Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

BACHELORTHESIS

Classification of Galaxy Spectra with Artificial Neural Networks

vorgelegt von

Julia Anabell Ziegler

Fakultät für Mathematik, Informatik und Naturwissenschaften

Fachbereich Physik

Studiengang: Bachelor of Science Physik

Matrikelnummer: 7066612

Erstgutachter: Prof. Dr. Jochen Liske

Zweitgutachter: Prof. Dr. Marcus Brüggen

Abstract

In astrophysics often huge amounts of data are collected and need to be evaluated, as for example galaxy spectra. However, some galaxy spectra might contain characteristic errors and it is desirable to flag these galaxies for each of these errors. This can be very time consuming when done by hand. Furthermore, when inspection of the data is undertaken by different people it can be difficult to stick to clear classification boundaries.
One possible solution for this is to inspect and flag the data with the help of machine learning.
This work aims to classify galaxy spectra from the GAMA spectroscopic survey for two different kinds of problems, namely 'fringing' and 'bad splicing', with the help of artificial neural networks (ANNs) which are a sub branch of machine learning. The ANNs are trained with spectra classified by eye and their performance is evaluated.

Zusammenfassung

In der Astrophysik werden oft große Mengen an Daten generiert, welche wiederum ausgewertet werden müssen, wie zum Beispiel Galaxiespektren. Hierbei kann es vorkommen, dass einige Spektren bestimmte charakteristische Fehler enthalten und entsprechend gekennzeichnet werden sollen. Dies kann sehr zeitaufwendig sein, wenn die Klassifizierung von Hand durchgeführt wird. Außerdem kann es schwierig sein, sich dabei an genaue Abgrenzungen der Klassen zu halten, wenn unterschiedliche Personen die Klassifizierung durchführen.
Einen möglichen Lösungsansatz zum Klassifizieren von Spektren bietet maschinelles Lernen.
In dieser Arbeit werden Galaxiespektren der spektroskopischen Untersuchung GAMA nach zwei unterschiedlichen Problemen, nämlich 'fringing' und 'bad splicing', klassifiziert. Dafür werden künstliche neuronale Netze verwendet, die eine Untergruppe des maschinellen Lernens bilden. Die künstlichen neuronalen Netze werden mit von Hand klassifizierten Spektren und trainiert und ihre Leistungsfähigkeit wird ausgewertet.

Contents

1 Introduction

1.1 Galaxy Spectra

The analysis of galaxy spectra is an important tool in astrophysics to determine for example the stellar population or redshift of galaxies. A key indicator to determine the characteristics above are emission and absorption lines of elements as well as the continuum which is a combination of many black body spectra of the stars in that galaxy.

For example, in general the spectra of elliptical galaxies are dominated by old cooler stars emitting light at longer wavelength, whereas the spectra of irregular and spiral galaxies are dominated by the blue light of young hot stars. Furthermore, most irregular and spiral galaxies show emission lines of Hα, [OII], [NII], [SII], where Hα (at 656.28 nm) is the strongest emission line and the most reliable indicator for the star formation rate [12]. Spectra of cooler stars on the other hand often show absorption lines, as atoms or molecules in the star's atmosphere absorb the star's light [3, Chapter 12] [12]. In Figure 1 four spectra of different galaxies are shown as an example. The optical images of the according galaxies can be found in Figure 2 for comparison.

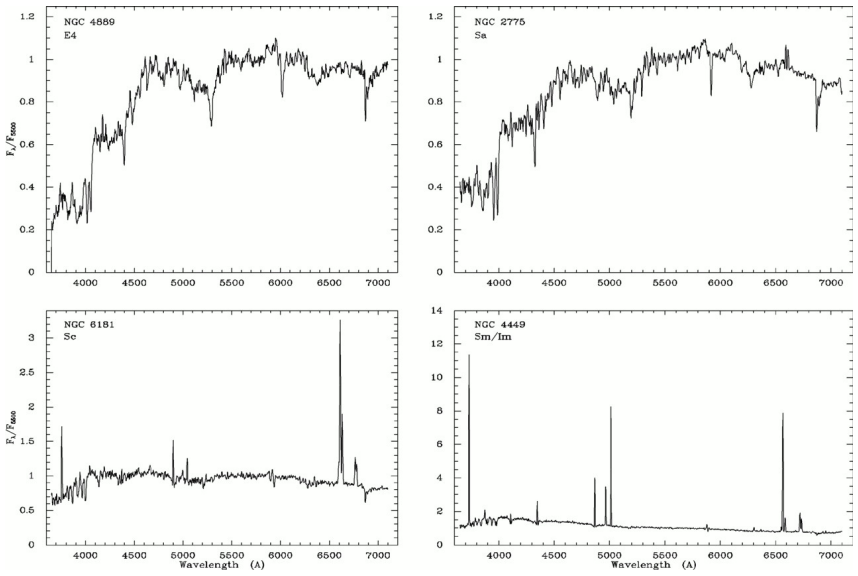
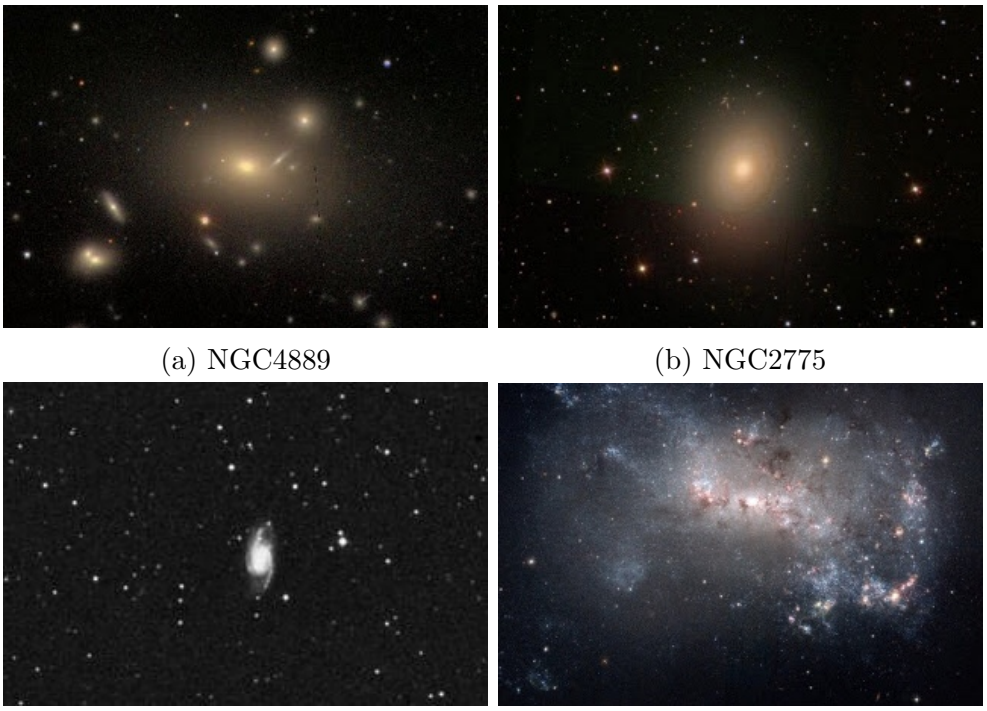


