HUBBLE SPACE : A QUANTITATIVE TUNING FORK", THE ASTRONOMICAL JOURNAL, 120:2835-2842

Abraham S. et al, 2018, "Detection of Bars in Galaxies using a Deep Convolutional Neural Network" arXiv:1711.04573v2

Ackermann S. et al, 2018, "Using transfer learning to detect galaxy mergers", arXiv:1805.10289v2

Aniyan A. K. & Thorat K., 2017, "CLASSIFYING RADIO GALAXIES WITH CONVOLUTIONAL NEURAL NETWORK", arXiv:1705.03413v1

Banerji M. et al, 2010, "Galaxy Zoo: Reproducing Galaxy Morphologies via Machine Learning", arXiv:0908.2033v2

Bazell D. & Aha D. W., 2001, "ENSEMBLES OF CLASSIFIERS FOR MORPHOLOGICAL GALAXY CLASSIFICATION", Apj 548 : 219-223

Blum A. L. & Langley P., 1996, "Selection of relevant features and examples in machine learning", Amficial Intelligence 97, p235-271

Conselice C. J., 2006, "The fundamental properties of galaxies and a new galaxy classification system", Royal Astronomical Society, Vol:373, p1389-1408

Calleja J. & Fuentes O., 2004 "Automated Classification of Galaxy Images", LNCS, volume 3215

Dieleman S. et al, 2015, "Rotation-invariant convolutional neural networks for galaxy morphology prediction" Royal Astronomical Society, Volume 450, Issue 2

Donalek C., 2011 "Supervised and Unsupervised Learning", Caltech University

Gauthier A. et al, 2016, "Galaxy Morphology Classification", Stanford University

GOTO "The Gravitational-Wave Optical Transient Observer" 2018, web page: *https://goto-observatory.org/*

Halkidi M. et al, 2001, "On Clustering Validation Techniques", Journal of Intelligent Information Systems, 17:2/3, 107–145

Hart R. & Berendzen R., 1971, "HUBBLE'S CLASSIFICATION OF NON-GALACTIC NEBULAE, 1922-1926", Boston University

Hearst M. A., 1998, "Support vector machines" University of California, Berkeley

Hocking A. et al, 2017 "An automatic taxonomy of galaxy morphology using unsupervised machine learning", arXiv:1709.05834v1

Huertas-Company M. et al, "Revisiting the Hubble sequence in the SDSS DR7 spectroscopic sample: a publicly available Bayesian automated classification", AA 525, A157

Huertas-Company M. et al, "A robust morphological classification of high-redshift galaxies using support vector machines on seeing limited images" , arXiv:0709.1359v1

Kgalifa N. et al, 2017, "Deep Galaxy: Classification of Galaxies based on Deep Convolutional Neural Networks", arXiv:1709.02245

Kim D. & Lee H., 2017, "Image Manipulation Detection using Convolutional Neural Network", ISSN: 0973-4562 Volume 12, Number 21

Kim E. J. & Brunner R. J., 2016, "Star-galaxy Classification Using Deep Convolutional Neural Networks" arXiv:1608.04369v2

LeCun Y. et al, 2015, "Deep learning" Nature volume 521, pages 436–444

Liaw A. & Wiener M., 2002, "Classication and Regression by randomForest" ISSN: 1609-3631

Lintott C. J., 2008 “Galaxy Zoo: Morphologies derived from visual inspection of galaxies from the Sloan Digital Sky Survey” arXiv:0804.4483v1

Maitra S. & Yan .J, 2008 “Principle Component Analysis and Partial Least Squares: Two Dimension Reduction Techniques for Regression”, Applying Multivariate Statistical Models, p79-90

Pan S. J. & Yang Q., 2010, “A Survey on Transfer Learning”, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 22, NO. 10

Peterson L. E., 2009 “K-nearest neighbor” webpage: *http://scholarpedia.org/article/K-nearest_neighbor*

Rish I. “An empirical study of the naive Bayes classier” T.J. Watson Research Center

