Developing a Convolutional Neural Network to Classify Detections in Large Astronomical Surveys

150194303

June 8, 2019

Abstract

Morphological classifications provide one of the most fundamental methods of categorising astronomical objects. The advent of improved telescope and computational processes have required novel techniques to be developed to classify objects at a sufficient rate. Machine Learning has provided the most viable solution, with multiple examples of successful classification using deep learning approaches in an astronomical context. Here, a Convolutional Neural Network was applied to GOTO images to classify extended/non-extended and featured/non-featured sources. Sources were detected and pre-labelled using simple Python methods, with cutouts created of individual sources. Due to the intrinsically lower number of extended objects, the dataset was augmented through rotations to provide balanced training sets. The images were also normalised which provided a dramatic increase in training accuracy. Accuracies across the classes were consistently $> 90\%$, which was comparable to manual classification, only significantly faster. When tested on unseen data the model was similarly successful, suggesting it had not overfit. Attempts were also made to improve the quality of the input data, with limited success. The sole use of accuracy as a metric presented problems when testing on unbalanced data due to a low precision. Future work should be focused around testing on larger datasets, with a more appropriate performance metric.

1 Introduction

One of the purest forms of astronomical classification is through the shape and structure of objects, known as its morphology. Morphological classifications provide clear and simple distinguishing features that allow for classification of objects into broad categories. For example, galaxies and nebulae appear as extended sources, while stars appear as point-like (non-extended) sources. With this simple distinction it is possible to distinguish between the majority of objects in the night sky.

Technological advances within astronomy, such as the advances in telescope optics and computational power throughout the 20th century has allowed for significantly more sensitive and wide-area observations. The increase in sensitivity has allowed for significantly more objects to be detected. The increased number of detections has put strain on the traditional methods of classification through small teams. Wide area surveys only amplify these concerns. Surveys such as the Gravitational-wave Optical Transient Observatory (GOTO) (GOTO 2018) and the Sloan Digital Sky Survey (SDSS) (SDSS 2018) cover vast regions of the sky, with SDSS detecting over 1.2 billion objects across around a third of the night sky (Gauthier 2016). This vast increase in available data has ushered in the 'petabyte regime' of astronomy (Hocking et al. 2017). As astronomy shifts from a traditionally data-poor, to a data-intensive discipline (Howard 2015), novel solutions must be proposed in order to allow for classification to match the rates at which data is acquired.

One such novel solution was the Galaxy Zoo (GZ) project, that attempted to capitalise on the power of crowd sourcing to categorise large amounts of objects. To date, over 60 million objects have been classified by over 100,000 unique users, making it the largest astronomical collaboration in history (Lintott et al. 2008). The catalogues produced by GZ contain up to 300,000 galaxies which is over an order of magnitude greater than the previous largest galaxy survey (Willett et al. 2013). The accuracies reported across GZ are in $> 90\%$ agreement with professional astronomers (Lintott et al. 2008). This represents a landmark for the study of morphological classification, as GZ has proven that the general public can produce comparable classifications to professionals, while being significantly faster due to the sheer number of potential users. The classification speed is so much faster that GZ reported ~ 40 million classifications in 165 days (Lintott et al. 2008), compared to the expected $\sim 10,000$ classifications in a year from a professional astronomer (Calleja and Fuentes 2004). However, despite this vast speed increase astronomical telescopes have already achieved data rates of > 10TB per day (Aniyan and Thorat 2017), so objects would not be classified at a sufficient rate using the approach outlined by GZ.

A promising solution that has emerged is Machine Learning (ML). In general, ML gives computers the ability to learn and recognise patterns without being explicitly programmed to do so (Eyono 2017). ML algorithms can be divided into two broad categories: supervised, and unsupervised learning. Supervised learning uses labelled input images and targets a known output, while unsupervised techniques requires no pre-labelling and has no previously known classes. While there have been some novel solutions employing unsupervised techniques (Hocking et al. 2017), currently the majority of literature associated with astronomy employs supervised learning techniques. Supervised learning generally requires the data to be split into a training set and test set. The training set is used as inputs into a ML algorithm which learns to recognise the features within the data and develop a trained model. The test set is then used to evaluate the model, and is desirable to be unseen by the training portion of the model. When applied to image classification, ML allows for automatic, probabilistic classification with speeds significantly faster than human identification. Typically, ML is then divided again into two further categories: shallow and deep learning. Shallow learning requires the use of 'features' that are representative of the physical properties of the system (Aniyan and Thorat 2017). Deep learning however, learns directly from the raw data (Aniyan and Thorat 2017), so does not require any potentially subjective feature selection. ML is employed in numerous fields, including astronomy, and will likely become a staple due to its high speed, accuracy and scalability as more advanced computing techniques are developed.

While there are several examples of multiple methods of performing ML (see Appendix B), it is arguable that the deep learning artificial neural networks (ANNs) are the most successful. The use of ANNs has produced accuracies that are significantly greater than the shallow learning counterparts in the field of image classification (LeCun et al. 2015, Aniyan and Thorat 2017), with the convolutional neural network (ConvNet) sub-species producing record breaking accuracies (Kim and Brunner 2016). ANNs exploit the architecture of the human brain, having an array of interconnected 'neurons' that are able to approximate complex functions (LeCun et al. 2015). The ability of ANNs to learn directly from the raw data makes them advantageous for astronomical classification, as it removes the need for any potentially ambiguous or subjective physical features. These considerations make ANNs and ConvNets very useful tools for the classification of astronomical images such as the ones used in this project.

The use of ML, specifically ConvNets, has been applied to a number of astronomical contexts. A focus has generally been on morphological classification of stars and galaxies. The methods employed have been largely successful, with studies such as Dieleman et al. (2015) achieving 'near-human' accuracy ($\gtrsim 90\%$) on GZ data. ConvNets have also been shown by Kim and Brunner (2016) to outperform an equivalent shallow learning algorithm when performing star/galaxy classifications. Examples of ConvNets have been used for high redshift studies (Huertas-Company et al. 2015), and when applied to radio galaxies (Aniyan and Thorat 2017), with Huertas-Company et al. (2015) significantly outperforming the shallow learning approach employed for the same data in Huertas-Company et al. (2014). ConvNets are therefore highly flexible and have been proven to produce excellent results on a variety of astronomical data.

In this report, we present a classifier employing a ConvNet model directly on images containing non-extended (stars and other point-like objects), and extended (galaxies and nebulae) sources from GOTO. The sources were detected through a simple Python algorithm and then positionally matched to the Galaxy Zoo catalogue in order to pre-label the sources to create a training set that was input into the model. We then demonstrate that the model produced was capable of a high accuracy when trained on the GOTO images, and can also be extended to classify featured and non-featured objects (late-type and early-type galaxies respectively). In Section 2 we outline the steps undertaken to form the sample images that were fed into the ML algorithm. We provide an overview of the theory and merits of ConvNets in Section 3. The full outline of the model is described in Section 4, including pre-processing (4.1), training (4.2), and testing of the model (4.3). In Section 5 we outline the results achieved and discuss their viability, with our conclusions being presented in Section 6.

2 Sample Data

The selection of a representative sample of objects is imperative for a successful and reliable ConvNet. A representative sample will allow the ConvNet to learn the general features that distinguish each class of object more easily with less errors. In this section we describe the formation of the sample images that would be used as inputs into the machine learning algorithm.

It is of note that the objects categorised were not split into the physical categories of 'star', 'galaxy', 'spiral' and 'elliptical'. This was due to the classification being purely based on the extended nature of the object. While the vast majority of non-extended objects will physically be stars, there is no measurement of properties that can explicitly define these objects as stars. As such, the non-extended label can include anything with a non-extended morphology, which can include quasars and artifacts within the image. Similarly for extended objects, there is a chance that objects identified were not strictly galaxies, instead being nebulae or extended artifacts. The use of general category labels over physically motivated labels was designed to remove ambiguity between the physical nature of objects in the same category. The use of 'featured' and 'non-featured' to describe late-type and early-type galaxies respectively was a consequence of following the nomenclature provided by the Galaxy Zoo catalogue.

2.1 GOTO

The sample images were provided by the Gravitational wave Optical Transient Observatory (GOTO) (GOTO 2018). GOTO currently comprises of four 40cm telescopes providing ~ 20 square degrees field of view, with a pixel scale of ~ 1.25" per pixel (GOTO 2018). The survey was initially designed to detect the optical counterparts of gravitational wave events, which are rare due to requiring the merging of high mass, compact objects (Abbott et al. 2017). The telescopes used to identify the initial gravitational wave events can only currently determine the position to within 10s of square degrees. Hence, it was important for GOTO to implement a wide-area approach due to the large 'error circles'. As a result, the images obtained by GOTO contained thousands of sources that can be categorised into broad physical classes. These sources would require detection and classification through an automated process, due to the vast amounts of sources in each image. The use of the GOTO images therefore provided an ideal demonstration of the performance of the ConvNet with a practical application.

A unique challenge presented by the GOTO data provided was the use of only single band images. Throughout the literature, ML is typically approached using multi-band images (see Appendix B 2.3 for further details). The use of multiple bands allows for colour to be introduced into the algorithm, which can prove valuable when visually classifying sources (Lintott et al. 2008). However, the use of single band images can be beneficial. As previously discussed in Section 1 and Appendix B, morphology is one of the most fundamental properties of an object, while colour can introduce problems due to external processes such as dust extinction and red/blue-shifting altering the colour of the object. While indeed providing an extra challenge, the use of single band images was important for providing more fundamental classifications using intrinsic properties of the object.

2.2 Source detection

As described in Appendix B Section 3.3, source detection is a vitally important step when preparing data for ML. Without any source detection, individual sources cannot be isolated from the larger images, with it being difficult for any algorithm to classify individual sources. As such, it was clear that the source detection must be robust and reliable. In order to optimise the source detection, multiple Python functions were tested and evaluated (see Appendix B Section 3.3 for further details). From these tests it was decided that the `detect_sources`[1] function was the best performing function from visual inspection.

A key component of the detection using `detect_sources` was the threshold for which sources would be detected. The input threshold used could either be a single value, or a 2D 'threshold image', of which both methods were tested on the GOTO data. Initially, a single value threshold was employed, with a discrimination of sources 1.5σ above the median of the image, where σ is the standard deviation of the image. This threshold allowed for the vast majority of sources to be detected in each image, which was determined through visual inspection. However, this led to many faint sources, along with defects to be included into the sample. A detection algorithm for ML purposes does not necessarily require every source in an image to be detected, instead only a representative sample is required for training, so the algorithm is able to learn the general features that define each class. It was therefore decided to employ a higher threshold value, which excluded more sources. It was also decided to implement a threshold that varied across the image, an example of which is shown in Figure 1. The use of a 'threshold image' over a single value allowed for a more detailed threshold to be obtained, as Figure 1 clearly shows the difference in σ across the image due to more objects or differences in the telescopes used by GOTO. Employing a larger threshold value reduced the total number of sources, as seen in Table 1. However, this simple change was invaluable when training the ML algorithm, as accuracies were seen to increase from $\sim 86\%$ to $\sim 95\%$ after implementing the 5σ threshold. The increased accuracy was likely a result of the sources being brighter due to the higher threshold, as such the characteristic features of each object

[1]`https://photutils.readthedocs.io/en/stable/api/photutils.segmentation.detect_sources.html`

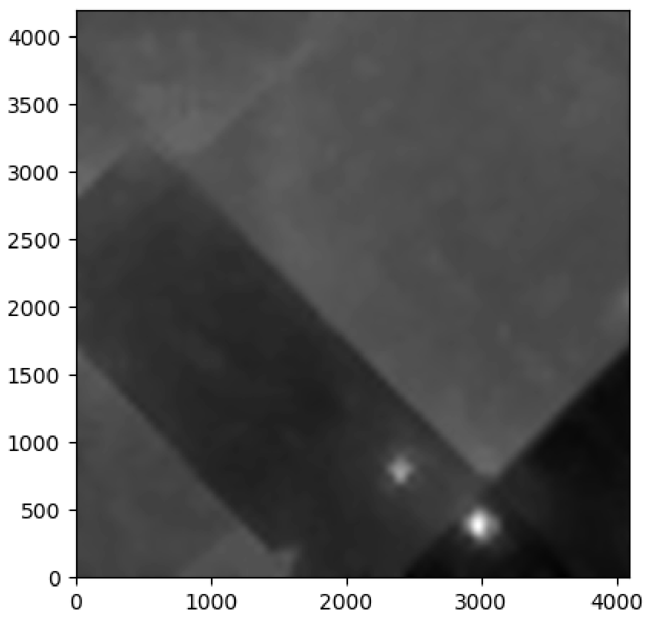


Figure 1: 2D threshold image where the threshold level was set to 5σ, so any sources that were below the value at each pixel are not considered when performing source detection. It is of note that employing this method added very little computation time to the detections. The two 'sources' that appear to be present are due to dense regions of objects.

were more easily visible, allowing for better training of the model. The threshold level of the source detection therefore has a significant effect on the final performance of the model.

Table 1: A demonstration of the effect that increasing the threshold level had on the total number of images, and final accuracy of the ConvNet model described in Section 4 when trained on different levels.

Threshold Level	Threshold type	Total non-extended	Total extended	ConvNet accuracy
1.5σ	Single value	314,000	8,000	$\sim 86\%$
5σ	2D image	200,000	6,000	$\sim 95\%$

2.3 Matching to Galaxy Zoo

Once the sources had been detected by the algorithm presented in Section 2.2, the sources were then positionally matched to the existing Galaxy Zoo (GZ) catalogue[2]. Matching allowed for pre-labelling of the input images, which is a key requirement for any supervised ML algorithm such as the model presented in this project. This matching allowed for the detected sources to be split into the major categories considered for this project.

GZ is a web-based project that has successfully classified > 60 million objects, through the use of a large pool of unspecialised 'citizen scientists' (Lintott et al. 2008). This study has been the largest astronomical collaboration in history (Lintott et al. 2008), and as such has proven immensely powerful for studies of galaxy formation and evolution (Banerji et al. 2010). Labelled catalogues produced by GZ have been shown by Lintott et al. (2008) to agree with professional astronomers to accuracies of $> 90\%$, with the largest catalogue containing $> 300,000$ galaxies.

The GZ project provided the ideal astronomical catalogue to distinguish between the extended and non-extended objects in each image. The classifications made by GZ are primarily galaxy-like extended sources, with only a small fraction of point like sources due to stellar objects having halos or diffraction spikes (Banerji et al. 2010). The extended sources categorised by GZ are derived from the Sloan Digital Sky Survey (SDSS), which covers $\sim 26\%$ of the entire sky (Lintott et al. 2008). The wide area covered

[2]`https://www.zooniverse.org/projects/zookeeper/galaxy-zoo/`

by GZ is crucial for this project, as it allows for significant overlap between regions of sky classified by GZ, and those imaged by the GOTO telescope. Hence, matching detected sources to GZ allowed for segregation between non-extended and extended sources in the GOTO images which overlapped with GZ fields.

The process of matching sources to GZ was simple and added very little computation time to the source detection. This was done by using the `match_to_catalog_sky`[3] function from Astropy, and simply stating that any source that was $\leq 1"$ from a source categorised by GZ, was indeed an extended object. It was assumed that the GZ catalogue contained no stars and was purely comprised of extended sources. This allowed for a simple separation of non-extended and extended sources, where if a source on a GOTO image was within the specified 1" of a source within GZ, that source would be defined as an extended object. This assumption introduced a risk that objects that were clearly extended would not be correctly labelled if they were not included in the GZ catalogue, potentially introducing contaminants into the training set. However, as the GZ catalogue employed contains $> 300,000$ objects, the number of contaminants was expected to be low ($\leq 1\%$) so the model would be largely unaffected by such a small contamination. As GZ also distinguishes between featured and non-featured galaxies, sources from GOTO images could also be split into these categories. Using these simple arguments it was possible to build up a large library of positionally matched sources and categorised images to be fed to the ML algorithm.

The final step before feeding the images into the ML algorithm (described in Section 4.1) was to take cutouts of the individual sources. This enabled the ML algorithm to be performed on the individually detected sources to allow the algorithm to distinguish specific sources. This was done using the `Cutout2D`[4] function and created 30 x 30 pixel 'postage stamps' of each source (for further details see Appendix B Section 3.5). The size of 30 x 30 pixels was chosen in order to reduce the size of the data to the ML algorithm, in an attempt to increase speed. The cutouts were originally 50 x 50 pixels (Appendix B), however only a negligible amount of sources were lost on reduction to a 30 x 30 pixel image, with $> 99.9\%$ being retained on the smaller images. The smaller images enabled a faster training of the algorithm, while still maintaining comparable levels of accuracy. After these processes, we were left with $\sim 6,000$ extended sources and $> 200,000$ non-extended sources to be fed into the ML algorithm.

3 Deep Learning

The concept of deep learning is to build an algorithm that can learn directly from the raw data (LeCun et al. 2015). Primarily in the form of artificial neural networks (ANNs), deep learning algorithms have already outperformed the equivalent shallow learning processes (Aniyan and Thorat 2017), specifically in the field of image classification (Krizhevsky et al. 2012). In this section, we present a brief overview of the theoretical concepts behind ANNs and a sub-species of ANN known as Convolutional Neural Networks (ConvNet).

Neural networks are inspired by biological neurons (Aniyan and Thorat 2017), as the human brain can more efficiently interpret the context of a situation better than computers. ANNs are able to approximate a set of non-linear functions from a set of inputs using relatively simple mathematical concepts (Aniyan and Thorat 2017). The structure of any ANN typically consists of a network of interconnected 'neurons', that consist of multiple inputs leading to a single output (Aniyan and Thorat 2017, Kim and Brunner 2016). The typical layered structure of an ANN is shown in Figure 2 (b) and is designed to replicate the learning process in humans.

The mathematics behind the neural network is relatively simple. The inputs ($\mathbf{x}$) are represented as a vector where $\mathbf{x} = (x_1, x_2, ..., x_n)$, and the weights of each input ($\mathbf{w}$) are also represented as a vector ($\mathbf{w} = (w_1, w_2, ..., w_n)$). There is also an associated bias (b) with each input. The output of each neuron (y) is calculated using Equation 1 (Kim and Brunner 2016), where σ is the activation function of the neuron. Equation 1 is also schematically represented in Figure 2 (a).