Figure 1: Shown are galaxy spectra for the galaxies (from left to right and top to bottom) NGC4889, NGC2775, NGC6181, NGC4449. [12]

As can be seen in Figure 1, the first two galaxies NGC4889 and NGC2775 show many absorption lines in their spectra which indicates an old population of stars found in spiral bulges and elliptical galaxies. Also, the continuum in the red half of the spectrum is higher than in the blue half which indicates cooler, older stars dominating the galaxy. Comparing this information with the optical images in Figure 2(a) and (b) shows that these two galaxies are in fact elliptical.
The two galaxies at the bottom in Figure 1 show strong Hα emission lines in their spectra



(a) NGC4889 (b) NGC2775

(c) NGC6181 (d) NGC4449

Figure 2: Shown are optical images of the galaxies of which the spectra are shown above. [Images from the NASA/ESA Hubble Space Telescope]

as well as a relatively high flux in the blue part of the spectrum indicating star formation and the existence of many young hot blue stars. And indeed, as can be seen in Figure 2(c) and (d), these galaxies are spiral and irregular respectively.

A galaxy's local velocity can be determined by calculating the optical Doppler shift of the light waves emitted by the galaxy. If a galaxy is moving away from the observer, the wavelength observed appears shifted to longer wavelengths (If it is moving towards the observer, the observed wavelength are shifted to shorter wavelength.). This is known as redshift z and is defined as:

$$z = \frac{\lambda_{observed} - \lambda_{emitted}}{\lambda_{emitted}} \tag{1a}$$

$$1 + z = \frac{\lambda_{observed}}{\lambda_{emitted}}. \tag{1b}$$

Where $\lambda_{observed}$ is the observed wavelength and $\lambda_{emitted}$ is the emitted wavelength [3, Chapter 2].
For velocities completely in radial direction the redshift can be calculated as:

$$1 + z = \sqrt{\frac{1 + \frac{v_{radial}}{c}}{1 - \frac{v_{radial}}{c}}} \underset{v_{radial} \ll c}{\approx} 1 + \frac{v_{radial}}{c}. \tag{2}$$

With v_{radial} being the speed of the galaxy in radial direction and c the speed of light in vacuum [3, Chapter 2].
Since characteristic combinations of absorption and emission lines and the corresponding wavelength are already known, the redshift can be determined by finding these characteristic lines in the galaxy's spectrum and comparing them to the known values for the wavelengths of these lines [3, Chapter 2]. Solving Equation 2 for the radial velocity we find:

$$v_{radial} = c \cdot \left(\frac{(1+z)^2 - 1}{(1+z)^2 + 1} \right). \tag{3}$$

But a redshift can not only be caused by motions through space, but also by the expansion of space itself. Since space is expanding, everything in the universe is moving away from everything else, similar to dots drawn onto the surface of an inflating balloon. This also means that the farther an object is away from an observer, the faster it has to move away from it since there is more space expanding in between them. For short distances this effect overlaps with the redshift caused by local velocities through space but for larger distances the velocities caused by the expansion of space increase and therefore effects caused by local velocities can be neglected. The relation between a galaxy's velocity v and its distance r can be written as:

$$v = H(t) \cdot r. \tag{4}$$

Where $H(t)$ is the time dependent Hubble constant.
This means that by measuring the redshift of characteristic absorption or emission lines, the distance of the galaxy which emitted these lines can be determined [3, Chapter 12] [2, Chapter 12]. But since space is expanding all the time, also while the light from that galaxy was on its way to the observer, the calculation has to be done using time dependent cosmic scaling factors. This will not be discussed in detail here.

1.1.1 GAMA Spectroscopic Survey

The GAMA project aims to study cosmology and galaxy formation and evolution. The main part of this project is the GAMA spectroscopic survey in which spectra of about 300,000 galaxies were taken using the AAOmega spectrograph on the 3.9 m Anglo Australian Telescope (Siding Spring Observatory, NSW, Australia). A field of about 286 deg^2 was covered during the observations taken from 2008-2014 and objects down to a magnitude of $r < 19.8$ mag were observed [1] [15]. The AAOmega spectrograph possesses a dual beam system and covers a wavelength range from 3750 Åto 8800 Å. For the observations 392 optical fibres are positioned by a robot gantry of the 'Two-Degree-Field' instrument to each observe different galaxies. Furthermore eight fibre bundles are used for accurate positioning of the telescope.

After observation the data is processed using a software called 2DFDR which applies standard procedures with some modifications to extract 1D spectra from 2D images [10]. Due to data reduction and other effects in the telescope and instruments different kinds of problems can appear as visible errors in the spectra. Some of them will be discussed in the next section.
For the readily processed spectra the redshifts are determined, which is the main goal of the GAMA survey.

1.1.2 Classification of Galaxies

The classification of galaxy spectra can have both scientific and technical goals.

For a **scientific classification** many different characteristics need to be taken into account, for example morphology, size, luminosity, mass, rate of star formation, history of star formation, metallicity, mass of cold gas and dust and their distribution, mass of Dark Matter halo, distance, and cosmological epoch to only name a few. Furthermore, many of these properties are correlated. [3] One commonly used way of classification by morphology is the Hubble diagram shown in Figure 3.