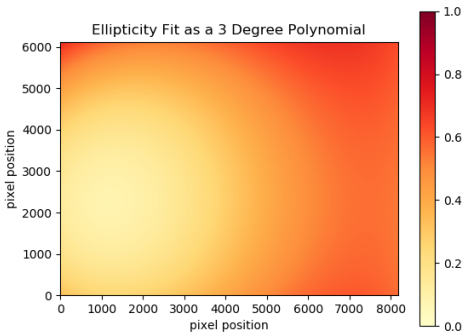
Rojas R., 1996, “Neural Networks”, Springer

Safavian S. R. & Landgrebe D., 1991, “A Survey of Decision Wee Classifier Methodology”, IEEE Transactions on Systems, Man, and Cybernetics, vol. 21, no. 3

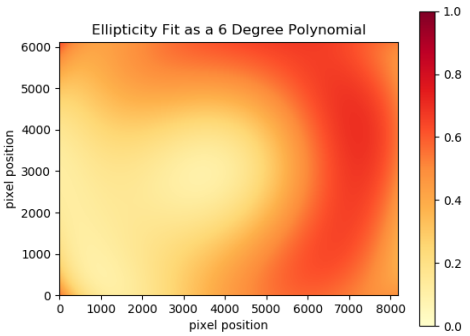
Schutter A. & Shamir L., 2015, “Galaxy morphology — An unsupervised machine learning approach”, Astronomy and Computing 12 (2015) 60–66

deVaucouleurs G. et al, 1991, “Third Reference Catalog of Bright Galaxies”

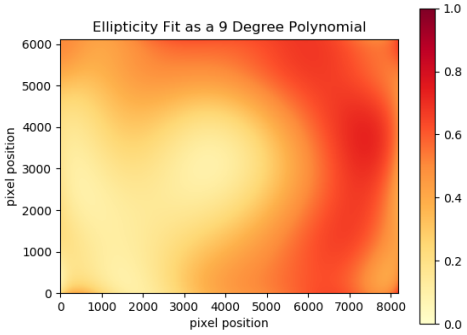
Appendices



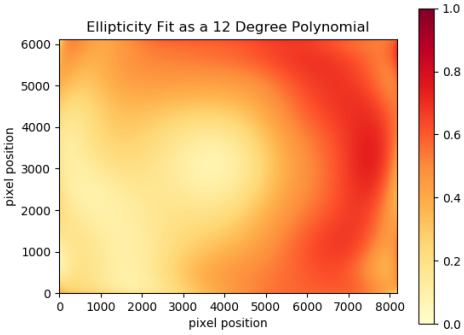
(a) 3 degree polynomial equation fit.



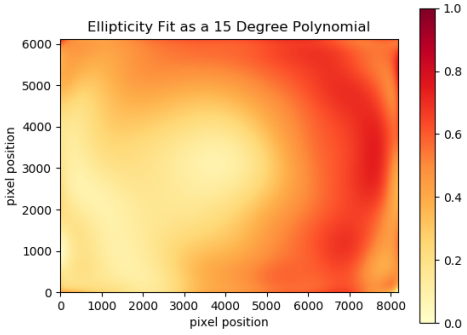
(b) 6 degree polynomial equation fit.



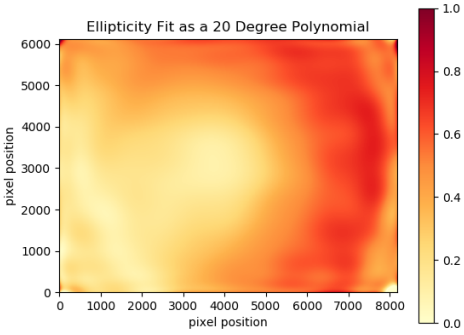
(c) 9 degree polynomial equation fit.



(d) 12 degree polynomial equation fit.



(e) 15 degree polynomial equation fit.



(f) 20 degree polynomial equation fit.

Figure 11: Plots of 2D polynomial equations fitted to ellipticity shown in figure 5, each with a different degree.