[3]`http://docs.astropy.org/en/stable/api/astropy.coordinates.SkyCoord.html`
[4]`http://docs.astropy.org/en/stable/api/astropy.nddata.Cutout2D.html`

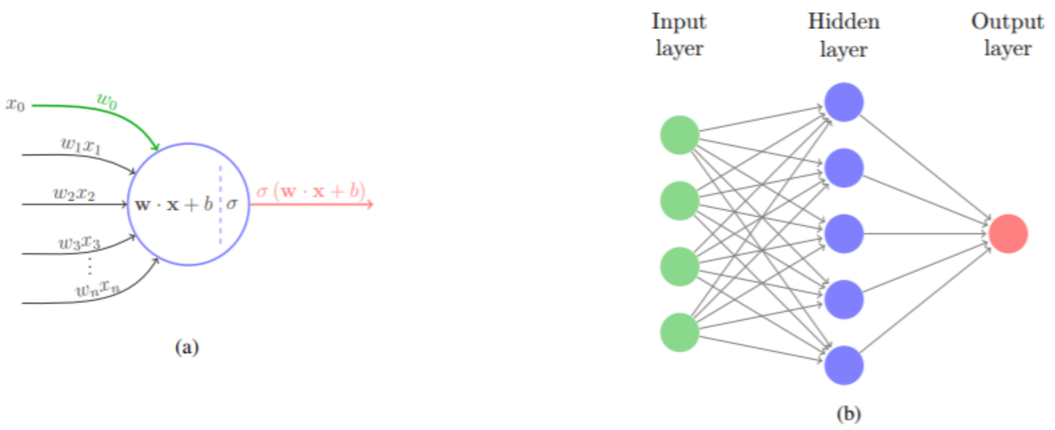


Figure 2: (a) A schematic showing the mathematical model used in each neuron of an ANN. Here, $\mathbf{w}$ represents the vector of input weights, $\mathbf{x}$ is the vector of inputs, σ being the activation function, and b is the bias applied to each neuron. (b) A schematic diagram of an ANN showing a single hidden layer. Image is taken from Kim and Brunner (2016)

$$y = \sigma(\mathbf{w}\cdot\mathbf{x}+b) \tag{1}$$

In many applications of deep ANNs the output of each neuron described by Equation 1 is then passed 'forward' to the next layer and eventually to the final output. There are generally no instances of the output of the model being fed back into itself. This approach is known as feed-forward and is the primary approach used in deep ANNs. This allows for complex functions to be easily learned that can map a fixed size input (e.g. an image) to a fixed size output (e.g. a binary classification) (LeCun et al. 2015).

The choice of activation function (σ in Equation 1) can vastly affect the performance of the model. Activation functions in general are used to define the output of a neuron in accordance with Equation 1, and in its most basic form is a binary function (i.e. the neuron either fires, or it does not). One of the most commonly used activation functions is the rectified linear unit (ReLU) (LeCun et al. 2015). The ReLU function is widely used within deep learning contexts due to it allowing significantly faster training of deep ANNs (Kim and Brunner 2016, LeCun et al. 2015, Krizhevsky et al. 2012). The increased speed is largely due to the shape of the function, which is displayed in Figure 3 (a). The ReLU function is 0 for all negative numbers and linearly increases for positive values. This means that calculations using this function are relatively simple as all negative numbers are set to 0, and there are no computationally expensive exponential, multiplication or division operations. This point is demonstrated in Figure 3 (b), where the ReLU function reaches a 25% error (therefore 75% accuracy) significantly faster than an equivalent tanh function (Krizhevsky et al. 2012).

ConvNets follow a very similar structure to ANNs, having a network of feed-forward neurons parameterised by a mathematical function. However, the mathematical function replaces the dot product shown in Equation 1 with a convolutional operator (Kim and Brunner 2016). This sum of convolutions causes the output of each layer to be known as feature maps (Kim and Brunner 2016, LeCun et al. 2015). The convolution layers are then paired with a pooling layer to create the feature maps. The pooling layer acts to reduce the amount of parameters by reducing the size of the network (Udofina 2018). An example shown in Figure 4 is the Max Pooling layer (Udofina 2018), which calculates the maximum of a local patch and removes all other values (LeCun et al. 2015). While similar to ANNs, the combination of convolution and max pooling layers allow CNNs to be easier to train, with a comparable accuracy to ANNs (Krizhevsky et al. 2012) due to reducing the size of the data while maintaining the important distinguishing features.

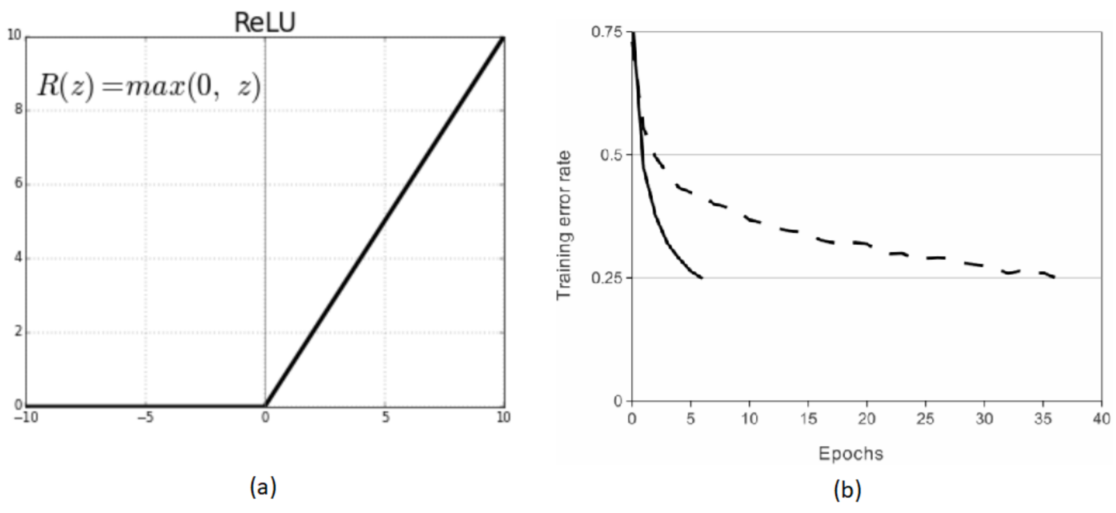


Figure 3: (a.) ReLU function plotted, image taken from `https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6`. (b) Image taken from Krizhevsky et al. (2012) showing that the ReLU activation (**solid line**) reaches a 25% error rate 6 times faster than an equivalent network with a tanh activation function (**dashed line**), demonstrating the increased speed of the ReLU activation function.

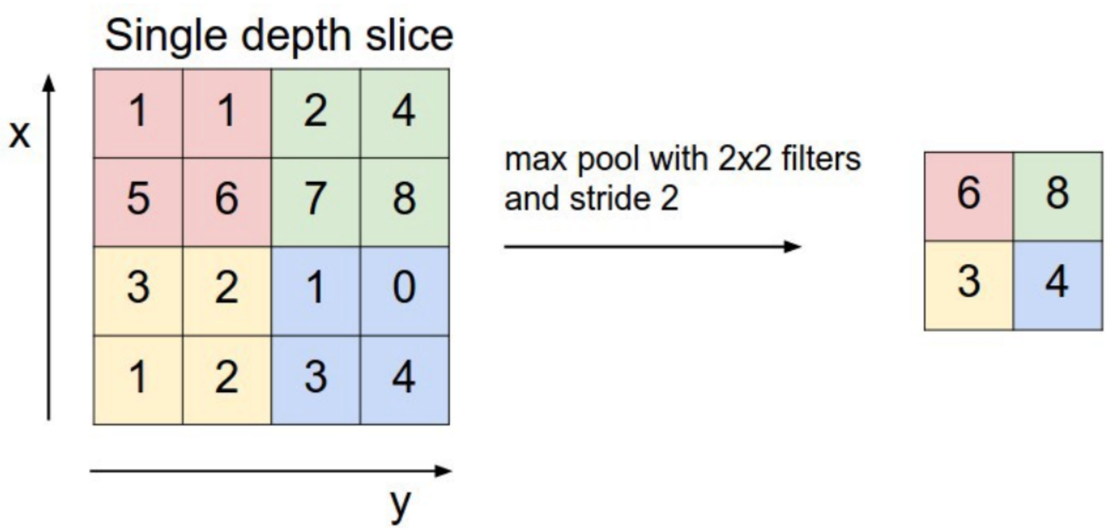


Figure 4: A demonstration of the 'Max Pooling' layer in a ConvNet. Image taken from Udofina (2018)

A key concern for ConvNets is the concept of overfitting. Overfitting is when the model is 'too good' at learning the patterns in the training data (Bilbao and Bilbao 2017). The concept of overfitting is demonstrated in Figure 5 where, despite the training set showing a low error, the error on the unseen validation set continues to increase. In extreme cases this can lead to a situation where an algorithm is 100% accurate on the training data, but only 50% accurate on the test data (Domingos et al. 2012). Overfitting can also manifest when presented with unbalanced training sets. From a typical image in the GOTO dataset the percentage of non-extended objects in the image can range from $\sim 95 - 100\%$. If this ratio was maintained during training the algorithm could potentially learn to identify every object as non-extended. This would result in a model with a high accuracy on training data, but when applied to unseen test data the accuracy would be $\sim 50\%$, similar to Figure 5. Methods of combating overfitting and providing equal training samples are key for creating a high quality algorithm.

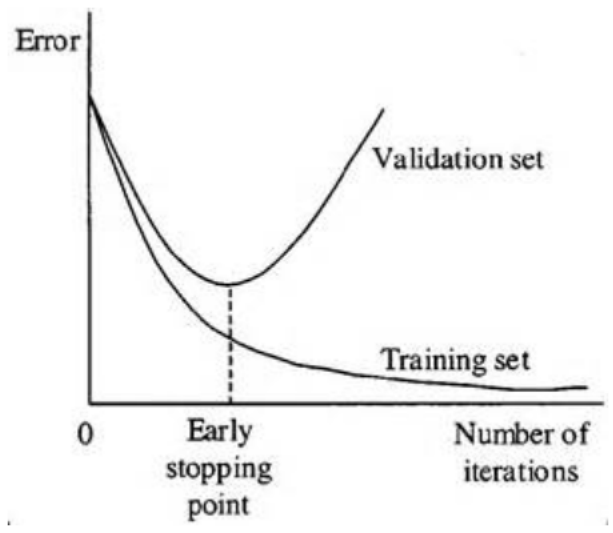


Figure 5: A demonstration of the effect of overfitting on a training and validation set. The early stopping point represents the optimum point to stop training and prevent overfitting.[6]

ConvNets have been successfully applied to a wealth of astronomical contexts (see Appendix B Section 2.3 for further details), with a primary focus concerning morphological classification. ConvNets have produced record-breaking results in the field of image classification (Kim and Brunner 2016, LeCun et al. 2015), with consistently high accuracies across a range of astronomical applications. The fact that the network can learn directly from the data is also advantageous for this study, as it did not require the use of potentially subjective features to distinguish between the different classes. These factors combined allowed ConvNets to be the most suitable choice of algorithm for this study.

4 Network Model

In general, there is no strict guideline to the optimum design of the network, as the architecture is designed to optimise the parameters of the specific problem (Aniyan and Thorat 2017). Hence, the decision for the number of hidden layers and neurons is related to the complexity of the data (Aniyan and Thorat 2017). For ConvNets, the complexity of the model generally increases with the complexity of the classifications. For example, an extended/non-extended classifier will require fewer layers than a featured/non-featured classifier, as the distinction between featured and non-featured sources is more subtle. This was shown through initial tests of a simple 4 layer network when classifying extended/non-extended and then featured/non-featured objects. In general, the distinction between extended and non-extended objects is considerably more simple, hence the 4 layer network provided accuracies of $\sim 97\%$ with extended/non-extended sources. However, the poor resolution of the GOTO image data (~ 1.25") can obscure the key distinguishing features between featured and non-featured sources; hence the same simple 4 layer network could not exceed 70% accuracy when considering featured and non-featured sources.

This study is primarily concerned with the classification of extended and non-extended objects, however the classification of featured and non-featured objects is also a consideration. Due to the complexity associated with featured/non-featured classification it was important to implement a model with many trainable parameters. These parameters would be able to determine the subtle differences between featured and non-featured objects, as well as providing strong classifications of extended/non-extended objects. The model employed for both classifications is displayed in Table 2. The large number of trainable parameters shown allowed for both classifications considered to be learned effectively.

[6]Image taken from: `https://elitedatascience.com/overfitting-in-machine-learning`

Table 2: Outline of the model architecture when considering both classifications.

Layer (type)	Output Shape	Params
conv2d (Conv2D)	(None, 28, 28, 64)	640
activation (Activation)	(None, 28, 28, 64)	0
conv2d_1 (Conv2D)	(None, 26, 26, 100)	57700
activation_1 (Activation)	(None, 26, 26, 100)	0
conv2d_2 (Conv2D)	(None, 24, 24, 136)	122536
activation_2 (Activation)	(None, 24, 24, 136)	0
conv2d_3 (Conv2D)	(None, 22, 22, 172)	210700
activation_3 (Activation)	(None, 22, 22, 172)	0
conv2d_4 (Conv2D)	(None, 20, 20, 208)	322192
activation_4 (Activation)	(None, 20, 20, 208)	0
conv2d_5 (Conv2D)	(None, 18, 18, 244)	457012
activation_5 (Activation)	(None, 18, 18, 244)	0
max_pooling2d (MaxPooling2D)	(None, 9, 9, 244)	0
dropout (Dropout)	(None, 9, 9, 244)	0
flatten (Flatten)	(None, 19764)	0
dense (Dense)	(None, 256)	5059840
activation_6 (Activation)	(None, 256)	0
dense_1 (Dense)	(None, 2)	257
activation_7 (Activation)	(None, 2)	0

Total params: 6,230,877
Trainable params: 6,230,877
Non-trainable params: 0

Train on 41436 samples, validate on 10360 samples

The model consists of six convolutional layers, each followed by an activation layer. The activation function chosen was the ReLU function described in Section 3. This activation function was chosen as it allows for significantly faster training of the algorithm largely due to simple calculations, with all negative values set to 0, and no exponential, multiplication, or division operators. Following the final convolution and activation layer was a max pooling layer in order to reduce the computational complexity. After the max pooling layer there was a 'dropout' layer. Dropout is a technique of reducing overfitting where the output of each hidden neuron is set to zero (Krizhevsky et al. 2012), with a probability that can be set by the user. As any neuron can be removed at any time, neurons cannot rely on the presence of other neurons in the same layer and so are required to learn more robust features (Kim and Brunner 2016). Without the dropout layer, the model would exhibit strong overfitting. The final layer had only two neurons corresponding to the two possible classifications.

4.1 Image pre-processing

Before training of the model could begin, certain image pre-processing steps were required in order to maintain the homogeneity of the samples (Aniyan and Thorat 2017). For ConvNets these steps are especially important as the neurons behave similarly to the human eye, so if a human can see an object, the network must also be capable of 'seeing' the object (Aniyan and Thorat 2017). The specific steps taken to maintain a homogeneous sample space are outlined in Section 4.1.1 - 4.1.2.

The initial focus of this project was primarily concerned with the classification of solely extended objects, due to non-extended classification being less represented in the literature. It was therefore desirable to remove any non-extended sources before training the ML algorithm. A non-ML approach was adopted for the extended object cut, with the primary reason being to increase the overall speed of the eventual ML algorithm. However, as described in Appendix B Section 3.4, the image data provided by GOTO proved challenging to realise these cuts.

The non-ML cuts were eventually abandoned in favour of performing a more general extended/non-extended ML classification. A key factor for abandoning this step was the lack of a robust and reliable cut that would be applicable to all the training images. This also allowed for the focus of the project to shift to one of categorising extended and non-extended objects, which is something that is highly represented in the literature.

4.1.1 Data Augmenting and overfitting

An important requirement for all neural networks is having a large amount of training samples. With a large number of training images the network will be able to more easily learn the features which distinguish the different classes (Aniyan and Thorat 2017). Generally, the total number of training images should exceed 10,000 samples in each category for deep ANNs (Aniyan and Thorat 2017). From the source detection presented previously, and from Table 1, it was desirable to increase the number of extended sources available for training.

One common method of preventing overfitting is to artificially increase the training dataset using methods that preserve the pre-assigned labels (Kim and Brunner 2016, Krizhevsky et al. 2012). From Table 1 it was clear that the extended objects would be the primary class subjected to data augmentation. The core method of augmenting the data was by rotating the images by small angles of $\pm1°, \pm2°, \pm3°, \pm4°, 90°$, and 270°, where a representation of the rotation is displayed in Figure 6. Rotation has no effect on what class the object was and simply acted to inflate the number of images in each class. In order to increase the total number of images further the images were also flipped horizontally and vertically. In addition to augmenting the dataset rotating the images also acts to improve the final model, as the model had to learn rotationally invariant features of the objects (Kim and Brunner 2016). The number of rotation angles was chosen in order to provide a significant number of images in the extended class to allow for the features to be correctly learned. The final step was to choose a random number of the non-extended class, so that each class would have the same number of sources, resulting in $\sim 62,000$ images in each class for the extended/non-extended classifier.

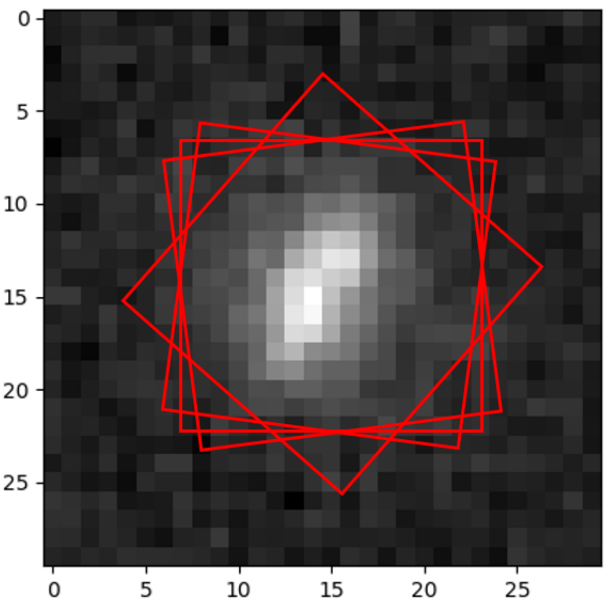


Figure 6: Representation of the rotation steps implemented into the image pre-processing phase of the algorithm. The red squares represent the rotation of the image through small angles and 90°.