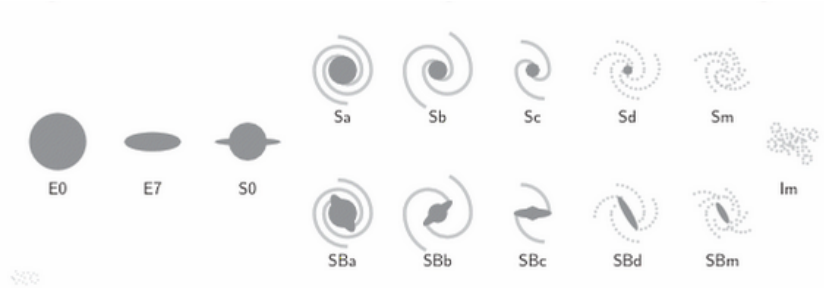


Figure 3: Shown is the Hubble diagram used to classify galaxies by their morphology. [3, Page 386]

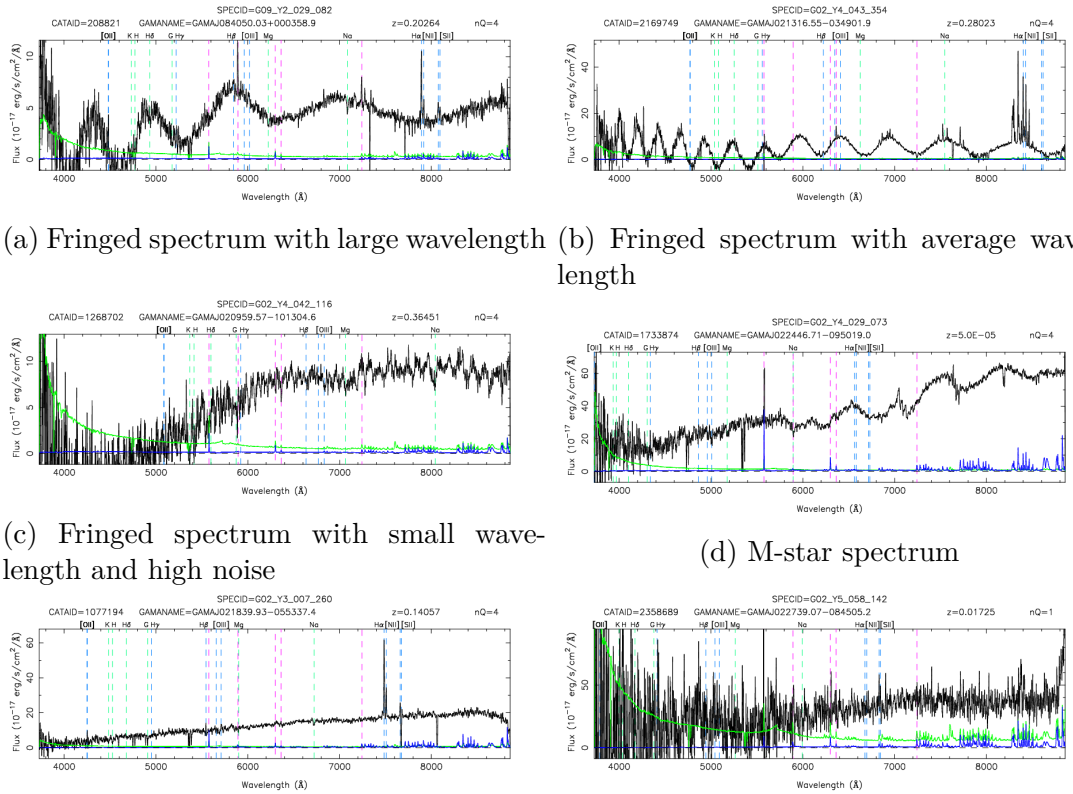
In this diagram three main types of galaxies can be seen: elliptical, spiral, and barred spiral galaxies. As described in the section above these shapes are also correlated for example to the stellar age and rate of star formation of a galaxy. Elliptical galaxies host an older star population whereas spiral galaxies have more young hot blue stars.

A **technical classification** can be done mainly based on effects due to data reduction and other errors occurring in the instruments.
In the GAMA spectroscopic survey for example some frequently arising problems are bad sky subtraction, missing data from the red or the blue half of the spectrum, 'bad splicing' and 'fringing'.
'Fringing' is the appearance of a sine-like pattern in the spectrum. This arises due to tiny air gaps in the glue at the connection points of the prism and the ferrule in the AAOmega instrument which cause oscillations in the flux [10]. The wavelength and amplitude of the

sine-like pattern vary from spectrum to spectrum and can even change within one spectrum or appear more visible only in some parts of a spectrum. Furthermore, the spectra have different signal to noise ratios which in some cases can make it difficult to recognize 'fringing' by eye, especially in cases where the amplitude is very small. The 'transition' from 'fringing' to 'normal' spectra is smooth and it is difficult to define a clear boundary for this effect.

Another difficulty in identifying fringed spectra can be the distinction between fringed spectra and M-stars. M-stars are cold stars which show broad bands of absorption lines due to them being cold enough so that molecules can form in their atmospheres [4]. But once one has learned what the characteristics of spectra of M-stars are, the distinction becomes fairly easy for the human eye.

In Figure 4 some fringed and other spectra are shown.



(a) Fringed spectrum with large wavelength

(b) Fringed spectrum with average wavelength

(c) Fringed spectrum with small wavelength and high noise

(d) M-star spectrum

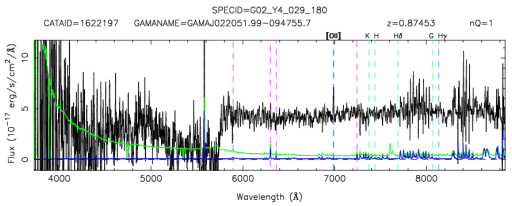
(e) Not fringed spectrum with high signal to noise

(f) Not fringed spectrum with low signal to noise

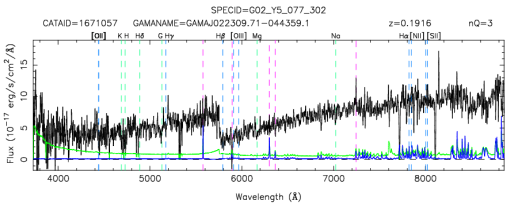
Figure 4: Shown are some spectra from the GAMA spectroscopic survey [1].

'Bad splicing' is a mismatch of the flux of the blue and the red half of the spectrum and appears due to poor continuum level estimation in one or both of the blue and red arms of the AAOmega instrument. This results in a visible step in the flux always appearing at ~ 5700Å [10]. This step can either go upwards or downwards and can have different heights as well as being sharp or slightly elongated. In some cases it might be difficult to see by eye, especially when spectra show strong emission lines which dominate the spectrum it might be necessary to zoom in to have a closer look at the continuum and

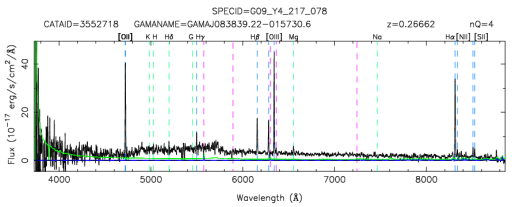
detect a bad splice. Below in Figure 5 some spectra with a bad splice are shown. As with 'fringing' the transition from spectra with 'bad splicing' to spectra not having this problem is smooth which complicates defining a clear boundary.