4.1.2 Normalisation

In many ML applications it is likely that the features chosen by the model will have different ranges (i.e. one feature might have values that range from 0-1, while another has a range of 50-100,000). When doing complex analysis it is possible that the feature that is larger will influence the result purely due to its larger value, but it does not mean that this feature is a better discriminator (Jaitley 2018). The affects of neglecting image normalisation is demonstrated in Figure 7, where it is clear to see that the normalised data had a significantly higher accuracy at each epoch, while maintaining a lower loss value. The non-normalised data maintains a $\sim 50\%$ accuracy over the 10 epochs. The accuracy remains flat due to the different features chosen by the model having different ranges, so will take a significantly longer time to learn any features. It is therefore common to include normalisation of the data in ML to apply a common scale for all features.

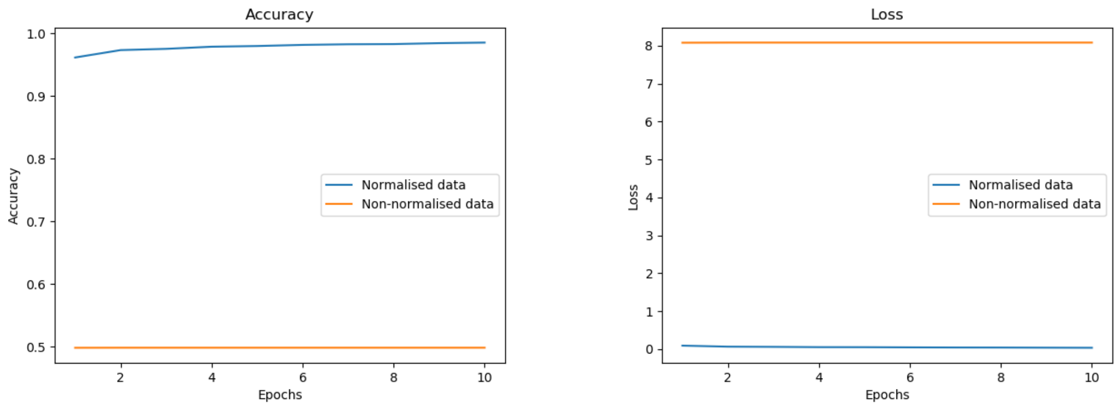


Figure 7: A demonstration of the value of normalising data in ML applications. *Left* shows the effect on the accuracy, while *Right* show the effect on loss. The non-normalised data shows a straight line in both plots showing that it is not training and so not learning the features.

The method of normalisation employed is stated in Equation 2, with a comparison between a standard image and a normalised image being shown in Figure 8. Here z is the normalised image, x is the original image, $\bar{x}$ is the mean of the image, and σ is the standard deviation.

$$z = \frac{x - \bar{x}}{\sigma} \tag{2}$$

Employing this method allowed for fast normalisation of the images, which was key when considering the large datasets used. This form of normalisation is known as the z-score and preserves the same distribution as the original image. Therefore, the images will still retain the original detail, but the features determined by the ConvNet will be over the same numerical range. As the mean and standard deviation of each image can be easily found, this method of normalisation was the most suitable due to its speed and the ease at which it was able to be implemented.

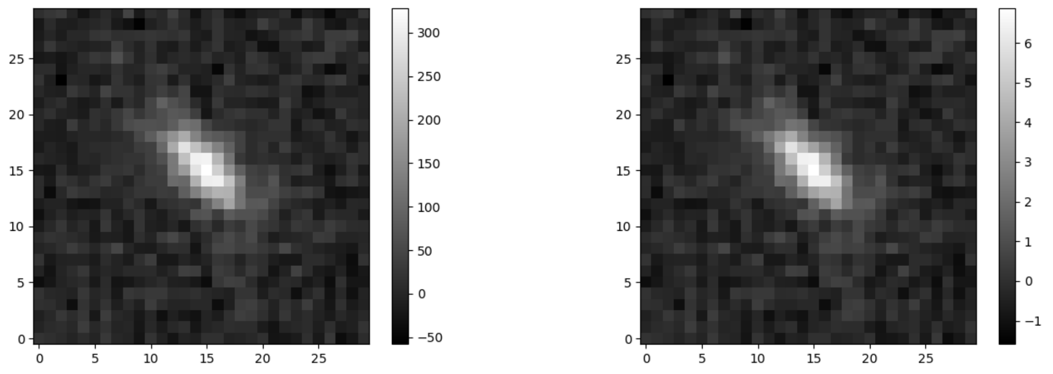


Figure 8: Demonstration of the effect that the normalisation process described in Equation 2 has on a single image of an extended object. *Left* shows the non-normalised image with the larger range of pixel values, while *Right* shows the normalised image with a tighter range of values while still maintaining the original detail. The $\bar{x}$ and σ values for the normalised image are ~ 0 and 1 respectively.

4.2 Model training

Training a machine learning model generally requires the data to be split into two parts. The first part of the split is typically the majority split and is used for training the model, while the second, minority split, is used to validate the performance of the model during training (Aniyan and Thorat 2017, Krizhevsky et al. 2012). For the validation to provide a useful indication of the performance, it is imperative that the validation set remains unseen during training. For all models produced for this report an 80-20 split ratio was implemented, therefore 80% of the data was used for training, with 20% used purely for validation.

It was also important to perform a further split to create an unseen test set which would evaluate the performance of the trained model. This split was again an 80-20 ratio. Applying these splits reduced the total number of extended and non-extended sources to $\sim 72,000$ for training, with $\sim 18,000$ images for validation.

The algorithm was implemented using the Tensorflow and Keras Python modules and are widely used for deep learning purposes. These modules provided several built-in functions that allowed for the model to be coded much quicker and more efficiently than initial predictions in Appendix A. The speed at which highly accurate code could be created using Tensorflow and Keras also enabled us to perform featured/non-featured classifications, which was previously not expected to be possible considering the time constraints of the project and the initial plan outlined in Appendix A. Tensorflow also provided the option to be supported on GPUs. This proved invaluable when training, as the use of GPUs allowed for significantly faster training. Traditionally, Python packages are installed directly onto the CPU that consists of a number of highly optimised processors. Generally CPUs only have one 'pathway', meaning that operations can only be performed sequentially. GPUs however, have multiple 'pathways' allowing for multiple operations to be performed simultaneously. As the ConvNet has to perform a large number of calculations, the ability to perform these calculations simultaneously by installing Tensorflow on the GPU allowed for training to be performed up to 10 times faster than when using a traditional CPU. Training was therefore done on a machine with Tensorflow GPU installed on a Nvidia GeForce GTX 750Ti taking ~ 40 mins to train for ~ 16 epochs.

As with any ML process employing a ConvNet, overfitting is a key issue that must be mitigated for a successful algorithm. The model in Table 2 contains $> 6 \times 10^6$ trainable parameters, with only $\sim 7.2 \times 10^4$ images the model is very likely to overfit without regulating. Mitigating overfitting was already discussed in Section 4.1, however there are further methods of prevention made available during training through implementing 'callbacks'. Callbacks are modules that can be called at certain points during training[7]. The key callbacks implemented were TensorBoard, and EarlyStopping. TensorBoard simply allowed for dynamic visualisation of the progress of the model, and EarlyStopping allowed the model to be stopped once a monitored quantity had stopped improving. The EarlyStopping callback was crucial as it could stop the model before the consequences of overfitting had taken effect, such as the error on the validation set increasing as seen in Figure 5. These steps combined with the image pre-processing drastically reduced the chance of overfitting occurring, hence improving the final model.

During training, it was useful to have certain metrics that measured the performance of the model. A standard metric that was employed was accuracy. The models employed in this project have only two possible classifications (extended or non-extended, and featured or non-featured), hence it was a binary classifier, with the accuracy being calculated by Equation 3

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

where TP = true positives, TN = true negatives, FP = false positives, FN = false negatives[8]. The model also evaluated the loss, where loss is a calculation of how bad the model's prediction was on a single example. For example, if a model produces a result which strongly deviates from the true results, the loss will be very large. These values were calculated at the end of each epoch during training. Using these metrics it was possible to create training curves, an example of which is shown in Figure 9 to evaluate the performance of the model during training.

4.3 Model testing

During training the model outputted the accuracy and loss on the training and validation sets that were useful metrics in order to assess the performance of the model. However, a more useful test of the performance was through attempting to classify unseen data in the test set. Using unseen data was vitally important as the accuracy quoted on the test data could be a result of overfitting, so the model would perform well on training data but poorly on test data. For consistency, the metric used to assess the performance of the model on the test data was also accuracy.

The testing process was implemented using the `predict_classes` function within Tensorflow. This function produced a binary classification of each input image based on the trained model. The output of this function was then compared with the true classes defined in the training set in order to calculate an accuracy through Equation 3. It was also useful to implement some method of visualising the different terms in Equation 3 to provide a more straight-forward method of assessing the performance of the model.

[7]`https://www.tensorflow.org/api_docs/python/tf/keras/callbacks`

[8]Equation reference: `https://developers.google.com/machine-learning/crash-course/classification/accuracy`

This was done through creating confusion matrices, which show the true positives, true negatives, false positives, and false negatives of each class. A further test was implemented to display the images of sources that were incorrectly labelled by the ConvNet, which are both shown in Section 5. This step was influential when iterating and improving the model, as it helped to determine problems in the dataset that could have caused the algorithm to mis-classify the sources.

5 Results and Discussion

Overall the results generally show we have developed a strong algorithm, with a high level of accuracy across binary classifications of non-extended/extended and featured/non-featured classifications. The accuracies reported are comparable to human classification ($\gtrsim 90\%$), while being significantly faster. However, a key problem for ConvNets in general is they are considered 'black boxes', where the outputs are hard to understand and are not 'human-readable'. As such, in order to be able to evaluate the performance of each model in more detail it was imperative to incorporate 'human-readable' visualisations of the output of the models. Common methods of visually presenting the performance of a model is through training curves and confusion matrices, that were implemented into the model and are displayed in this section.

Figure 9 shows the performance of both models over a training period of 14 epochs. Both classifiers reach $> 95\%$ accuracy on the training data, but show a lower validation accuracy that is expected from having fewer samples. The slight increase in loss on the validation set of the extended/non-extended classifier after ~ 8 epochs could indicate that the model was beginning to overfit. However, the increase was negligible and training was stopped by the EarlyStopping callback before the problem became more serious. The curves of both accuracy and loss are more flat for the extended/non-extended classifier compared to the featured/non-featured curves. This suggests that the extended/non-extended classifier was not learning as many features as the featured/non-featured classifier beyond the first few epochs. This was likely a result of the large number of trainable parameters in Table 2 efficiently learning the distinguishing features significantly faster, as the features distinguishing extended and non-extended objects were much more obvious. It is therefore likely that the same accuracy could have been achieved when trained for fewer epochs, reducing the computational time significantly.

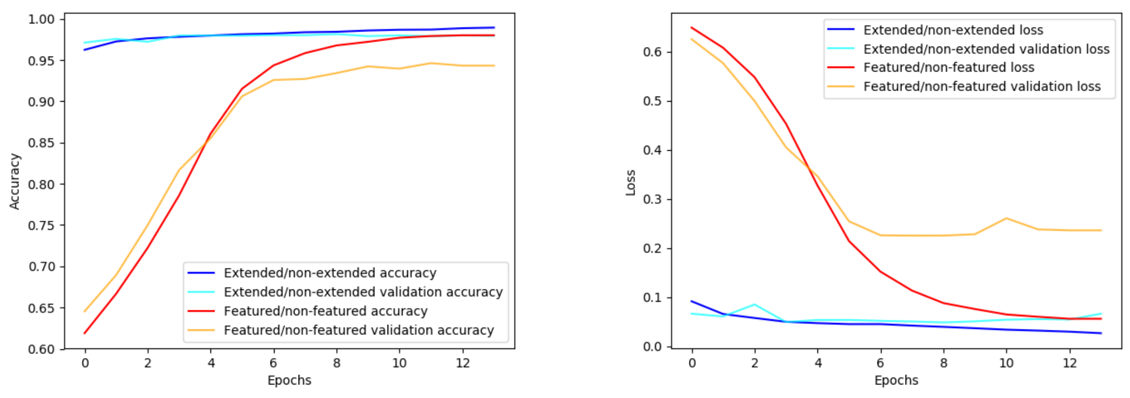
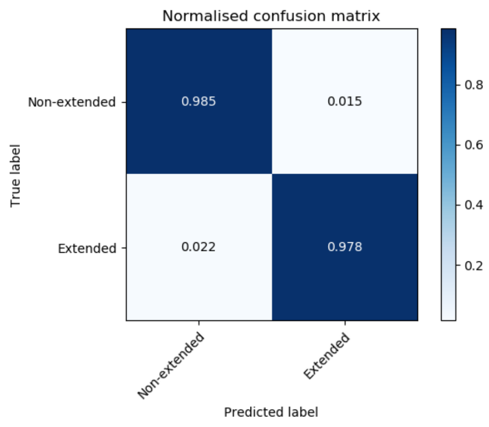


Figure 9: Training curves to visualise the performance of the models when classifying extended/non-extended and featured/non-featured sources over 14 epochs. *Left* shows the accuracy during training and *Right* shows the loss calculated. Both models were trained for a total of 14 epochs before stopped by the EarlyStopping callback to prevent overfitting.

While showing strong results from the training data, the true indication of the performance of the models was when tested on unseen data. A clear visualisation of the results was through confusion matrices, shown in Figure 10. Confusion matrices are means of summarising the output of the model, through displaying the terms in Equation 3 in a clear visual format. The confusion matrices generally show strong models, with true positives for both classes of $> 90\%$ for both models and the extended/non-extended classifier reaching 98.5% true positives for non-extended sources. While still achieving a high accuracy, it is of note that both the extended and non-featured classes achieve lower accuracies in their respective models.



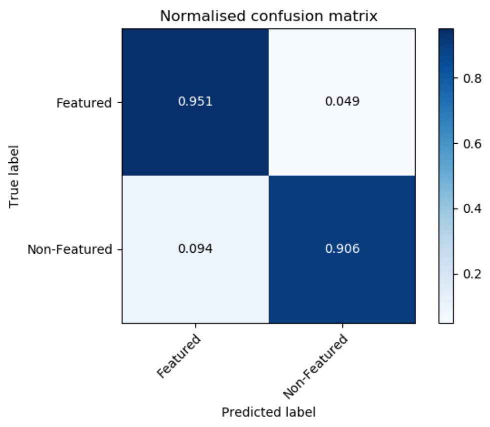


Figure 10: Normalised confusion matrices produced for the (*Left*) extended/non-extended model and (*Right*) featured/non-featured model.

It is difficult to determine why the models achieve the results they do from Figure 10, specifically why the results for the extended and non-featured classes are lower than the non-extended and featured classes respectively. Examples of 25 incorrectly labelled images from the extended/non-extended (Figure 11) and featured/non-featured (Figure 12) models are presented in an attempt to understand this. From outputting the incorrectly labelled images it was clear that there were problems with the input images, especially with the extended/non-extended model in Figure 11. These issues appeared to be a result of images being distorted, and multiple sources in the images. The number of multiple sources in the featured/non-featured image was significantly less, with only 1 example compared to the 4 examples in the extended/non-extended model. However, the increased number of training samples produced a higher overall accuracy in the extended/non-extended classifier. While only taking a very small sample, it is clear from Figures 11 and 12 that the quality of the input images was a major problem within the model, and is discussed further in Section 5.1.

During testing the probabilistic output of the model was converted into class labels through a probability threshold. This threshold stated that if an object had $P_{class} < 0.5$ the object was non-extended, and $P_{class} > 0.5$ the object was extended as an example (Kim and Brunner 2016), where P_{class} is the probability of being in a specific class. While this method has been used in the literature (Henrion et al. 2011, Fadely et al. 2012), it does ignore any scientific requirements the model may have, with potential implications on unbalanced test sets (Kim and Brunner 2016) as described in Section 5.2. Such a low threshold can also present the scenario where an object has an equal probability of being classified in either class, and choosing a single class to classify it as would not result in a good classifier. To overcome this it would have been beneficial to increase the probability threshold to allow for stronger predictions in each class. Another solution would be to output the probability values directly along with the right ascension (RA) and declination (Dec) values of each source (similar to Aniyan and Thorat (2017)). This would allow for clear visualisation of the outputs of the model, and would allow any potential future user to define a suitable probability threshold. Outputting the RA and Dec values would also allow for manual confirmation of the class of the object in the case of ambiguously classified objects that are classified close to the probability threshold.

It is of note that the extended/non-extended model consistently produced higher accuracies across the training and test data compared to the featured/non-featured model. In general, ConvNets improve in performance with more input data. As the extended/non-extended training set contained $\sim 72,000$ training images, compared to the $\sim 41,000$ for the featured/non-featured training set, it was likely that this was the cause for its increased accuracy and should be a consideration when moving forwards.

Obtaining automated, accurate classifications is particularly important for astronomical surveys due to the large amounts of data that will be obtainable with the advent of more advanced telescopes and computational processes. As such, there will always be a requirement for high quality automatic classification within an astronomical context. However, there are still notable problems with the model such as the images input into the models and the concept of the ‘accuracy paradox’. These problems are explored in Sections 5.1 and 5.2 respectively.

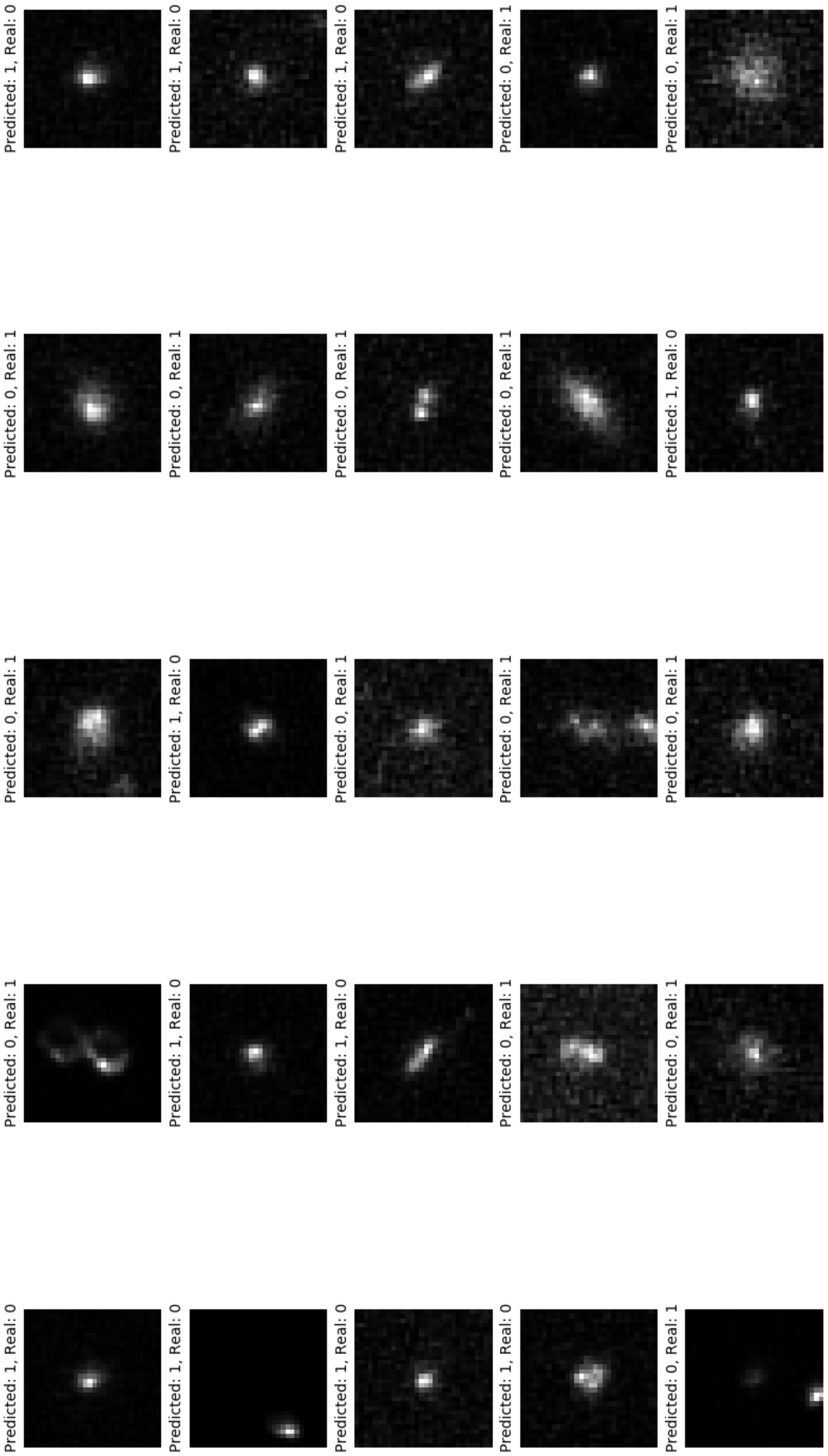


Figure 11: Examples of 25 incorrectly labelled images from the extended/non-extended model. In this case class 0 is a non-extended object, and class 1 is an extended object.

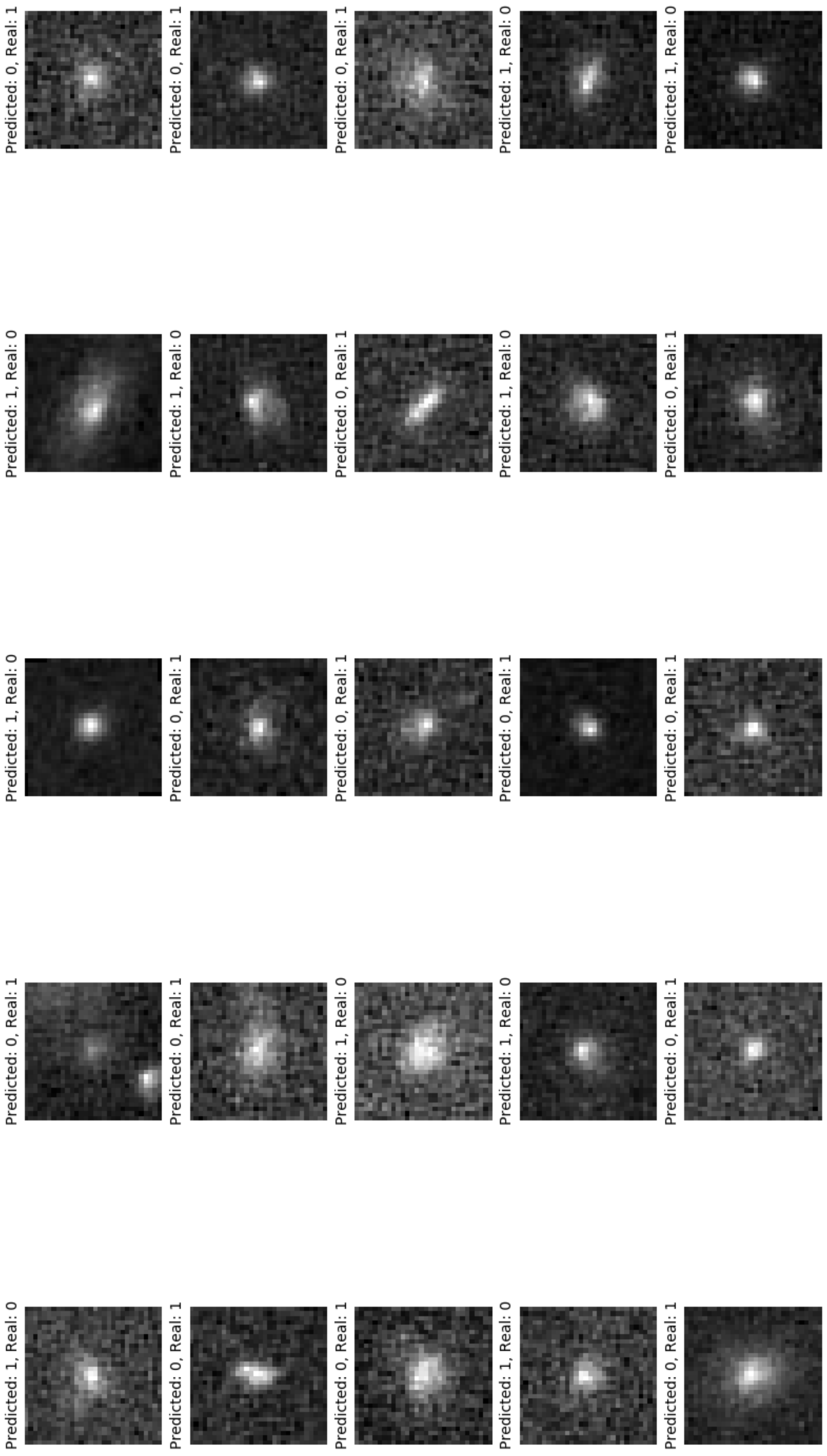


Figure 12: Examples of 25 incorrectly labelled images from the featured/non-featured model. In this case class 0 is a featured object, and class 1 is a non-featured object.

5.1 Input Data

A common statement within machine learning is that 'the network is only as good as the input data'. This is particularly important for ConvNets as the neurons behave similarly to how human eyes behave (Aniyan and Thorat 2017). It was therefore important to feed high quality, unambiguous data into the model in order to achieve the optimum accuracy.

While overall the ConvNet was able to produce accurate classifications using the input data provided by GOTO, there were still several problems present throughout. These problems were largely related to the quality of the images, as several defects were present in all images, along with an inherently poor resolution. Problems were also associated with the reliance on the Galaxy Zoo catalogue for separating the extended and non-extended objects when creating the training set. If steps were implemented to mitigate these problems, it is suspected that the ConvNet will have been able to produce more reliable and higher accuracies. In the following subsections, we outline a number of steps that could have been taken to achieve this goal.

5.1.1 Data 'cleaning'

Several image pre-processing steps were outlined in Section 4.1 that were designed to improve the algorithms' predictive power. These pre-processing steps were of particular importance due to employing a ConvNet, as a homogeneous sample set is crucial. However, despite these steps there were still several problems with defects and multiple sources in a single cutout being prevalent across all images, examples of which are shown in Figure 13. Attempts were therefore made to reduce the number of these 'defect' images, effectively 'cleaning' the dataset, in order to improve the accuracy between extended and non-extended objects.

One of the problems we encountered was the presence multiple sources in an image, specifically where the off-centre source was significantly brighter than the central one, so the centre of the image appeared dark as in the *bottom right* image in Figure 13, and there are two examples of such instances being incorrectly classified in Figure 11, and a single example in Figure 12. A simple, computationally inexpensive method was implemented to attempt to reduce the number of these images reaching the ML algorithm. This was achieved through stating that if a source was within a defined pixel distance to a neighbouring source, the image would not be passed into the training or test sets. The equation to calculate the distance used simple geometrical arguments as shown in Equation 4 where r_{source} is the pixel distance between sources, and x_{image}, y_{image} are the x and y dimensions of the image respectively, so for the images used in this model $x_{image} = y_{image} = 30$. In theory this method would remove the multiple source images entirely from the dataset.

$$r_{source} \geq \sqrt{(\frac{x_{image}}{2})^2 + (\frac{y_{image}}{2})^2} \tag{4}$$

This simple method proved partially successful, as images with multiple sources were removed from the training and test datasets. The reduction in multiple sources in an image had an impact on the performance of the model, with an increase in accuracy during training of $\sim 1-2\%$. However, not all the images were removed, with an estimated $\sim 50\%$ of multiple source images remaining in the data through visual inspection. It was hypothesised that the whole sample of multiple images was not removed due to the calculation requiring the sources to be adjacent to each other in the list of detected sources, which was clearly not always the case. Using the method outlined in Equation 4 would potentially exclude highly dense regions, which could have a noticeable impact on the number of sources if retrained on data focusing on dense regions of space.

However, the images with multiple sources are likely not the major concern with the data. The vast majority of images were shown through visual inspection to have bright central sources that dominated the image. This allowed the ConvNet to be able to learn that the central source is the important source in the image for the majority of cases. The concerns were raised when the off-centre source was dominant, causing the central source to be faint. It was unclear whether the ConvNet was still learning features and producing accurate predictions on the central image in this case. It would have been desirable to produce maps of the output at each layer. These 'feature maps' are prevalent in the literature (Kim and Brunner 2016, Aniyan and Thorat 2017, Krizhevsky et al. 2012, Dieleman et al. 2015, LeCun et al. 2015) and help to better understand the training process of the model through visualisation of the features learned at each layer. While the issue with multiple sources in the cutouts might not have been major, it was thought that the inclusion of large-scale defects in the training sample was a much more serious concern.

Another problem we encountered was defects in the image, an example of a defect is shown in the *bottom left* of Figure 13. These defects were common throughout the GOTO images, with several images

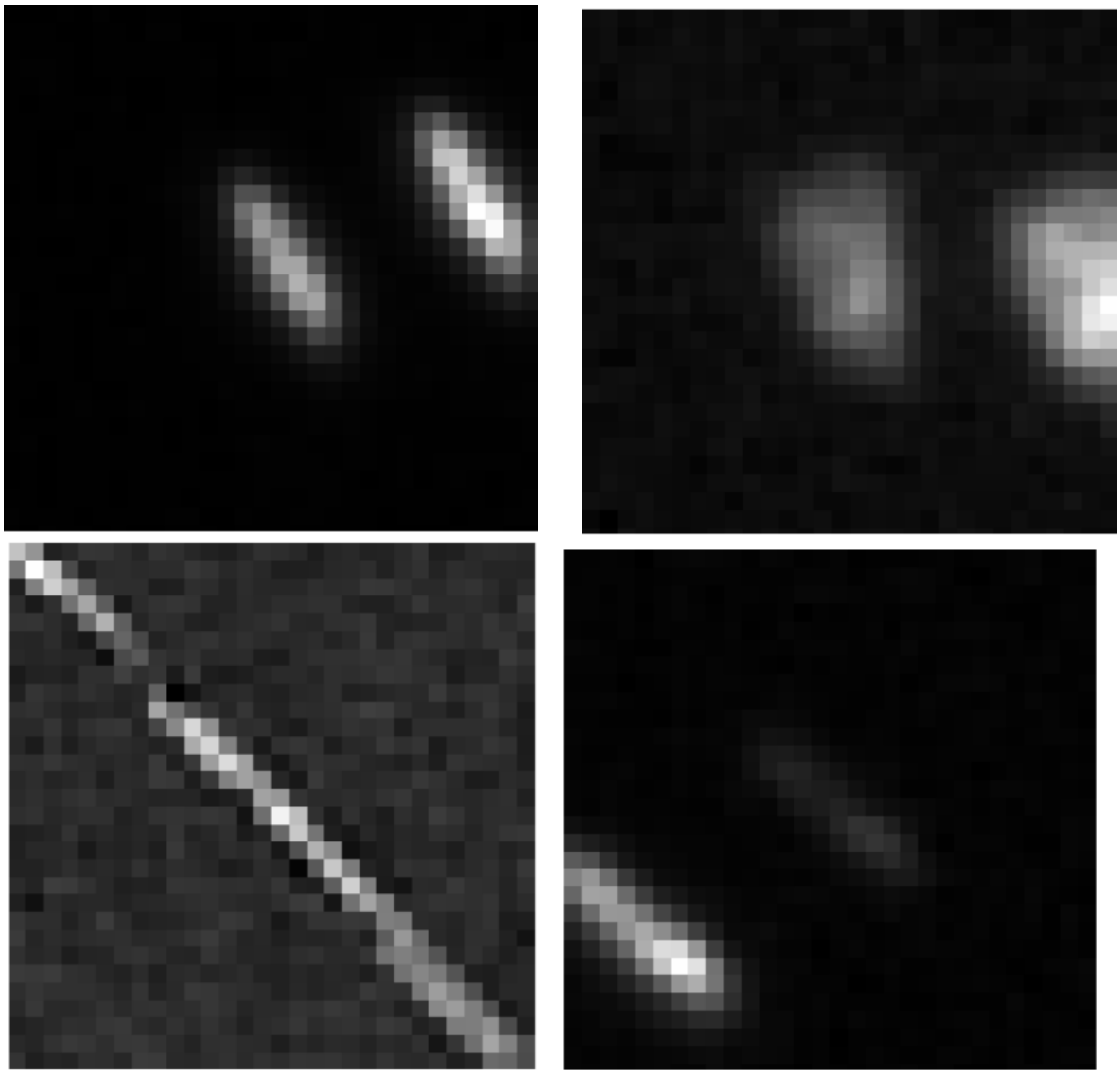


Figure 13: Examples of the defects and double detections that were prevalent across all images in the GOTO sample.

containing multiple defects that stretched across the majority of the image. It was hypothesised that these were either due to some form of transient object (shooting stars, or satellites etc.) or an error with the CCD as these defects were prevalent in all images. It was therefore desirable to remove these images, with consideration put into creating an additional class during training containing defects. However, the method employed for multiple sources performed poorly, with very few defects being correctly removed through visual inspection. The source detection algorithm also had multiple parameters that could be adjusted, although none proved capable of removing the defects efficiently. It was therefore decided to abandon the techniques to attempt to remove these objects despite how common they were across the GOTO sample.

5.1.2 Poor quality images

The resolution of an astronomical image is highly important, with high resolution images allowing for more detailed structures of the sources to be visible. Having a sufficiently high resolution is particularly important in astronomical applications as the information obtained from images is currently the only possible method of probing the properties of objects. The typical pixel scale of telescopes is < 1" per pixel, with the SDSS telescope having a pixel scale of 0.346"[9]. However, the image data provided by the GOTO telescope had a pixel scale of ~ 1.25" meaning that several features of each image will be irrecoverably lost. This was a particular concern when categorising featured and non-featured objects, where the distinguishing features were much more subtle.

The poor resolution can also introduce problems when matching extended sources to the Galaxy Zoo catalogue. As described in Section 2.3 sources were matched to within a ≤ 1" radius. This provided matches to a resolution that was less than the resolution of the image. Therefore, this created sub-pixel measurements that limited the number of sources that were correctly matched. It is known that increasing the matching radius increases the number of sources matched, with a 5" radius increasing the number of matched sources by a factor of ~ 2. While the reduced number of extended sources matched was not necessarily important, as only a representative sample was required for effective training, the reduced number of matches could cause there to be objects labelled as non-extended when in reality they are extended and vice-versa. If numerous enough, these contaminant images could have seriously affected the performance of the model and contributed to the incorrectly labelled images seen in Figures 11 and 12.

The use of a more complex model with more trainable parameters was able to successfully categorise featured and non-featured objects. As this method was more computationally intense; the time taken to train the model almost doubled. However, this increased training time is unlikely to present a significant problem as once the model is trained to a suitable accuracy, theoretically it does not require retraining. However, it is unknown how the model will perform on higher resolution data. The features which distinguish between classes that have been learned by the model are specific to the low resolution images, therefore there is a potential that the model will require retraining.

5.1.3 Reliance on Galaxy Zoo

The Galaxy Zoo (GZ) database was the only method of separating the extended and non-extended objects within our samples. Separating at this stage allowed for pre-labelling of the training data for the ConvNet to learn features that distinguishes between the classes. GZ was chosen as it primarily consists of only extended sources and covers a wide region of the sky due to being derived from SDSS data.

While GZ does cover a wide area of the sky, the catalogue is not completely comprehensive. The GZ catalogue we used (Galaxy Zoo 2) contained $> 300,000$ galaxies across the whole region of the SDSS data (Willett et al. 2013). While being an order of magnitude greater in size than the previous largest galaxy survey (Willett et al. 2013), there is still potential for misclassifications and incompleteness in the dataset. This is only amplified when considering the comparatively small region of overlap between GZ and GOTO.

Positionally matching sources to the GZ2 catalogue was the only method employed to separate extended and non-extended objects for pre-labelling the training set. Therefore, if there was an object that was clearly an extended object, but missing in the GZ2 catalogue, then it would be labelled as a non-extended object. The size and scope of the GZ2 catalogue meant that the number of mis-labelled extended objects was expected to be few, so the model would be able to distinguish them as extended objects once trained. However, when producing plots such as Figure 11 the predicted labels were compared against the pre-labels defined during source detection. If these pre-labels were incorrect then the model

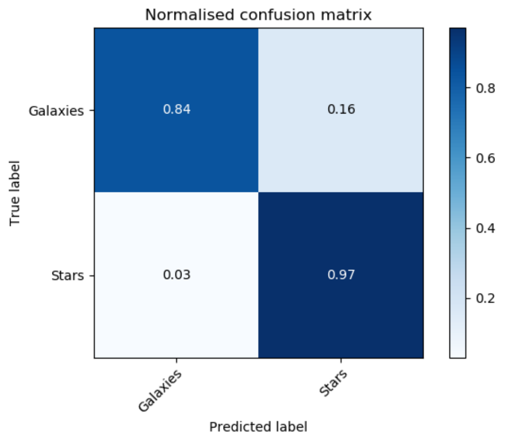
[9]https://classic.sdss.org/dr7/instruments/technicalPaper/index.html

would wrongly display mis-classified images, that would potentially affect the accuracy of the model. It would therefore be beneficial to employ either a second catalogue to match extended objects to, or measure some appropriate distinguishing features to validate the matching to GZ2.