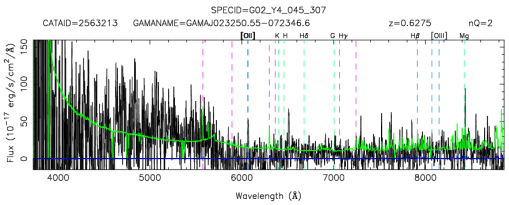
(a) Bad splicing spectrum with strong step going up



(b) Bad splicing spectrum with average step going down



(c) Bad splicing spectrum with soft step



(d) Bad splicing spectrum with low signal to noise

Figure 5: Shown are some spectra from the GAMA spectroscopic survey [1].

Some spectra can even have a combination of different problems. However, for both 'fringing' and 'bad splicing' usually a redshift can still be determined since the position of emission lines is not affected by this. Furthermore, 'fringing', 'bad splicing' and other problems appear in only a small fraction of the spectra.

Nevertheless it is useful to identify these spectra for different reasons. For example, by finding fringed spectra and checking which optical fibre was positioned on that galaxy to create this spectrum, it might be possible to identify which of the optical fibres causes this problem so that it can be replaced in the future. Also it can be useful to keep track of how many of the spectra show problems as bad splicing an fringing to see where the process of creating the spectra can be improved further. Also the spectra might be used for other problems than identifying the redshift. Then the correct flux values could be needed and fringed spectra or spectra with a bad splice need to be sorted out.

1.2 Artificial Neural Networks (ANNs)

Artificial neural networks (ANNs) are a type of weak artificial intelligence inspired by biological neural networks found in the brains of animals. Their two key characteristics are the ability to learn and the ability to handle probabilistic information. This makes them useful tools for different kinds of computational tasks like for example image recognition, classification, recognition of patterns or even medical diagnostics. Promising results using ANNs were also obtained in astrophysics, such as the foreground-cleaning of the cosmic microwave background [17] or for classification of pulsar-like candidates [11].
But the use of ANNs is also limited. To understand the limitations, it is important to understand how ANNs work.

1.2.1 How ANNs work

The general way of how ANNs work is to mimic the pattern connecting the input data with the output data. There are many different ways of achieving this, which have different advantages and disadvantages.
Here we describe how ANNs works for a one-dimensional input, so where the input is a vector of size n (for a two-dimensional input for example an image the same concepts still work). In general an n-dimensional input vector $\vec{x}$ with the components x_i, $i = 1, 2, ..., n$ is passed through a layer of m neurons, which can connect the x_i in different ways. (Each x_i represents one data point of the input.) There are many different kinds of layers. We will explain the concept of two kinds of layers namely dense layers and convolution layers since we will use them for this work.
Dense layers, also called fully connected layers, apply different arbitrary weights to each component x_i of the input and calculate the weighted sum. If the input $\vec{x}$ is an n-dimensional vector and is passed through m neurons, it is multiplied with the corresponding weights W which is an $(m \times n)$-dimensional matrix:

$$\vec{x} = (x_1, x_2, ..., x_i, ..., x_n)^T; \quad i = 1, 2, ..., n$$

$$W = \begin{pmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & \dots & w_{mn} \end{pmatrix}.$$

The result is the weighted sum $\vec{y}$ which is an m-dimensional vector, where each component y_j is one weighted sum for one of the m neurons.

$$\vec{y} = W \cdot \vec{x} = (y_1, y_2, ..., y_j, ..., y_m)^T; \quad y_j = \sum_{i=1}^{n} w_{ji} \cdot x_i; \quad j = 1, 2, ..., m \tag{5}$$

Each neuron then applies the activation function f to each component of $\vec{y}$. A bias θ may be applied. Two exemplary activation functions can be seen in Figure 7.

$$f(y_j + \theta) = f(\sum_{i=1}^{n} w_{ji} \cdot x_i + \theta) \tag{6}$$

This results in the output $\vec{o}$ which is an m-dimensional vector.

$$\vec{o} = (f(y_1), f(y_2), ..., f(y_j), ..., f(y_m))^T = (o_1, o_2, ..., o_j, ..., o_m)^T; \quad j = 1, 2, ..., m \tag{7}$$

The output is then passed on to the next layer where it serves as input. In contrast to convolution layers, dense layers keep the information about the position of a characteristic within the input, because the inputs are directly connected to the outputs via the weighted sums. They are also employed in classification problems, where they connect input vectors to the corresponding class, represented by an output vector.

Convolution layers take the input vector $\vec{x}$ and pass a convolution window, the kernel $\vec{k}$, over it. The kernel has a defined size m smaller than the dimension of $\vec{x}$ and consists of arbitrary weights. The convolution of a function h with a function g is defined as:

$$(h * g)(t) = \int_{-\infty}^{\infty} h(\tau) g(t - \tau) \, \mathrm{d}\tau. \tag{8}$$

For the convolution of two vectors, the input $\vec{x}$ corresponds to the function g and the kernel $\vec{k}$ corresponds to the function h, τ corresponds to the indices of $\vec{k}$ and $(t - \tau)$ corresponds to the indices of $\vec{x}$. Passing $\vec{k}$ over $\vec{x}$ means that for each index t the components k_τ and $x_{t-\tau}$ are multiplied and summed up for all indices τ. The result is a vector with the indices t which has the same dimension as $\vec{x}$. The components of this vector are:

$$(\vec{k} * \vec{x})_t = \sum_{\tau=1}^{m} k_\tau x_{t-\tau}. \tag{9}$$

(In reality the dimension of the output is slightly smaller than the one of $\vec{x}$ because padding has to be applied depending on the dimension of $\vec{k}$. This is due to the fact that $\vec{x}$ does not have indices lower than 1, so the value $x_{1-\tau}$ might not exist. Instead the lowest value for t should be $(\tau + 1)$.)