5.2 The Accuracy paradox

The goal of a machine learning algorithm is to maximise the predictive accuracy on unseen test data, rather than the training data (Dietterich 1995). The use of the accuracy metric allows for simple, intuitive information of the performance of the model through measurements of the ratio of correct classifications to total classifications (Equation 3). While certainly a useful metric to employ, using it as the sole metric can be misleading in a classification problem. A model with a high accuracy does not necessarily translate to a model with better classifier performance (Valverde-Albacete and Peláez-Moreno 2014), which is known as the accuracy paradox.

The accuracy paradox can be demonstrated on our model when testing with unbalanced test sets. Highly unbalanced data can be found in several areas of nature (Valverde-Albacete and Peláez-Moreno 2014), with the proportions of galaxies to stars being a relevant example. On average, the source detection algorithm described in Section 2.2 detected > 40 times more non-extended than extended objects per image. This imbalance was accounted for during training through augmenting the extended sources to produce balanced training sets. However, when applying this balanced model to unbalanced test data, the classifier was less successful, as shown by Figure 14.



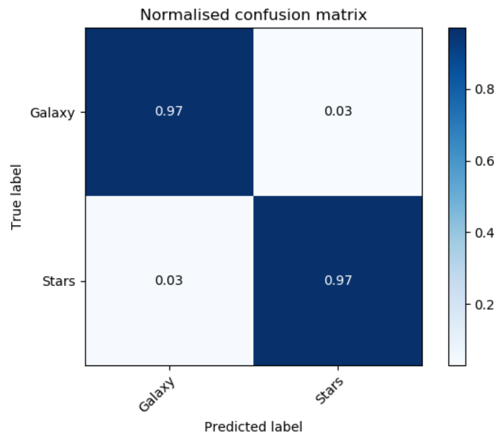


Figure 14: *Left* Normalised confusion matrix showing the performance of the model on unbalanced test sets. The test dataset consisted of 50 extended objects and 2000 non-extended objects, which was the typical ratio of sources found in the GOTO images. *Right* Normalised confusion matrix when applying balanced test sets for extended and non-extended sources.

The unbalanced test data appears to have achieved results that are comparable to the balanced test data, with both achieving high accuracies of $\sim 97\%$. However, when introducing more suitable metrics the extent of the accuracy paradox can be observed. Common metrics used throughout ML applications are precision and recall. Precision gives a measure of the proportion of correct identifications, and recall gives a measure of the correctly identified positive classes and are both shown in Equation 5, where again TP = true positives, TN = true negatives, FP = false positives, FN = false negatives[10].

$$Precision = \frac{TP}{TP + FP}, Recall = \frac{TP}{TP + FN} \tag{5}$$

For a perfect model the precision and recall would both equal 1, as FP and FN would equal zero. It is useful to measure both of these metrics to evaluate the performance of a model, however it is often seen that improving one metric acts to decrease the other. It is therefore common to evaluate the F_1 score of an object, which combines both precision and recall (Aniyan and Thorat 2017), shown in Equation 6, where for a perfect model $F_1 = 100\%$.

[10]Equation reference: `https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall`

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{6}$$

When calculating the F_1 score for the test sets shown in Figure 14 there was a large disparity in the values obtained. For the unbalanced test set the F_1 score was 55.2%, compared to the 97% for the balanced set. Table 3 shows that the unbalanced test set had a very low precision at just 41.1%. While only displaying false positives of 3% in the *left* of Figure 14, as this was on the larger non-extended dataset, the absolute number of false positives was actually 60. Therefore, despite being a small proportion of the non-extended dataset, the unbalanced test sets means this still dominated over the relatively high fraction of correctly identified extended sources. This shows that the accuracy paradox is present in our model, as the classifier is poor due to the low precision despite a seemingly excellent accuracy.

Table 3: Calculations of the Accuracy, Precision, Recall, and F_1 score when considering the values outlined in Figure 14, demonstrating the effect of the accuracy paradox when using unbalanced test sets.

dataset	**Number of objects** (non-extended, extended)	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	F_1 **score (%)**
Unbalanced	(2000, 50)	96.7	41.1	84	55.2
Balanced	(10,000, 10,000)	97	97	97	97

The model presented is still highly accurate, with a high F_1 score when considering balanced test sets. However, in a typical image from any telescope it is extremely unlikely that the number of extended and non-extended objects would show this balance. Therefore in an astronomical application the model is not useful. The F_1 score would have been a much more suitable metric to include during testing, rather than solely accuracy, as it gives a much better measure of the performance of the algorithm on more realistic test data (Shung 2018) Therefore, future studies should focus on maximising the F_1 score, rather than solely the accuracy, to obtain more useful results in an astronomical context.

6 Conclusion

In this project we have presented results from a machine learning model that performed binary classifications of extended/non-extended and featured/non-featured objects using data provided by GOTO, with positional matching to pre-label extended sources from Galaxy Zoo. The model employed a deep convolutional neural network (ConvNet) due to the lack of input features required, as the model learns directly from the image themselves as opposed to implementing selected features. ConvNets have been repeatedly shown to produce high accuracies within an astronomical context. Sources were detected, and isolated from larger images using a simple Python algorithm to obtain a representative sample of each class. The initial $\sim 6,000$ detected extended objects were then augmented through rotations to produce equal training sets of $\sim 62,000$ sources in both the extended/non-extended classes, with an equal number of featured/non-featured objects. The model was then tested on a separate, unseen test set. Both models for extended/non-extended and featured/non-featured objects display strong accuracies of $> 90\%$ during training and testing. These results show that high classification accuracies are achievable through deep neural networks and provide large numbers of classifications significantly faster than manual classification.

For future work it would be desirable to include a more suitable metric to measure the performance of the model. The sole use of accuracy presented problems when testing the model on unbalanced test sets. Unbalanced test sets were shown to have a low precision ($< 50\%$). This is a major drawback to the model, chiefly because unbalanced test sets are extremely common within most fields, especially astronomy. If lots of false positives were present in the test data, these would need manual confirmation of the morphological classes. This would be highly time consuming, and goes against the philosophy of machine learning to remove the need for human classifications. The accuracy of the model was also limited by data with a poor resolution that had multiple defects present. Attempts were made to improve the quality of the input data through crude geometrical arguments, with limited success. It is hypothesised that specific features of the defects could be determined during source detection, which would allow them to be removed from the input data entirely. At present, there is no way to improve the resolution of the images without improvements made to GOTO.

Some of the issues identified are related to the specific use of ConvNets. One of the main requirements, and disadvantages, of any deep learning process is the requirement for a large number of samples. For deep learning neural networks the accuracy is coupled to the size of the dataset. The size of the detected sample was increased through augmenting the dataset, however the techniques used have a strong dependence on pre-processing. It is likely that if applied to a future GOTO data release the pre-processing steps applied here will need to be adapted to obtain similar accuracy levels. As the performance of the model is dependent on the size of the dataset, it is clear that future work should include a larger dataset. Effort was placed into reducing overfitting, allowing for training using large numbers of images, so the classification accuracy will likely be increased by using even larger datasets. The current dataset was also limited by the requirement for matching to the Galaxy Zoo catalogue. If effort can be placed to remove this requirement it is likely that very large ($\gtrsim 100,000$) datasets could be employed in the network.

The model presented can be adapted for future work within the GOTO pipeline. Theoretically, as the model is fully trained to a high accuracy it will not require retraining. However, if there are improvements to the resolution of GOTO images, or inclusion of a more suitable metric it is likely that retraining will be required. The model was trained on consumer hardware with source code that will be made publicly available. The high classification accuracies produced on large-scale data continues to show the viability of machine learning within astronomy and will be especially important as data rates continue to increase with more new facilities.

References

Abbott, B. P. et al., *GW170817: Observation of Gravitational Waves from a Binary Neutron Star Inspiral*, Phys. Rev. Lett. 119, 161101, (**2017**)

Aniyan, A. K. and Thorat K., *Classifying Radio Galaxies with Convolutional Neural Networks*, arXiv:1705:03413v1 [astro-ph.IM], (**2017**)

Banerji, M. et al., *Galaxy Zoo: reproducing galaxy morphologies via machine learning*, Mon. Not. R. Astron. Soc. 406, 352-353 (**2010**)

Bilbao, I. and Bilbao, J., *Overfitting problem and the over-training in the era of data: Particularly for Artificial Neural Networks*, 2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS), Cairo, pp. 173-177, (**2017**)

Calleja, J. and Fuentes. O, *Machine Learning and image analysis for morphological galaxy classification*, Mon. Not. R. Astron. Soc. 349, 87-93 (**2004**)

Dieleman, S. et al., *Rotation-invariant convolutional neural networks for galaxy morphology prediction*, Mon. Not. R. Astron. Soc. 000, 1–20 (**2014**)

Dietterich, T., *Overfitting and Undercomputing in Machine Learning*, Oregon State University, (**1995**)

Domingos, P., *A few useful things to know about Machine Learning*, Communications of the ACM, Vol. 55 Issue 10, 78-87, (**2012**)

Eyono, R. H., *Feature Extraction and Selection of Optical Galaxy Data*, University of Cape Town (**2017**)

Fadely, R. et al., *Star-Galaxy Classification in Multi-band Optical Imaging*, The Astrophysical Journal, 760, 1, (**2012**)

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

The Gravitational-Wave Optical Transient Observer, WWW document `https://goto-observatory.org`, (**2018**)

Henrion, M. et al., *A Bayesian approach to star–galaxy classification*, Monthly Notices of the Royal Astronomical Society, 412, 4, (**2011**)

Huertas-Company, M. et al., *Measuring galaxy morphology at z ¿ 1. I - calibration of automated proxies*, arXiv:1406.1175v1 [astro-ph.GA], (**2014**)

Huertas-Company, M. et al., *A Catalog of Visual-like Morphologies in the 5 CANDELS Fields Using Deep Learning*, arXiv:1509.05429v1 [astro-ph.GA], (**2015**)

Hocking, A. et al., *An Automatic taxonomy of galaxy morphology using unsupervised machine learning*, arXiv:1709.05834v1 [astro-ph.IM], (**2017**)

Howard, E. M., *Machine learning algorithms in Astronomy*, Astronomical Data Analysis Software and Systems XXV, 512, 245-248 (**2017**)

Jaitley, U., *Why Data Normalization is necessary for Machine Learning Models*, WWW document, `https://medium.com/@urvashilluniya/why-data-normalization-is-necessary-for-machine-learning-models-681b65a05029`, (**2018**)

Kim, E. J. and Brunner, R. J., *Star-Galaxy Classification Using Deep Convolutional Neural Networks*, arXiv:1608.04369 [astro.ph.IM], (**2016**)

Krizhevsky, A. et al., *Advances in neural information processing systems*, 1097 - 1105, (**2012**)

LeCun, Y. et al., *Deep Learning*, Nature, 521, 436-444, (**2015**)

Lintott, C. et al., *Galaxy Zoo: Morphologies derived from visual inspection of galaxies from the Sloan Digital Sky Survey*, arXiv:0804.4483v1 [astro-ph], (**2008**)

The Sloan Digital Sky Survey: Mapping the Universe, WWW document `https://www.sdss.org/`, (**2018**)

Shung, K. P., *Accuracy, Precision, Recall or F1?*, WWW document, `https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9`, (**2018**)

Uodfina, U., *Basic Overview of Convolutional Neural Networks (CNN)*, WWW document, `https://medium.com/@udemeudofia01/basic-overview-of-convolutional-neural-network-cnn-4fcc7dbb4f17`, (**2018**)

Valverde-Albacete, F. J. and Peláez-Moreno, C., *100% Classification Accuracy Considered Harmful: The Normalized Information Transfer Factor Explains the Accuracy Paradox*, Plos One, `https://doi.org/10.1371/journal.pone.0084217`, (**2014**)

Willett, K., et al., *Galaxy Zoo 2: detailed morphological classifications for 304,122 galaxies from the Sloan Digital Sky Survey*, Mon. Not. R. Astron. Soc. 000, 1-29, (**2013**)

Appendix

Appendix A: Project Plan

The next phase of the project is to implement the data into the chosen ConvNet. Before this however, there are still several steps which are required to be taken, the approximate time frames of which are summarised in Figure 13. An important aspect is to gain some practical experience of implementing a ConvNet in Python. This can be achieved through the plethora of online tutorials available, and will be the focus over the Christmas break. The next step that must be taken is to continue to test the methods of data reduction which are discussed in **3.4**. If this data reduction cannot produce robust, reliable results within a short enough time frame, then this section will have to be abandoned in favour of allowing sufficient time to perform the various ML steps. The data reduction is not essential for an accurate ML module, and was designed more as a means to increase efficiency.

The primary focus of the work will be concerned with building a large enough training set to effectively train the ConvNet. A likely candidate for a training set will be Galaxy Zoo (GZ). Like GOTO, GZ is also a wide area survey that covers a large area of the sky. This will mean that it will provide a similar dataset, unlike if a deep field survey, like CANDELS, was employed. It will also be important to cross-reference the sources (using RA and Dec calculations from creating the cut-outs), to ensure that the training set covers a similar patch of sky to the GOTO data. Careful considerations into overfitting will have to be considered along this, and so a training set that is equally weighted between the broad morphological classes of galaxies, and stars will have to be implemented. Once a training set that is sufficiently large has been created, the GOTO images can then be inputted into the ConvNet to begin classification.

Classification through use of the ConvNet will initially be focused on star-galaxy separation. If the data reduction techniques prove successful, this will be a less vital stage, however it is still possible that the simple algorithms employed will not cut all the stars from the sample. ConvNets were chosen due to their ability to exploit transfer learning, which will allow us to repurpose an existing ConvNet which will be trained on the training data created. Therefore, it will be necessary to review the various source codes of available ConvNets to test which provides the most accurate results. Once the ConvNets have been narrowed down to a single algorithm, it will then be possible to input the GOTO data and begin performing classification of stars and galaxies initially. Due to the differences in image quality on each CCD, it is likely that the model will need to be individually trained on each CCD separately. If time permits, once the star-galaxy separation has been proven to be an effective classifier based on morphology, it will then be the aim to classify the galaxies into broad morphological categories, such as late and early type.

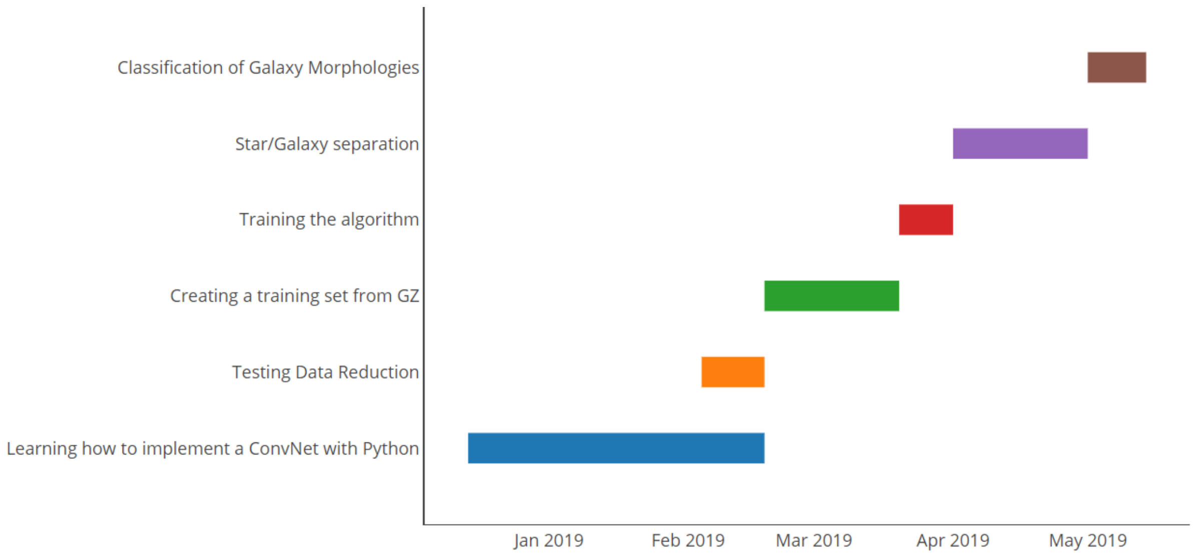


Figure 15: Gantt chart showing the approximate time frames of the different tasks which are required to be completed in future work in this study.

Appendix B: Literature Review

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

150194303

March 23, 2019

Abstract

Morphological classification of astronomical objects, particularly galaxies, is a vital tool for astronomers. Recent advances in telescope technologies have meant that image data is being received at a rate significantly faster than can be classified by experts. The implementation of machine learning in an astronomical context potentially provides a solution to the large data rates. Here, multiple methods of machine learning have been described, with their uses in astronomy being discussed. The current study revolves around the GOTO telescopes, and classifying the astronomical objects in the images produced, through morphology alone. Currently, work has revolved around preparing the images to be inputs into the machine learning algorithm of choice, which will likely be a ConvNet. Source detection algorithms have been developed, using modules within Python. It was also proposed to use a simple algorithm to separate stars and galaxies, through the use of ellipticity and extent parameters, which has shown some success. However, further testing will be required on this splitting due to the varying quality of images sourced from GOTO. Finally, 50 x 50 pixel cutouts of the detected sources were created, which will ultimately become the inputs for the ConvNet. Future work will be focused around building a training set, likely using the Galaxy Zoo catalogues, and then performing the morphological classification using the ConvNet.

1 Introduction

Among the most fundamental properties of an astronomical object is its shape and structure, known as morphology. It is possible to distinguish objects in the night sky purely based on morphological differences. Stars appear as unresolved point sources, while objects such as galaxies and nebulae appear as extended objects with distinct structures. Therefore, the measurement of morphology is a crucial tool in astronomy for distinguishing between objects in the night sky.