The result of the convolution is then passed to the neuron which applies the activation function f to each component. With convolution layers edges can be detected independent of their location within $\vec{x}$. These edges can for example be the edges of an object in a two-dimensional picture or jumps in a one-dimensional signal. After a convolution layer, usually a pooling layer is added to reduce the dimensionality of the output and improve run time. To do so a pooling window of defined size runs over the output and passes the maximum or average value within the pooling window to the next layer, depending on whether maximum pooling or average pooling is chosen. Unnecessary information is

dropped this way.
Usually this process is repeated M times with different kernels of same size but with different arbitrary weights. This creates M different copies (also called filters) of the convolution to recognize M different features in the input vector.
To recognize more complex patterns several convolution layers need to follow. Usually with each convolution layer the number of copies M is increased, as the dimensionality of the output decreases with each pooling layer.
To summarize, in each layer artificial neurons take the input which can for example be a weighted sum or convolution and apply an activation function to it. The output is passed on to the next layer and becomes its input until the last layer, the output layer, is reached. ANNs are often used for classification problems where the output layer represents the different classes, but they are also applied for regression problems. This work, however, will be concentrate on classification problems.
In Figure 6 a schematic of an artificial neuron in a dense layer and two commonly used activation functions are shown.

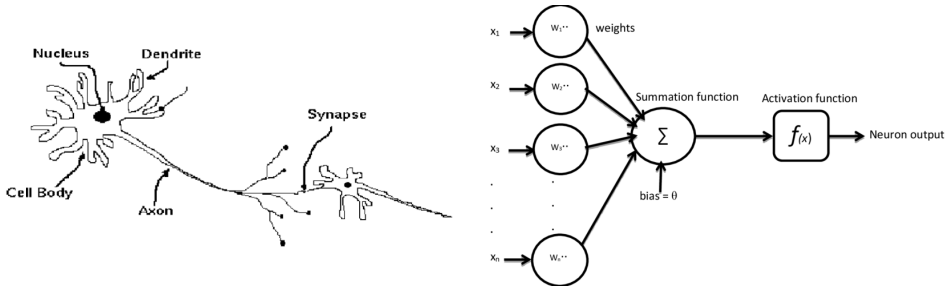
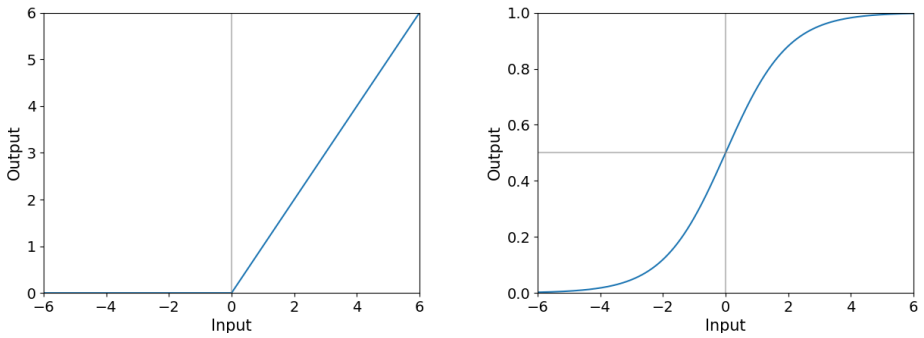


Figure 6: Shown are the schematics of a biological (left) [6] and an artificial (right) [19] neuron.



(a) ReLU Function (subsubsection A.1.1)

(b) Sigmoid Function (subsubsection A.1.2)

Figure 7: Shown are two commonly used activation functions. The argument is the weighted sum of the inputs and the bias equals zero for the shown cases.

In Figure 8 some exemplary architectures can be seen. At each node a neuron applies an activation function and passes the output to the next layer. In this work we will take

advantage of the deep convolution neural network, which consists of a few convolution layers followed by a number of dense layers.

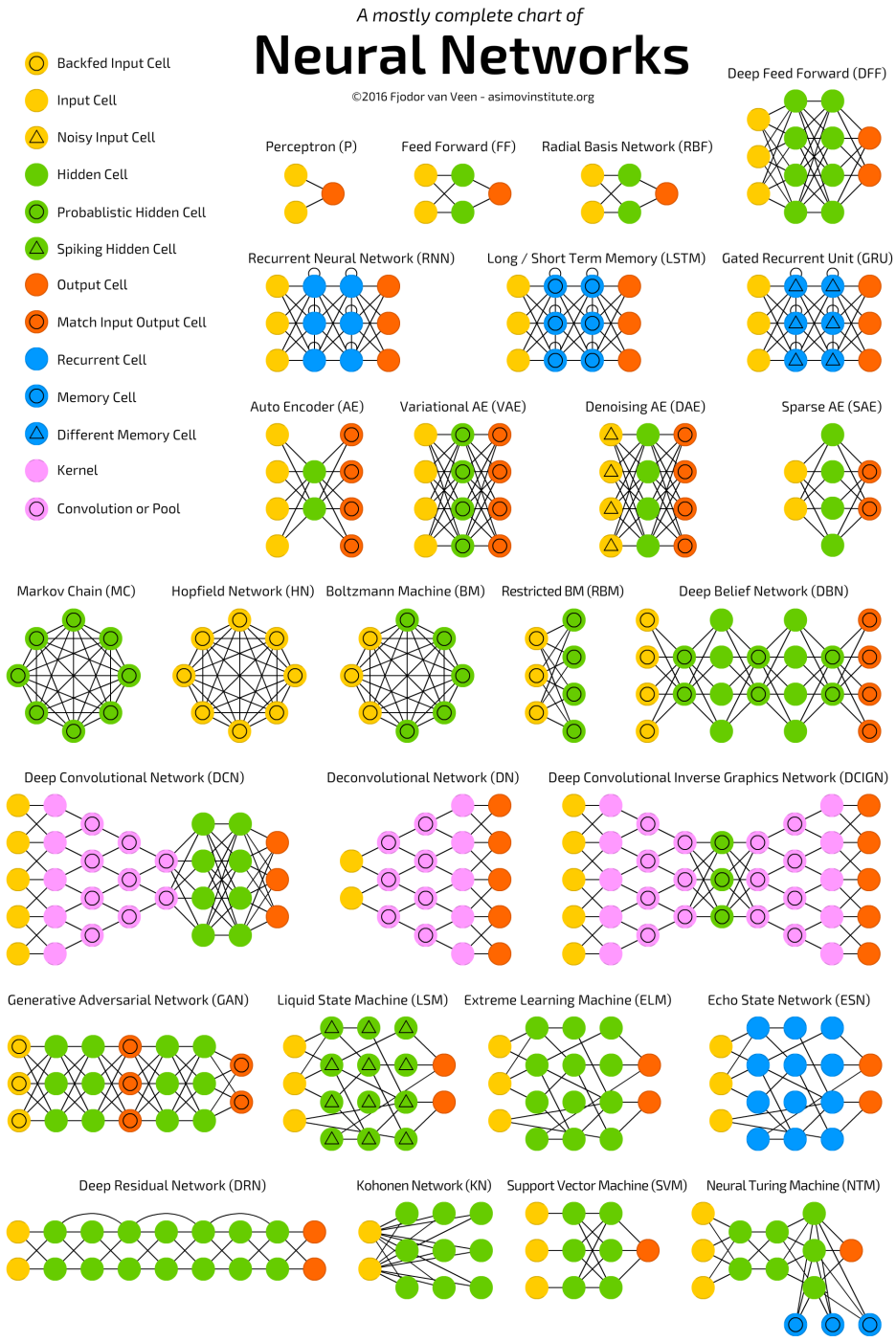


Figure 8: Shown are different architectures of ANNs. [18]