Early astronomers were able to classify astronomical objects by eye, or through the use of rudimentary telescopes. Advancements in telescopes throughout the early 20th century allowed for significantly more detail, and crucially, more objects to be detected. The majority of classifications and catalogues shifted to being compiled by individuals, small teams of astronomers, or 'citizen scientists' in projects such as Galaxy Zoo (Lintott 2008). This approach is becoming less viable as astronomy moves into the 'petabyte' regime (Hocking 2017). The advent of more advanced telescopes has enabled astronomy to become one of the first disciplines to have to deal with this extensive amount of data, shifting from a traditionally data-poor science, to a highly data-intensive discipline (Howard 2015).

The issue surrounding the significant influx of data is especially apparent when considering that it can take several years for an expert to classify only ∼10,000 objects (Calleja and Fuentes 2004). Wide-area surveys only exemplify these problems. Surveys such as the Gravitational-wave Optical Transient Observatory (GOTO 2018) is currently capable of imaging 20 square degrees of the night sky, with plans to increase to 40 square degrees in summer 2018. The GOTO telescope can produce images containing thousands of potential sources to be classified. Even larger surveys such as the Sloan Digital Sky Survey (SDSS 2018) has reported to have catalogued over 1.2 billion objects across a third of the night sky (Gauthier 2016) through use of its 2.5m telescope. The vast amount of data being received shows that it is clear that robust, automated analysis of the data is required to keep up with the demand.

While it is possible to classify all astronomical objects based on morphology, the first step often taken in wide-area surveys is to separate extended objects, typically galaxies, from point sources, and then separating galaxies into morphological classes. The crucial distinction in morphology between galaxies provided the basis to Hubble's Tuning Fork, developed in 1926. The tuning fork has remained as a standard for galaxy classification, with only small revisions made by De Vaucouleurs (De Vaucouleurs

1959) to include ringed, and lensed spirals. Morphology remains one of the primary methods of extracting the physical information of galaxies (Gauthier 2016), and relates directly to structural and dynamical properties (Perez-Carrasco 2018). Categorising galaxies purely based on morphology also provides the fundamental distinction between early and late type galaxies (Lintott 2008).

The problem concerning classifying galaxies from the large amount of data being acquired has already produced novel solutions, such as Galaxy Zoo (GZ). GZ is a web-based project that has classified ~ 50 million objects[1] through harnessing the power of crowd-sourcing. This study has proved enormously powerful for studies of galaxy formation, and evolution (Banerji 2010). Labelled catalogues produced by GZ have been shown by Lintott (2008) to agree with professional astronomers to accuracies of $> 90\%$, with the largest catalogue containing $> 300,000$ galaxies at more than 5σ level of confidence. However, this project was still unable to classify objects at a sufficient rate, and this problem will only worsen as data rates increase.

A promising solution that has been developed is machine learning (ML). ML has been applied to a wealth of applications within astronomy, most notably among which for this discussion is star-galaxy classification based on morphology. These methods have been largely successful, with papers such as Aniyan and Thorat (2017) reporting an accuracy of 88% across three morphological classifications of radio galaxies. This level of accuracy is comparable to manual classification, while being significantly faster. Machine learning techniques will likely become a staple of future astronomical classification due to their speed, high accuracy, and ability to improve with more advanced computing techniques.

2 Machine Learning

In general, ML gives computers the ability to learn and perform tasks without being explicitly programmed to do so (Eyono 2017). When applied to image classification, this can allow for automatic, and probabilistic classification in a much shorter time span than compared to human identification. Machine learning is generally split into supervised, and unsupervised learning. Supervised learning uses a known, labelled training set, and targets a known output. Unsupervised learning does not require any known training sets, and no expected classes are previously known (Howard 2015).

A key problem to overcome in supervised ML algorithms is the concept of *overfitting*. While particularly prevalent in neural networks (see **2.2**), overfitting can be seen in all supervised learning algorithms. At its core overfitting refers to an algorithm which has modelled the training data 'too well'. In this case the algorithm has learned the detail and noise in the training data to such an extent that it begins to negatively affect the performance of the algorithm when presented with new data[2]. This can result in an algorithm being 100% accurate on training data, but only 50% accurate on test data as an example (Domingos 2012). Methods of combating overfitting are particularly important for creating a high quality algorithm.

Further splitting of categories of ML can be done by considering the method of how the algorithm learns. This results in two further classes: shallow, and deep learning (Aniyan and Thorat 2017). Shallow learning learns from 'features' of the data. Features are the input variable which is used in making predictions, and are extracted from the observational data. Deep learning however, learns directly from the raw data (Bengio 2009), and so does not rely on any physical interpretation of the data. Key examples of these two methods of learning are presented in the following sections.

2.1 Shallow Learning

Shallow learning, also referred to as classical learning (Aniyan and Thorat 2017), is a branch of ML that learns from features extracted from the raw data. Hence, shallow learning can also be referred to as feature-based learning. The features that are chosen are representative of specific physical properties of the system, and so the efficiency of the algorithm is highly dependent on the quality of the features used (Aniyan and Thorat 2017).

Feature extraction is a highly important part of shallow learning techniques, and as such multiple extraction algorithms have been developed, an example of which is Principal Component Analysis (PCA). Through PCA the components of an image that explain the maximum amount of variance possible within the image are extracted (Calleja and Fuentes 2004). These components represent the features that are then used an inputs into a ML module. An advantage of PCA is the number of extracted features can be reduced from potentially thousands, to just a few hundred or less (Gauthier 2016). This advantage

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

[2]Sourced from https://towardsdatascience.com/deep-learning-overfitting-846bf5b35e24

is shown in Figure 1, where the galaxies and their structure have been clearly reproduced using just 25 features. This feature reduction is referred to as dimensionality reduction (Eyono 2017). Therefore PCA has the ability to not only extract relevant features, but also acts to increase the efficiency of the ML module.

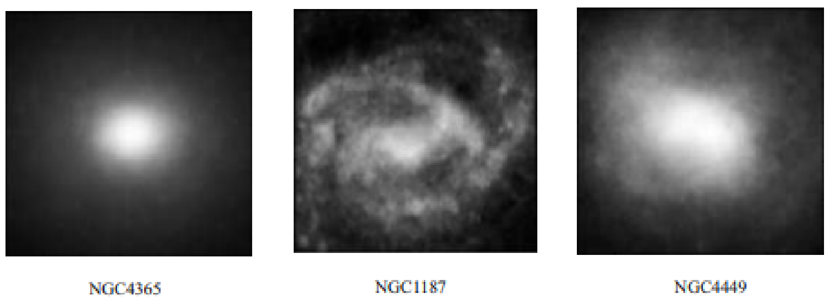


Figure 1: An example of galaxy images being reconstructed by PCA from Calleja and Fuentes (2004). This image uses 25 principal components, which represents 85% of the information in the original image.

Once feature extraction has been performed, there is a wealth of shallow learning algorithms which are available for use. The potential algorithms used have a wide range of complexity. k Nearest Neighbour algorithms are perhaps the simplest ML method - see Li (2008), and use Euclidean distance between the data and the training set to obtain classification. Decision trees (see **2.1.1**) use a tree-like model of decisions to obtain a classification, through branching nodes. Another widely employed method is the Support Vector Machine (see **2.1.2**), which constructs hyperplanes between the data, to a potentially infinite amount of dimensions. The wide range of algorithms on offer show that it is important to select the algorithm which is most suitable for the task.

2.1.1 Decision Trees

The structure of a tree has proven to be an important architecture in ML. Decision trees (DT) have been in use since the 1970s - see Ball (2006) - and have been widely used for classification. DT have a structure predictably analogous to trees, where it begins in a root node, with branching decisions built up from the feature selection. Each possible outcome of a test results in a branch from the node (Vasconcellos 2011). The process of creating branches is then repeated until the final node, the 'leaf'. This structure is shown in Figure 2 using an astronomical example (Vasconcellos 2011).

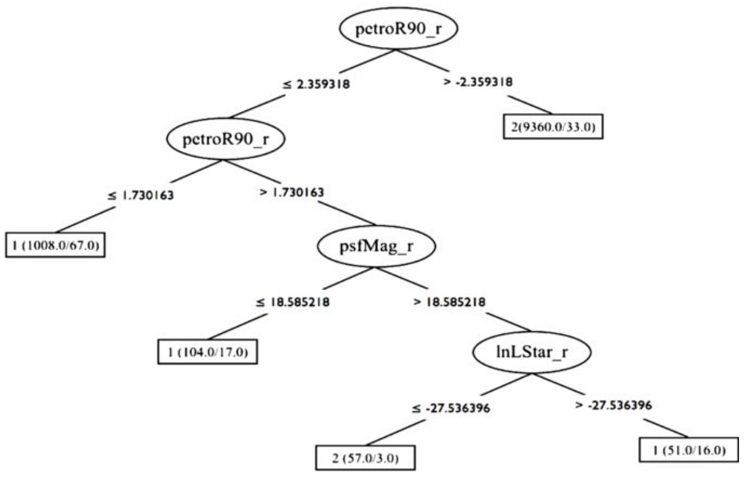


Figure 2: Example of a simple decision tree, trained using 50,000 objects using a spectroscopic sample - image taken from Vasconcellos (2011).

2.1.2 Support Vector Machines

Support Vector Machines (SVM) are another example of a simple feature-based algorithm, first introduced by Cortes (1995), that have been successfully employed to solve classification problems. The goal of a SVM is to separate classes of data-points. This is done by constructing a plane (hyperplane) that has the maximum distance between the data points (Huertas-Company 2010), which is visualised in Figure 3. The simplest implementation is when the hyperplane is in feature space, and so SVM are often described as N-dimensional, where N is the number of features.

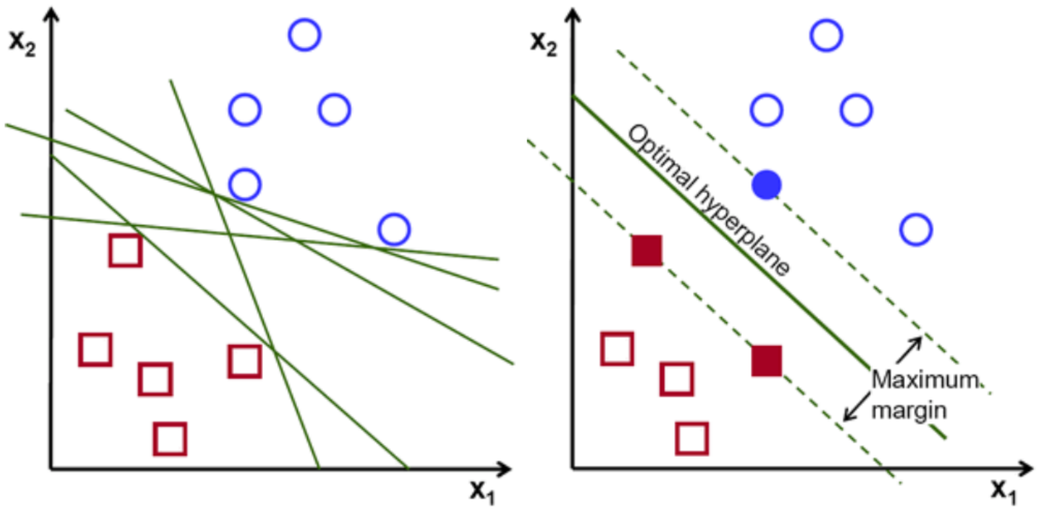


Figure 3: (*left*) Visualisation of potential hyperplanes constructed between two sets of data, where (*right*) shows the optimal hyperplane calculated by the SVM[4].

2.2 Deep Learning

Deep learning is a branch of ML where the algorithm learns directly from the raw data, such as images (Bengio 2009). As the algorithms learn directly from the images, they can also be described as non-feature based. In general, deep learning, primarily in the form of neural networks, are more computationally expensive, and take more development time compared to the shallow learning counterparts. However, deep learning algorithms have outperformed all other shallow algorithms by significant margins (Aniyan and Thorat 2017), specifically in the field of object recognition (Krizhevsky 2012).

A crucial feature of deep neural networks is transfer learning (Yosinski 2014), which allows for an existing neural network to be repurposed for a new application with a much smaller data set (Yosinski 2014). In astronomical applications, data sets are difficult to obtain for different applications, and so transfer learning can be exploited to train an existing deep neural network for a different classification problem (Aniyan and Thorat 2017). Transfer learning also improves the existing model without requiring to retrain from scratch, which is the case in shallow ML algorithms.

A key family of deep learning algorithms are neural networks, namely Artificial Neural Networks (**2.2.1**), and Convolutional Neural Networks (**2.2.2**). As stated in section **2**, overfitting is a common problem encountered in developing an effective neural network. Overfitting is common in neural networks due to the large number of parameters that can be learned from the image data. It is common for a network to have millions of learned parameters, as Krizhevsky (2012) reported 60 million parameters, and Kim and Brunner (2016) reporting 11 million parameters for only 4×10^4 images.

There are generally two main methods employed which can reduce overfitting in neural networks. While not exclusive to neural networks, the most common method is to artificially enlarge the data-set by applying random perturbations (Perez-Carrasco 2018). Common perturbations introduced are: rotation, reflection, and translation of the images (Kim and Brunner 2016). An additional advantage to augmenting the data size is the resulting model is more highly rotationally invariant, and adds very little computational cost (Krizhevsky 2012, Kim and Brunner 2016). The second technique commonly

[4]Image from `https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca4`

used is ‘Dropout’ (Kim and Brunner 2016). This consists of randomly setting the output of a neuron to zero, with a probability of 0.5 (Krizhevsky 2012). Since these neurons can be removed, it cannot rely on the presence of the other neurons in the network, and so must learn more robust features (Kim and Brunner 2016). These two methods in combination are both effective ways of combating overfitting in neural networks.

2.2.1 Artificial Neural Networks

Human brains are able to interpret the context of situations much more efficiently than computers can. While not exclusive to deep learning, Artificial Neural Networks (ANN) can approximate functions from a set of inputs, producing a single output. ANNs are typically structured in multiple layers containing neurons that are represented by a mathematical function, typically a dot product. Each neuron is connected to neurons of the previous, and later layers (Kim and Brunner 2016). By convention, all layers except the input and output layers are referred to as the hidden layers (Kim and Brunner 2016). The layered structure shown in Figure 4 is inspired by biological neurons (Aniyan and Thorat 2017), and designed to replicate the learning process in humans.

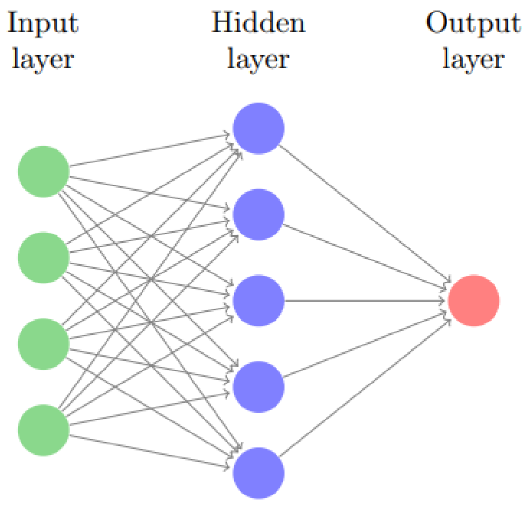


Figure 4: A schematic diagram from Kim and Brunner (2016) showing an ANN with one hidden layer. As this diagram only contains one hidden layer this would in fact be described as a shallow network.

2.2.2 Convolutional Neural Networks

Convolutional Neural Networks (ConvNets) are conceptually, and structurally similar to ANN. ConvNets also follow the structure outlined in Figure 4, however there are some subtle mathematical differences which distinguish ConvNets from ANN. In all neural networks the outputs of each neuron is *weighted*, and while in ANN’s the weighting is a single value for each neuron, in ConvNets the weighting is a vector called filters (Kim and Brunner 2016). The output of each neuron is therefore a convolution of the inputs and the weights, instead of the dot product used in ANN. In recent years ConvNets have achieved record-breaking results in image classification tasks (Kim and Brunner 2016), and have since been widely adopted for a variety of problems.

2.3 Machine Learning in Astronomy

There is a wealth of ML algorithms that have been successfully applied to astronomical problems. The task of classifying galaxies has been recognised to be well suited to ML for some time. For example, ANNs have been used as far back as 1992 to reproduce galaxy morphologies (Storrie-Lombardi 1992). Since then a number of techniques have been used for classifications, primarily employing supervised learning techniques. Unsupervised techniques can create novel solutions, such as in Hocking (2017) where the process to classify galaxies was previously unseen in astronomy. Unsupervised techniques also remove the need for expert classifications prior to performing any ML. However, they are less suitable for object classification as unsupervised techniques cluster the data together into groups that have no physical significance. Therefore, for this project only supervised techniques are considered, with their applications in astronomy discussed. However, it should be stressed that there is no single best algorithm, or a straight-forward way to decide on the optimal algorithm (Howard 2015).

2.3.1 Huertas-Company in astronomy

A prominent figure in the field of supervised ML in astronomy has been Huertas-Company (Huertas-Company (2009, 2010, 2014, 2015)). The papers discussed have featured both SVM classification using the galSVM code (Huertas-Company 2009, 2014), and ConvNets (Huertas-Company 2015). These papers used deep surveys at redshifts of $z > 1.5$, and so provide an ideal comparison between the two algorithms.

The accuracies reported in the SVM based study (Huertas-Company 2014), and ConvNet based study (Huertas-Company 2015) appear to suggest that ConvNets significantly outperform SVMs. Huertas-Company (2014) used a SVM with 7 parameters to separate early, and late type galaxies at $z > 1$. Whereas Huertas-Company (2015) used a ConvNet which was designed to mimic human perception in order to classify $\sim 50,000$ galaxies at a median redshift of z ~ 1.5. Both studies used the CANDELS deep field surveys as training sets. Huertas-Company (2015) reported misclassifications of $< 1\%$, compared to the maximum of $20 - 30\%$ misclassifications from Huertas-Company (2014). These results appear to suggest that, especially at higher redshifts, a ConvNet is more effective method of classifying galaxies.

While the results produced by Huertas-Company (2015) provide a strong argument in favour of ConvNets for this study, the GOTO data that will be used is not specific to high redshifts. The GOTO telescopes are designed to search for gravitational wave events (GOTO 2018), and so are not specifically searching the high redshift regime seen in Huertas-Company (2014, 2015). However, in principle training a ConvNet using wide area surveys is still suggested to be more effective by these two studies.