By adjusting the weights (either of the convolution window or in dense layers of the weighted sum) ANNs learn. This can be done by supervised learning, unsupervised learning, or reinforced learning.

Reinforced learning is done by passing some kind of a reward which is a measure of how well the ANN performed, similar to playing a game like Super Mario and trying to

reach the highest score. By adjusting the weights, the ANN tries to maximize this reward.
Unsupervised learning on the other hand does not have some kind of a reward. The ANN tries to find a pattern within the different input data and groups them in different categories. Like if you would pass someone a bucket with different fruits and they have to find out which fruits are similar and belong to the same kind, without telling them how many or which kinds of fruits there are.
Supervised learning works by passing previously labelled input data to the ANN and training on these. The ANN compares the predicted outputs to the labels and calculates how well it performs with a loss function. Using the results from the loss function the ANN adjusts the weights to minimize the loss. This would be like passing someone a bucket with different fruits and telling them which one belongs to which kind of fruit to teach them.
It remains to say that there are endless ways to construct an ANN and many different ways to train it. But within the framework of this work the concepts described above should be sufficient.

During supervised training often over fitting can appear. This means that the ANN tries to fit to the specific training examples too much, which will result in poor performance when unknown examples are presented to the ANN. One way to analyze this problem is by defining a validation sample which comes from the same sample as the training data, but other than the training data the ANN will not train on the validation set but only evaluate. That means predictions are compared to the actual labels but the weights of the ANN will not be adjusted. By comparing the performance on the training set to the performance on the validation set over fitting can be detected. An ANN which does not over fit should perform comparably well on the validation set like on the training set.
To tackle this problem different techniques can be applied. For example, it is desirable to have a simple ANN rather than a complicated one to avoid over fitting since an ANN with too many parameters will be able to tweak those parameters to over fit. One simple solution is applying a drop out layer. This causes the ANN to randomly put input nodes to zero which simplifies the network.
In addition to the validation set, a test set should be used which is not from the same sample like the training data, to evaluate performance on completely unknown data. However in many cases having one training and one test set is sufficient. Some ways to measure performance are discussed in the following section.

1.2.2 Evaluating ANNs

When supervised learning is applied, after training on a training sample (and evaluating on a validation sample) ANNs are evaluated on a test sample. There are different ways to measure the performance of a classifying ANN. We will explain the most commonly

used measures using the example of a classification problem where there are two classes, one which is called 'positive' and one which is called 'negative'. The network then can make true or false predictions on either class. So, the possible outcomes are: true positive (TP), false positive (FP), true negative (TN) and false negative (FN).
One way to evaluate performance is by accuracy (ACC) which is defined as the number of all true predictions divided by the number of all predictions.

$$ACC = \frac{TP + TN}{TP + FP + TN + FN} \tag{10}$$

When the performance on a specific class is to be evaluated two common measures are precision (PRE) and recall (REC).
Precision is the number of true positives over all positive predictions (true positives and false positives). This is the probability that a positive prediction is correct.

$$PRE = \frac{TP}{TP + FP} \tag{11}$$

Recall on the other hand is the number of true positives over all actual positives (true positives and false negatives). This is the probability that a positive example is correctly identified.

$$REC = \frac{TP}{TP + FN} \tag{12}$$

Depending on whether it is more important that a positive prediction is correct or that positive examples are identified one should pay more attention to either precision or recall. If both are equally important the F1 score ($F1$) is a good measure. It is a combination of both and is defined as:

$$F1 = \frac{2 \cdot PRE \cdot REC}{PRE + REC} \tag{13}$$

Another way to compare performance of different ANNs is the receiver operating characteristic (ROC) curve and the area under this curve (AUC). A ROC curve is created by plotting the true positive rate ($TPR = REC = \frac{TP}{TP+FN}$) against false positive rate ($FPR = \frac{FP}{FP+TN}$). The line where both rates are equal shows the performance of a classifier with random predictions. The goal is to stay above this line. The point (0,0) corresponds to a classifier which never predicts any positive classes, whereas the point (1,1) corresponds to the opposite. A classifier which randomly predicts positive 50% of the time will result in the point (0.5,0.5) and one that for example randomly predicts positive 70% of the time results in the point (0.7,0.7).
The curve is generated by evaluating test data on different class thresholds. If predictions of a classifier are assigned with the probability (between 0 and 1) for them belonging to the positive class, the true positive and false positive rate are calculated for different thresholds (between 0 and 1). Each threshold value creates a pair (true positive rate, false positive rate) which corresponds to one point in ROC space [8].

This way the ROC curve and AUC are independent of decision thresholds and class distributions. However in many cases there is a good agreement between AUC and accuracy. [8] [5]. An example ROC curve is shown in Figure 9.