2.3.2 Other examples of ConvNets in Astronomy

ConvNets have also been directly compared to other algorithms in an astronomical context by Kim and Brunner (2016). Kim and Brunner (2016) employed a ConvNet to perform star-galaxy classification using the SDSS as a training set. This training set is more comparable to the data obtained by GOTO, as they are both wide-area surveys covering large portions of the sky. Kim and Brunner (2016) compared a ConvNet to a species of DT, a Tree for Probabilistic Classifications (TPC). Two TPC's were created, one trained only with photometric data (TPC$_{phot}$), and the other trained using morphological properties (TPC$_{morph}$). This study concluded that the ConvNet and TPC$_{morph}$ both outperformed TPC$_{phot}$ in all metrics used, which is shown in Figure 5. This reinforces the suggestion that morphology is a key distinction between stars and galaxies. This result is crucial for this project, as the GOTO data being used is single band, and therefore primarily contains morphological data.

Kim and Brunner (2016) also provides a direct comparison between shallow and deep learning algorithms. In this study the non-feature based ConvNet performed slightly worse than than the feature based TPC$_{morph}$. However, both produced very high accuracies of $\sim 99\%$ across multiple metrics, with minimal loss. It is of note that not only did the TPC$_{morph}$ perform better at fainter magnitudes, but it was also reported to outperform the star-galaxy classifier within SDSS. However, the SDSS catalogue provides a concentration parameter that is highly optimised for star-galaxy classification. This included parameter was potentially instrumental in TPC$_{morph}$ outperforming the ConvNet. Further evidence for this claim is shown by the results that Kim and Brunner (2016) obtained when using the Canada-France-Hawaii Telescope Lensing Survey (CFHTLenS) data. This catalogue did not provide any parameters specifically provided for star-galaxy classification, consequently the ConvNet outperformed the TPC$_{morph}$. This comparison does show that both shallow and deep learning algorithms can produce highly accurate results, and which will perform better is highly dependent on the data provided to train the algorithms on.

ConvNets have also been successfully applied to classifications of radio galaxies. As previously discussed, Aniyan and Thorat (2017) achieved an accuracy of 88% across three morphological classes of radio galaxies. This accuracy is comparable with manual classification, but has the added benefit of being significantly faster. While an 88% accuracy is certainly an impressive feat, it could become problematic when applied to very-large datasets. If a dataset is comprised of billions of sources, misclassification of $\sim 12\%$ could result in hundreds of millions of sources being incorrectly classified. This could potentially require an expert to verify the classifications, which reduces the time saved by employing a ML algorithm. However, currently ConvNets have been shown by Aniyan and Thorat (2017) to provide successful classification of radio galaxies.

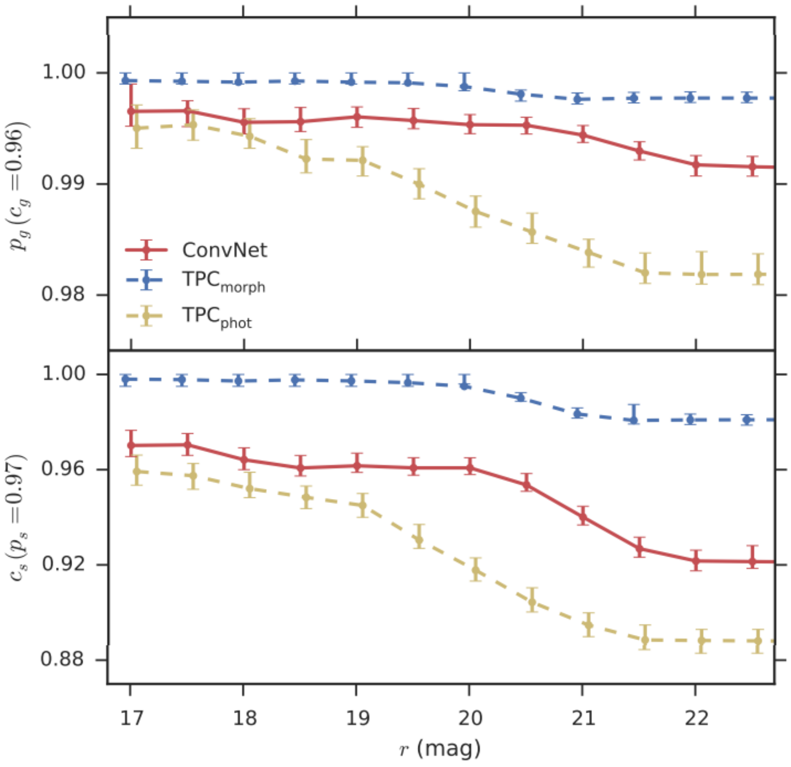


Figure 5: Graph of galaxy precision (p_g) and star sensitivity (c_s) as a function of r-band magnitudes on the SDSS data set. From this plot it is clear to see that, while all results produce highly accurate results (> 0.90), the TPC$_{morph}$ outperforms the ConvNet at all magnitudes, and TPC$_{phot}$ is outclassed by both other methods at all magnitudes. Image taken from Kim and Brunner (2016)

2.3.3 Artificial Neural Networks in Astronomy

While ConvNets have provided record-breaking classifications, the importance of ANN in astronomy should also be stated. A successful implementation of an ANN in astronomy can be seen by Banerji (2010), who attempted to recreate the Galaxy Zoo classifications using an ANN. This was largely successful as Banerji (2010) reported accuracies of 87% for early type, and 86% for spiral galaxies, and 95% agreement for point sources with the human classification by Galaxy Zoo. Banerji (2010) also reported that using certain features were unable to distinguish between point sources and early type galaxies. It was also found that using a bright sample to train the ANN, then performing morphological classifications on fainter samples results in $\sim$ 90% agreement between human classifications in all three morphological classes. As discussed for Aniyan and Thorat (2017) accuracies of $86-87\%$ are impressive, yet can prove problematic when applied to the largest data sets. The fact that galaxy classification had accuracies $8-9\%$ lower than for point sources, suggests that ML is still not quite as good as humans for classifying the subtle differences between galaxy classes. Yet the ANN was still able to produce reliable morphological classifications, that are comparable to results produced by ConvNets in previous studies.

A further example of an ANN being successfully implemented in astronomy is Calleja and Fuentes (2004). Here an ANN was compared to a locally weighted regression algorithm (LWR), with data taken from the NGC catalogue[5]. When considering two classes of galaxy (elliptical and spiral) the ANN reported accuracies of 93.82%. However, in the same class the LWR algorithm outperformed the ANN, achieving accuracies of 95.11% while also being faster than the ANN. However, this study only used 293 galaxies for classification. In general, ANNs improve with more data added, and so it is likely that given a much larger dataset the ANN would potentially outperform the LWR.

It has been shown that multiple ML algorithms have been successfully applied to astronomical problems. The key algorithms presented are SVM, ANN, and ConvNets, all of which are capable of providing

[5]`http://www.apsky.org/ngc/ngc.html`

classifications that are $\sim 90\%$ accurate. While comparisons between algorithms have shown that one algorithm performed better than another for that specific study, there is no one best general algorithm to be applied to astronomy. Choosing which algorithm to employ is therefore highly dependent on the sample data used to train the algorithm, as well as the classification data used.

3 Progress completed so far

A primary aim of the project for the first semester was to familiarise ourselves with the vast amount of literature associated with the problem of ML in astronomy. Being familiar with the surrounding literature available is especially crucial as there is no objectively 'best' algorithm which can be employed. The limitations and strengths of each algorithm in general, and in an astronomical context was a key aspect of research into the literature. Knowledge of these aspects of each commonly used algorithm allowed the wide range available to be reduced to a single algorithm.

Before any actual ML can take place, it was clear from the literature that several image analysis phases were crucial. While there are many external programs available to perform these actions, in this project all image analysis was performed within Python. This allowed for further development of our individual knowledge of the language, as well as streamlining the processes within a single file. The various steps employed for image analysis are as follows: source detection (**3.3**), data reduction (**3.4**), and creating cutouts (**3.5**). It should be noted that the initial testing of the algorithms created for each step were all tested on the same image[6], however data releases from GOTO allowed for testing on new, higher quality images to occur. All of these steps combined aided in producing a homogenous sample space, which is crucial for the ML algorithm.

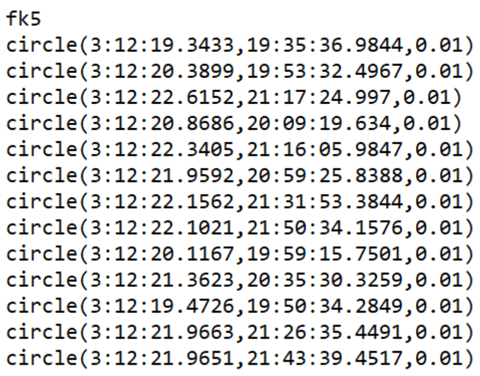
3.1 Choice of Algorithm

The literature provides a wide range of algorithms that have been proven to be successful at classifying images for a diverse range of astronomical contexts. The wealth of algorithms available required careful consideration of their individual strengths and weaknesses. From the methods presented, a deep learning ConvNet is likely to be the algorithm used in further work. ConvNets are considered to be the optimal algorithm for this study, as they have been proven to provide record-breaking results in the field of image classification (Kim and Brunner 2016), and have shown multiple successes within astronomy (Aniyan and Thorat 2017, Huertas-Company 2015, Kim and Brunner 2016). The use of ConvNets will also allow the exploitation of transfer learning, allowing us to circumvent the extensive development time associated with deep learning approaches. ConvNets therefore provide the ideal algorithm for the constraints on this study.

3.2 Region Files

A vital test of all methods employed was through visual inspection of the outputs. Visual inspection allows for clear, fast analysis particularly for the source detection, and data reduction phases. This was done through the creation of region files in Python, which can then be imported into SAOImageDS9[7] (DS9) to easily view detected sources, along with specific properties, such as ellipticity. An example of the format of the region file within Python, and the output when viewed within DS9 is shown in Figure 6. Visual inspection in general proved invaluable when testing different source detection, and data reduction methods, which was significantly aided through the use of region files.

[6]calexp-2-133758.fits
[7]`http://ds9.si.edu/site/Home.html`



```
fk5
circle(3:12:19.3433,19:35:36.9844,0.01)
circle(3:12:20.3899,19:53:32.4967,0.01)
circle(3:12:22.6152,21:17:24.997,0.01)
circle(3:12:20.8686,20:09:19.634,0.01)
circle(3:12:22.3405,21:16:05.9847,0.01)
circle(3:12:21.9592,20:59:25.8388,0.01)
circle(3:12:22.1562,21:31:53.3844,0.01)
circle(3:12:22.1021,21:50:34.1576,0.01)
circle(3:12:20.1167,19:59:15.7501,0.01)
circle(3:12:21.3623,20:35:30.3259,0.01)
circle(3:12:19.4726,19:50:34.2849,0.01)
circle(3:12:21.9663,21:26:35.4491,0.01)
circle(3:12:21.9651,21:43:39.4517,0.01)
```

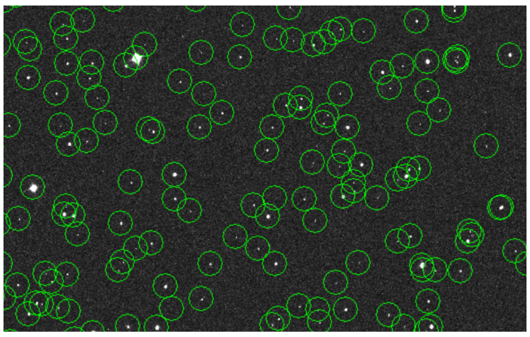


Figure 6: (*Left*) An example of a region file created within python. The format is stating the coordinate system used, in this case fk5, then the shape of the region (circle in this case), then right ascension, and declination in hour angle coordinates, then the radius of the circle in arcseconds. (*Right*) shows the region file being overlaid on top of a portion of the original image (calexp-4-67965.fits) within DS9, showing the number of detected sources.

3.3 Source Detection

Source detection is among the most fundamental step in preparing the data for ML. Without this, individual sources cannot be isolated from the larger image, and any ML algorithm employed would not be able to classify individual objects. As this step is so fundamental to the success of later ML, it was crucial that a robust, accurate source detection algorithm was developed.

Python has numerous built in functions which can act as source detectors. In an astronomical context, the `photutils` module[8] provided multiple methods that were tested in this project. The major source detection algorithms tested were `DAOStarFinder`[9], `find_peaks`[10], and `detect_sources`[11] in combination with `deblend_sources`[12]. Visual inspection of the detected sources showed that `DAOStarFinder` and `find_peaks` were significantly less effective than `detect_sources`. It was found that these two methods were not detecting sources which could easily be detected by eye. `Detect_sources` also had this problem, but to a much lesser extent, as `detect_sources` was able to identify $\sim$ 4850 sources in a single image, compared to `DAOStarFinder`'s $\sim$ 680 sources while using similar settings. `Detect_sources` had the added benefit of being able to be combined with the `deblend_sources` function. This allowed for multiple properties of the sources to easily be measured, which would later be used to reduce the data. From the methods tested, it was clear that the `detect_sources` function was the most effective on the GOTO data.

While `detect_sources` was the optimum algorithm of those tested, it was still far from ideal. As discussed earlier, this function was still not detecting objects which were easily detected through visual inspection. There were also multiple 'deblending errors' for individual sources, which could only be solved by editing the functions source code. However, once tested on the most recent GOTO data releases, `detect_sources` achieved much more robust source detection, with the frequency of deblending errors reduced, even before editing the source code.

3.4 Data Reduction

While already faster than manual classification, the speeds of ML algorithms can generally be improved through inputting less data. The GOTO data is likely to be mostly consisting of stars (proven through visual inspection) with significantly less galaxies expected in each image. Stars are generally less morphologically interesting compared to galaxies, shown by the lack of morphological classification within the literature. The total data can therefore be reduced by introducing a simple algorithm to separate stars and galaxies, where the galaxies advance to the ML module.

[8]https://media.readthedocs.org/pdf/photutils/latest/photutils.pdf
[9]https://photutils.readthedocs.io/en/stable/api/photutils.DAOStarFinder.html
[10]https://photutils.readthedocs.io/en/stable/api/photutils.find_peaks.html
[11]https://photutils.readthedocs.io/en/stable/api/photutils.segmentation.detect_sources.html
[12]https://photutils.readthedocs.io/en/stable/api/photutils.segmentation.deblend_sources.html#photutils.segmentation.deblend_sources

Initially it was decided to take advantage of the extended nature of galaxies to separate them from stars. Extended objects such as galaxies will show a distinctly different ellipticity compared to point source objects like stars. The deblend function used during source detection allowed for measurements of the ellipticity (e), and multiple other properties, to be calculated. The ellipticity of the sources was then plotted across the image, as shown in Figure 8. While showing there are some distinct sources that are more elliptical than the background stars, there is also a clear smooth variation across the image. This was likely due to the Point Spread Function (PSF) being non-uniform across the image, a result later found to be an issue with the GOTO telescope's original optics. The effect of this issue is demonstrated by Figure 7. Before the separation of stars and galaxies could take place, it was clear that the variation in the PSF would have to be corrected for.

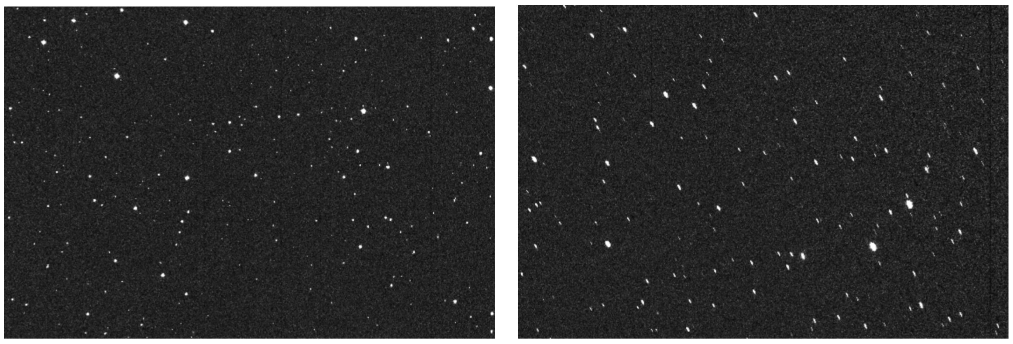


Figure 7: Demonstration of the defect in the optics of the GOTO telescope. *Left* shows a slice from the centre of the image, whereas *right* shows a slice from the right-hand edge of the image. It is clear that in *right* the objects that were observed have been elongated by some external factor. It is likely this is due to a non-linear PSF, as a result of defects in the early GOTO data releases.

In order to correct for the non-uniform PSF it was decided to model the ellipticity variation as a 2D polynomial using the astropy `Polynomial2D`[13] function, and a `LevMarLSQFitter`[14]. These functions allowed a generic 2D polynomial to be created, and then fit to the ellipticity data created, for a number of different powers. The results of the fit are shown in Figure 9, and visually appear to follow the variation shown in Figure 8. In order to correct for the PSF, the ellipticity values from Figure 8 were divided by the output values of the fit. The majority of objects in the image were expected to be stars, and so the normalised ellipticity values would cluster around 1, shown by Figure 10, with the outliers of the fit likely representing galaxies. As the majority of points were clustered around 1, the PSF has successfully been corrected for.

However, as with the source detection the methods described had several problems associated with them. A key problem is that ellipticity alone is a poor feature for separating stars and galaxies in such poor quality images. A more suitable feature to use would be extent, area, or orientation, as galaxies are inherently more extended compared to stars, which are point sources. Upon visual inspection of the 'highly elliptical' sources it was also seen that objects that could be easily classified as galaxies were not identified as elliptical. This was the case even when detecting spiral galaxies, which are inherently more elliptical. It was also the case that stars were being classified as elliptical objects, which is a clear problem with this method.

New data releases and improvements to the optics of the GOTO telescopes have since minimised these problems. Figure 11 shows that there is a less extreme variation of ellipticity across the image, when compared to Figure 9. This allows us to remove the ellipticity fitting stage, and purely search for extended, elliptical objects. It is also of note that the deblending errors described in **3.3** were significantly reduced in the new images. While certainly improved over the original images, there are still problems associated with the newer images. There is still distortion present in some of the CCD images, with CCD 4 showing the lowest distortion, through visual inspection. Coupled with this is an error with CCD 1, which has prevented any source detection being able to be performed on it.