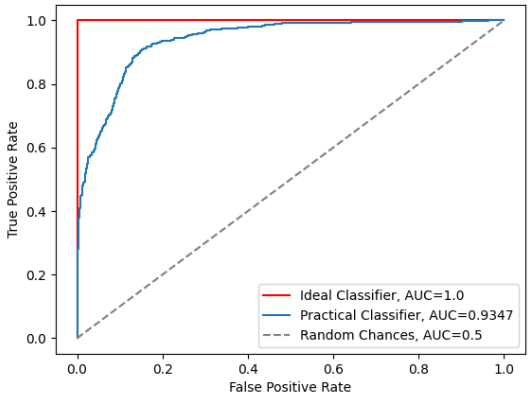


Figure 9: Exemplary ROC curve, with AUC values

There are other measures for the predictive power of ANNs but in this work we only use the ones mentioned above.
However, we want to point out that accuracy not always is a good measure for the performance of classifying ANNs. Especially when the distribution among classes is unbalanced. If for example 98% of a test sample belong to the negative class and only 2% belong to the positive class, and we imagine an ANN which predicts negative in every case it would be correct 98% of the time. But clearly the ANN would not be a good one and probably would not have learned how to classify for this problem correctly. For those cases it is better to concentrate on precision, recall, and F1 scores and the ROC curve. F1 score, precision, and recall are good measures when analyzing performance on a specific class, these values can especially be interesting when there is an imbalance in class distribution and a good performance on one specific class, for example the minority class, is desired. However, it should be kept in mind that some of these values depend on class distributions. For example, precision lower than 0.5 does not necessarily mean that predictions are worse than random if there are imbalances among the classes. If the ratio of positive examples in the test set is x, then precision of a random classifier (which randomly predicts positive in 50% of the cases) would also be $PRE_{random} = x$, however recall of a random classifier is $REC_{random} = 0.5$ independently of class distribution and F1 score of a random classifier is $F1_{random} = \frac{x}{x+0.5}$. To sum it up it depends on the application which values one should use to evaluate an ANN. If it is important to find all positive examples recall should be maximized, if it is more important to have a clean set of positives precision should be maximized and if both is equally important F1 score should be maximized. However, to analyze and compare overall performance of an ANN the ROC curve is a good measure, as it is independent of class distributions.
The problem of unbalanced data sets is explained in the following section.

1.2.3 The Unbalanced Data Problem

As pointed out earlier the performance of ANNs is limited, especially when supervised training is applied. One way it is limited is due to ANNs being biased towards balanced data sets. Earlier works show that ANNs perform best when trained and evaluated on balanced data, that means the data is distributed evenly across the classes [16] [14] [7]. As said earlier ANNs learn by adjusting their weights to minimize the loss which is calculated with a loss function. When training on unbalanced data the majority class has a bigger impact on the loss function which is why the ANN will try to fit to the data of this class better than to the data of the minority class. This results in a bias towards the majority class and in better performance on it but poor performance on the minority class.
Different techniques to achieve a good performance when training on unbalanced data include cost-sensitive learning, one-class learning and sampling methods.
When **cost sensitive learning** is applied, the ANN trains and evaluates on unbalanced data but a higher cost of loss is applied to the underrepresented classes, to even out the impact on the loss between majority and minority classes.
One-class learning on the other hand performs training only on one of the classes. The ANN learns to create strong boundaries for this class and can later, when evaluated on data not belonging to that class, distinguish between examples belonging to that class and examples not belonging to it.
Different **sampling methods** include over sampling and under sampling. When doing under sampling only a few samples from the majority class are used for training in order to create a balanced data set. Over sampling on the other hand is done by duplicating samples from the minority class to achieve balance. This way the loss of potentially important examples from the majority class is avoided. As an option also synthetic data can be generated.
In this work we will apply under sampling.

1.2.4 Other Limitations of ANNs

In addition to the unbalanced data problem described above ANNs are sensitive to input data in that way that wrongly labelled data prevent the ANN from learning correctly. Furthermore, big data sets are needed to achieve good performance. For classification problems it is crucial to have a clear distinction between the classes and strong similarity within one class. Also, to emphasize the characteristics responsible for the distinction between the classes, some data preparation has to be done.
Lastly, it remains to be said that currently there is no such thing as a recipe for designing ANNs or for how to prepare the data for best performance. Every problem is unique and finding the best performing ANN can be difficult since there are many different options on how to design the ANN and many different parameters involved.

2 Problem/Task

This work aims to classify two kinds of problems which appear frequently in the spectra taken with the AAOmega spectrograph on the Anglo Australian Telescope for the GAMA spectroscopic survey, namely 'fringing' and 'bad splicing'. To do so two deep convolution ANNs (one for 'fringing' and one for 'bad splicing') are created, and supervised training is applied. Construction and training of the ANNs is done with Keras [1], a deep learning framework implemented in Python. Furthermore, it is analyzed how the class distribution in the training data affects on the performance when evaluating test data with extremely unbalanced class distribution.

This work is split into four tasks:
In task 1 performance of the first ANN on the problem 'fringing' and the distinction of fringed spectra and spectra of M-stars is analyzed.
In task 2 the effect of class distribution in training data when evaluating on test data with extremely unbalanced but fixed class distribution is analyzed for the problem 'fringing'.
In task 3 again the problem 'fringing' is considered and the effect of training set size on performance is analyzed.
Finally, in task 4 performance of the second ANN on the problem 'bad splicing' is analyzed.

First, selection of training and test samples is explained. Then data preparation and the architecture of the two ANNs are discussed. After that results for the four tasks, described above, are presented.

[1] `https://keras.io/`

3 Main Part

3.1 Sampling

As stated earlier in this work random under sampling is applied to even out the imbalances among the classes. To do so random subsets from the GAMA spectroscopic survey are selected for both 'fringing' and 'bad splicing'. However, this is done only for the training data since ANN train best on balanced data. For the test data random sub samples are selected without artificially creating balance between the classes. Training on balanced and evaluating on unbalanced data usually will decrease the performance of ANNs because they are not only biased towards balanced data but also biased towards the class distribution they are trained on. But when evaluating on real data one does not know the labels for the test data beforehand and therefore cannot adjust the class distribution. By evaluating on a random sub sample with natural class imbalances we try to stay true to the actual problem.
During checking the data one problem arising is that many spectra are flagged wrongly or not flagged at all even if they are fringed or show a bad splice, so we decided to recheck all spectra used for this work. Due to time being limited we decide on creating small sub samples for training and test data. For each problem ('fringing and 'bad splicing') own training and test data sets are created.
For 'fringing' we create a balanced training sample with three classes, namely 'other', 'fringed' and 'M-star', to see whether the ANN can not only distinguish fringed spectra but also learn the difference between fringed spectra and M-stars. Each class contains 400 spectra. For the test set a random sub sample of 3,213 spectra with random class distribution is sampled. This set contains 3,082 'other', 92 'fringed' and 39 'M-stars'. The size of the test sample is chosen to still have enough spectra in each class to get representative results when evaluating. A larger sample (with more than minimum 39 examples per class) usually is necessary to evaluate an ANN properly but due to time being limited we do not sample more spectra. Also, a bigger training set probably is needed for proper training and better results. In many cases an amount of several ten thousand examples per class is used, but as we will see later it is possible to train on even such a small set.
Our criteria for 'fringed' spectra are that the sine-like pattern has to be clearly visible to the eye and the spectrum has to be dominated by it.