[13]`http://docs.astropy.org/en/stable/api/astropy.modeling.polynomial.Polynomial2D.html`
[14]`http://docs.astropy.org/en/stable/api/astropy.modeling.fitting.LevMarLSQFitter.html`

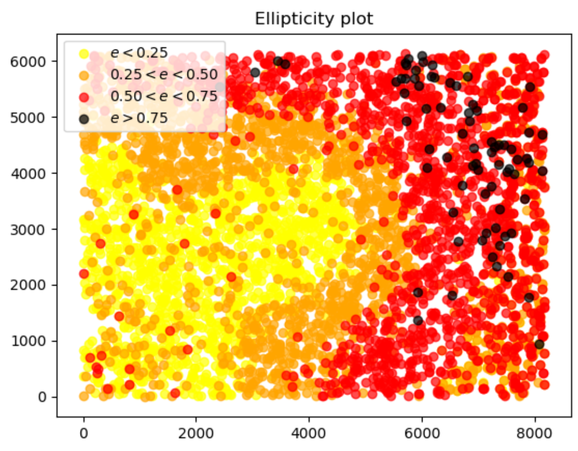


Figure 8: Demonstration of how the ellipticity varies across the image. This shows a very clear smooth increase across the image, with a maximum at the right-hand edge, consistent with direct observation shown in Figure 7.

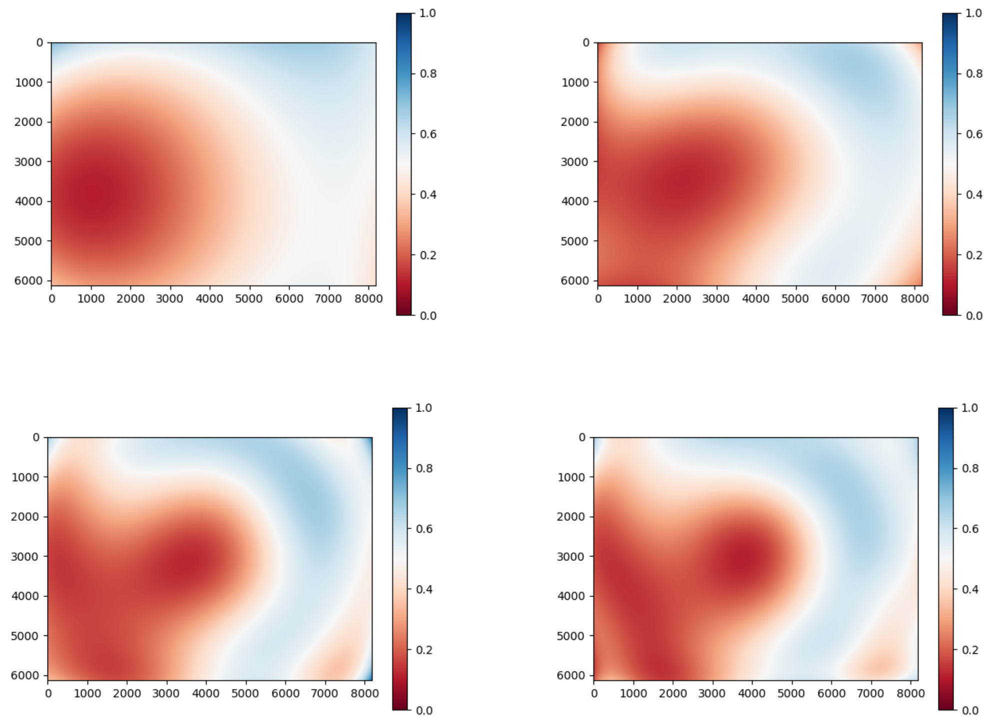


Figure 9: Visualisations of the ellipticity fitting. Plots are shown for a (*Top left*) 3rd, (*Top right*) 5th, (*Bottom left*) 7th, and (*Bottom right*) a 9th degree polynomial.

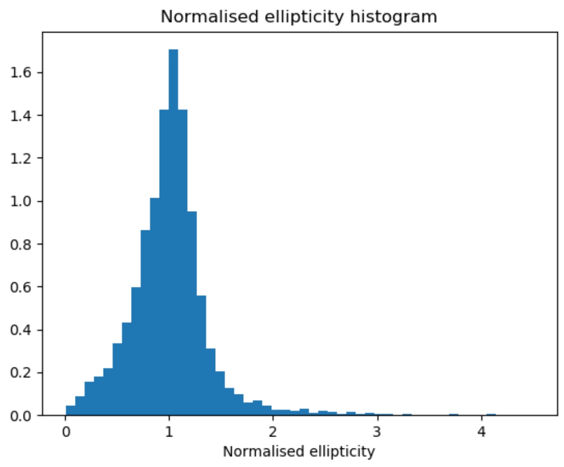


Figure 10: An example of the normalisation of ellipticity for a 5^{th} degree polynomial. There is a clear clustering around 1, with outliers that likely represent galaxies.

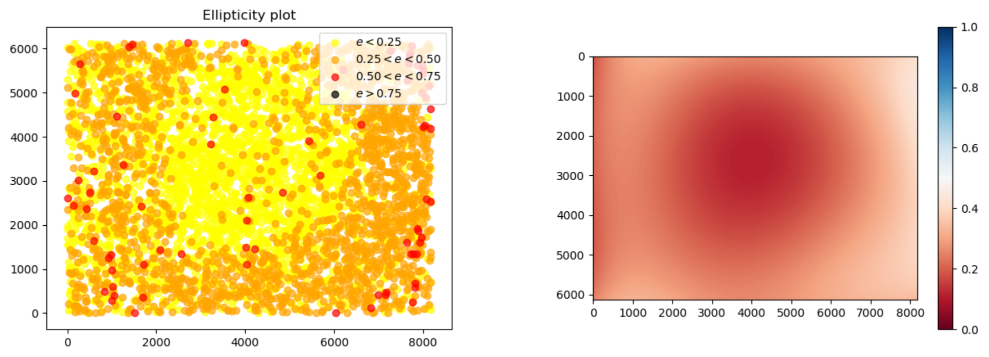


Figure 11: *Left* Demonstration of the ellipticity variation across a new image from the GOTO telescope, and *right* a visualisation of the ellipticity fitting. Both plots were performed on the calexp-4-67965.fits image, and show a much less extreme variation than in Figures 8 and 9

3.4.1 Improvements to Data Reduction

The methods presented for source detection, and data reduction had some success, however both had significant problems which have been presented. Both steps are fundamental for an effective ML algorithm, and so they must be as robust, and efficient as possible. Another method that was available was to take advantage of the source detection within the GOTO pipeline. This source detection was capable of robustly detecting significantly more sources within the GOTO images. Crucially for the data reduction stage, it was also able to measure several extent parameters of the sources, and the PSF. Further improvements were made by considering multiple parameters, such as area and orientation. These parameters were directly measured through the `deblend_sources` function. However, current tests have not shown any obvious outliers when using these parameters, and further investigation will need to be required before choosing one method over another.

3.5 Creating Cutouts

In order for the machine learning to be performed on the individually detected sources, it was required to create patches centered on the source to be cut out from the main image. This was completed in

Python through the `Cutout2D`[15] function to create 50 x 50 pixel cutouts. The cutouts were then saved as `FITS` files, with the right ascension (RA) and declination (Dec) of the sources saved within the header file. Examples of 50 x 50 cutouts are shown in Figure 12. Once a cutout has been created, the images created will become the inputs to the ConvNet.

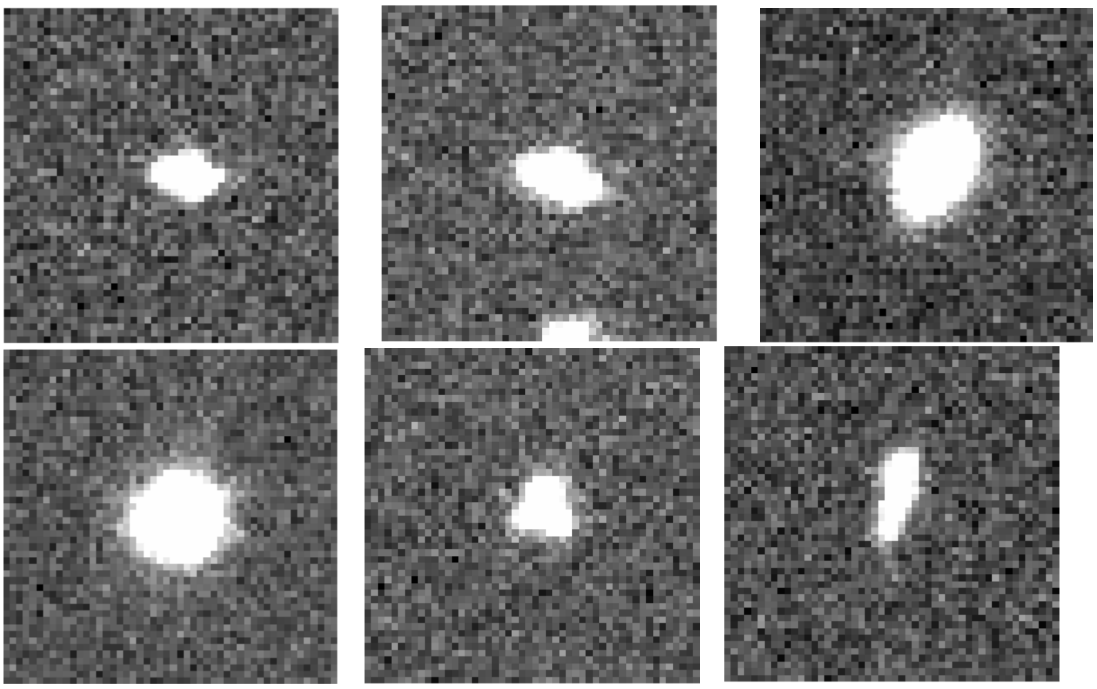


Figure 12: Examples of 50 x 50 cutouts taken from calexp-2-133758.fits. These are the images that will be inputted into the ConvNet to perform classifications.

4 Progress Plan

The next phase of the project is to implement the data into the chosen ConvNet. Before this however, there are still several steps which are required to be taken, the approximate time frames of which are summarised in Figure 13. An important aspect is to gain some practical experience of implementing a ConvNet in Python. This can be achieved through the plethora of online tutorials available, and will be the focus over the Christmas break. The next step that must be taken is to continue to test the methods of data reduction which are discussed in **3.4**. If this data reduction cannot produce robust, reliable results within a short enough time frame, then this section will have to be abandoned in favour of allowing sufficient time to perform the various ML steps. The data reduction is not essential for an accurate ML module, and was designed more as a means to increase efficiency.

The primary focus of the work will be concerned with building a large enough training set to effectively train the ConvNet. A likely candidate for a training set will be Galaxy Zoo (GZ). Like GOTO, GZ is also a wide area survey that covers a large area of the sky. This will mean that it will provide a similar data set, unlike if a deep field survey, like CANDELS, was employed. It will also be important to cross-reference the sources (using RA and Dec calculations from creating the cut-outs), to ensure that the training set covers a similar patch of sky to the GOTO data. Careful considerations into overfitting will have to be considered along this, and so a training set that is equally weighted between the broad morphological classes of galaxies, and stars will have to be implemented. Once a training set that is sufficiently large has been created, the GOTO images can then be inputted into the ConvNet to begin classification.

Classification through use of the ConvNet will initially be focused on star-galaxy separation. If the data reduction techniques prove successful, this will be a less vital stage, however it is still possible that the simple algorithms employed will not cut all the stars from the sample. ConvNets were chosen due to their ability to exploit transfer learning, which will allow us to repurpose an existing ConvNet which

[15]`http://docs.astropy.org/en/stable/api/astropy.nddata.utils.Cutout2D.html`

will be trained on the training data created. Therefore, it will be necessary to review the various source codes of available ConvNets to test which provides the most accurate results. Once the ConvNets have been narrowed down to a single algorithm, it will then be possible to input the GOTO data and begin performing classification of stars and galaxies initially. Due to the differences in image quality on each CCD, it is likely that the model will need to be individually trained on each CCD separately. If time permits, once the star-galaxy separation has been proven to be an effective classifier based on morphology, it will then be the aim to classify the galaxies into broad morphological categories, such as late and early type.

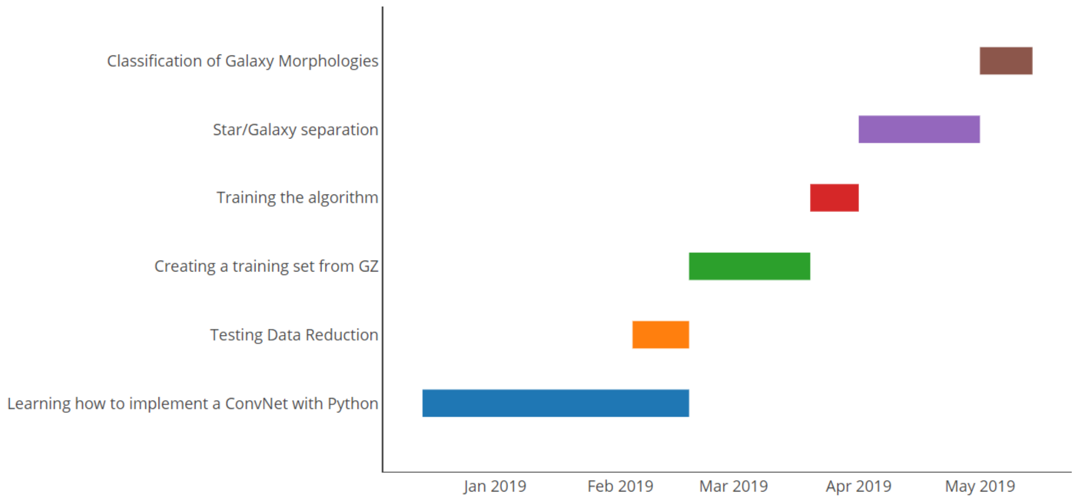


Figure 13: Gantt chart showing the approximate time frames of the different tasks which are required to be completed in future work in this study.

References

Aniyan, A. K. and Thorat K., *Classifying Radio Galaxies with Convolutional Neural Networks*, arXiv:1705:03413v1 [astro-ph.IM], (**2017**)

Ball, N. M. et al., *Robust Machine Learning Applied to Astronomical Datasets I: Star-Galaxy Classification of the SDSS DR3 using Decision Trees*, arXiv:astro-ph/0606541v1, (**2006**)

Banerji, M. et al., *Galaxy Zoo: reproducing galaxy morphologies via machine learning*, Mon. Not. R. Astron. Soc. 406, 352-353 (**2010**)

Bengio, Y., *Foundations and trends in Machine Learning*, 2, 1 (**2009**)

Buisson, L., *Machine Learning in Astronomy*, University of Cape Town, (**2015**)

Calleja, J. and Fuentes. O, *Machine Learning and image analysis for morphological galaxy classification*, Mon. Not. R. Astron. Soc. 349, 87-93 (**2004**)

Cortes, C. Vapnik, V., *Support-vector Networks* V. Mach Learn, 20: 273., (**1995**)

De Vaucouleurs, G., *Classification and Morphology of External Galaxies*, Handbuch der Physik. 53: 275., (**1959**)

Dieleman, S. et al., *Rotation-invariant convolutional neural networks for galaxy morphology prediction*, Mon. Not. R. Astron. Soc. 000, 1–20 (**2014**)

Domingos, P., *A few useful things to know about Machine Learning*, Communications of the ACM, Vol. 55 Issue 10, 78-87, (**2012**)

Eyono, R. H., *Feature Extraction and Selection of Optical Galaxy Data*, University of Cape Town (**2017**)

Gauthier, A. et al., *Galaxy Morphology Classification*, (**2016**)

The Gravitational-Wave Optical Transient Observer, WWW document `https://goto-observatory.org`, (**2018**)

Hocking, A. et al., *An Automatic taxonomy of galaxy morphology using unsupervised machine learning*, arXiv:1709.05834v1 [astro-ph.IM], (**2017**)

Howard, E. M., *Machine learning algorithms in Astronomy*, (**2015**)

Huertas-Company, M. et al., *A robust morphological classification of high-redshift galaxies using support vector machines on seeing limited images II: Quantifying morphological k-correction in the COSMOS field at* $1 < z < 2$*: Ks band vs. I band*, Astronomy Astrophysics 1255, arXiv:0811.1045v", (**2009**)

Huertas-Company, M. et al., *Revisiting the Hubble Sequence in the SDSS DR7 spectroscopic sample: a publicly available Bayesian automated classification*, Astron. Astrophy., 15735, (**2010**)

Huertas-Company, M. et al., *Measuring galaxy morphology at z ¿ 1. I - calibration of automated proxies*, arXiv:1406.1175v1 [astro-ph.GA], (**2014**)

Huertas-Company, M. et al., *A Catalog of Visual-like Morphologies in the 5 CANDELS Fields Using Deep Learning*, arXiv:1509.05429v1 [astro-ph.GA], (**2015**)

Humphreys R M, et al. *Experiments in automating the morphological classification of galaxies.* Am Astron Soc, 33: 1322, (**2001**)

Kim, E. J. and Brunner, R. J., *Star-Galaxy Classification Using Deep Convolutional Neural Networks*, arXiv:1608.04369 [astro.ph.IM], (**2016**)

Krizhevsky, A. et al., *Advances in neural information processing systems*, 1097 - 1105, (**2012**)

Li, L. et al., *k-Nearest Neighbors for automated classification of celestial objects*, Y. Sci. China Ser. G-Phys. Mech. Astron, 51:916, (**2008**)

Lintott, C. et al., *Galaxy Zoo: Morphologies derived from visual inspection of galaxies from the Sloan Digital Sky Survey*, arXiv:0804.4483v1 [astro-ph], (**2008**)

Pérez-Carrasco, M. et al., *Multiband galaxy morphologies for CLASH: a convolutional neural network transferred from CANDELS* , arXiv:1810.07857 [astro-ph.IM], (**2018**)

The Sloan Digital Sky Survey: Mapping the Universe, WWW document `https://www.sdss.org/`, (**2018**)

Storrie-Lombardi, M. et al., *Morphological classification of galaxies utilizing neural networks*, American Astronomical Society, 181st AAS Meeting, id.65.08; Bulletin of the American Astronomical Society, Vol. 24, p.1222, (**1992**)

Vasconcellos, E. C. et al., *Decision Tree Classifiers for Star/Galaxy Separation*, The Astronomical Journal, 141:189(12pp), (**2011**)

Yosinski, J. et al., *How transferrable are features in deep neural networks?*, arXiv:1411.1792v1 [cs.LG], (**2014**)