For 'bad splicing' we create a balanced training sample with two classes, namely 'other' and 'bad splicing'. Each class contains 800 spectra. For the test set the same sub sample containing 3,213 spectra with random class distribution as used for the problem 'fringing' is used, but sampled for the classes 'other' and 'bad splicing'. This set contains 2,849 'other' and 364 'bad splicing'. We chose a bigger training set because for the ANN it seems harder to learn the difference between 'bad splicing' and 'other' than 'fringing' and

'other'.
Our criteria for bad splicing are that the step must be clearly visible to the eye and that the jump in continuum, which can either go up or down, has to be sharp (this means the change in flux has to happen within a few pixel). Furthermore, there has to be a visible change in continuum. Spectra with a soft step or spectra with a sharp increase but also sharp decrease right afterwards down to the same flux are not marked as 'bad splicing'. It is possible that there is a bias towards spectra with a rather flat continuum, as 'bad splicing' is more visible in them. Also, there might be a bias towards spectra with a higher signal to noise ratio for both 'bad splicing' and 'fringing' as the characteristics of spectra can be hidden by high noise.

For both 'fringing' and 'bad splicing' randomly 10% of the training set is put aside for validation. Therefore we have two sets each on which we do not train but only evaluate: the validation set, which has the same class distribution as the training set, and the test set, which has a different class distribution than the training set. The validation set is used to detect over fitting and analyze performance on balanced data and the test set serves to analyze performance on a natural unbalanced distribution.
A full list with all used spectra for training and testing can be found on the Github page[2].
Next we describe how the data is prepared.

3.2 Data Preparation

For the two problems 'fringing' and 'bad splicing' different methods are applied to prepare the data. Some steps, however, are equal.
In both cases first the spectra are linearly interpolated using Scipy[3] to replace missing values. After that strong emission lines are removed using a sliding window created with Numpy[4], since for these problems emission lines are not relevant and could potentially lead the ANN to learn from them which is not desired. However in some spectra still smaller emission lines remain visible, since trying to completely remove them without affecting the sine-like shape in fringed spectra is difficult and leads to worse performance, especially when the emission lines are right on top of a wave maximum.
For the problem of finding **fringed** spectra, after removing emission lines, the spectra are cropped to a range of 4400 Å to 8750 Å (this corresponds to 4190 pixel), their mean values are set to zero and finally they are normed to the maximum of their absolute values. The reason for cropping the spectra is mainly to cut off some of the noise in the blue part of the spectra but also to bring all spectra to the same range of wavelength since this varied slightly.
For the problem of finding spectra with a **bad splice** a convolution with a box kernel

[2] https://github.com/juliaziegler/ann_for_galaxy_classification/tree/master/data_sets
[3] https://docs.scipy.org/doc/scipy/reference/interpolate.html
[4] https://numpy.org/

implemented with Astropy[5] with a size of 10 pixel is applied to emphasize the jump in continuum. The spectra are cropped to a range of 5000 Å to 6500 Å (this corresponds to 1440 pixel) since bad splicing appears in this area. This prohibits the ANN from learning from irrelevant parts of the spectra and improves computation time. Again their mean values are set to zero and they are normed to the maximum of their absolute values.
The effects of both ways of data preparation can be seen in the Figures 10, 11, and 12 below.

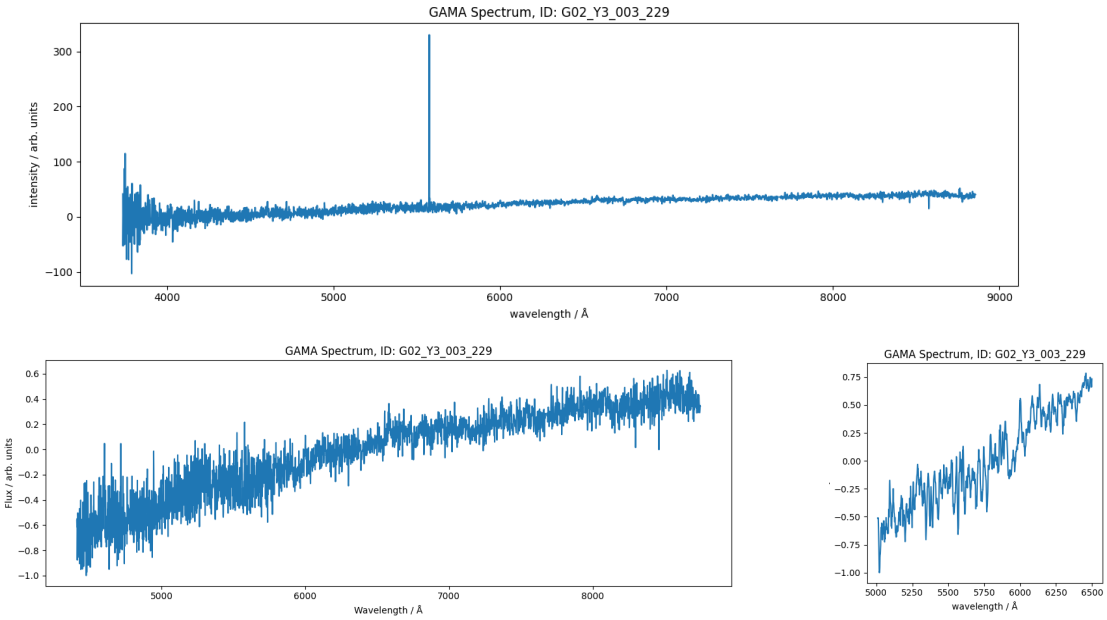


Figure 10: GAMA spectrum which is neither fringed nor has a bad splice: original, after data preparation for problem 'fringing' and after data preparation for problem 'bad splicing'

Some other data preparation techniques were applied but led to worse performance so they were dropped again. For example, we tried reducing noise and removing emission lines with lowpass filters and convolution with a Gaussian kernel. This results in the sine-like shape in fringed spectra being more visible to the human eye, as the spectra appear as sharp lines without much noise. But the ANN apparently has problems learning from these fine lines. One reason could be that this way the ANN learns too strong boundaries and only recognizes sine-like pattern which look exactly like the ones it trained on. Whenever there are slight differences (which obviously appear since every spectrum looks a little bit different) the ANN does not recognize this different shape as fringing anymore. However, with more noise the ANN allows for bigger variances and in turn can recognize fringed spectra with slightly longer/shorter or higher/lower amplitudes still as fringing.
In the next section the architecture of the ANNs is explained.

[5] `https://docs.astropy.org/en/stable/convolution/`

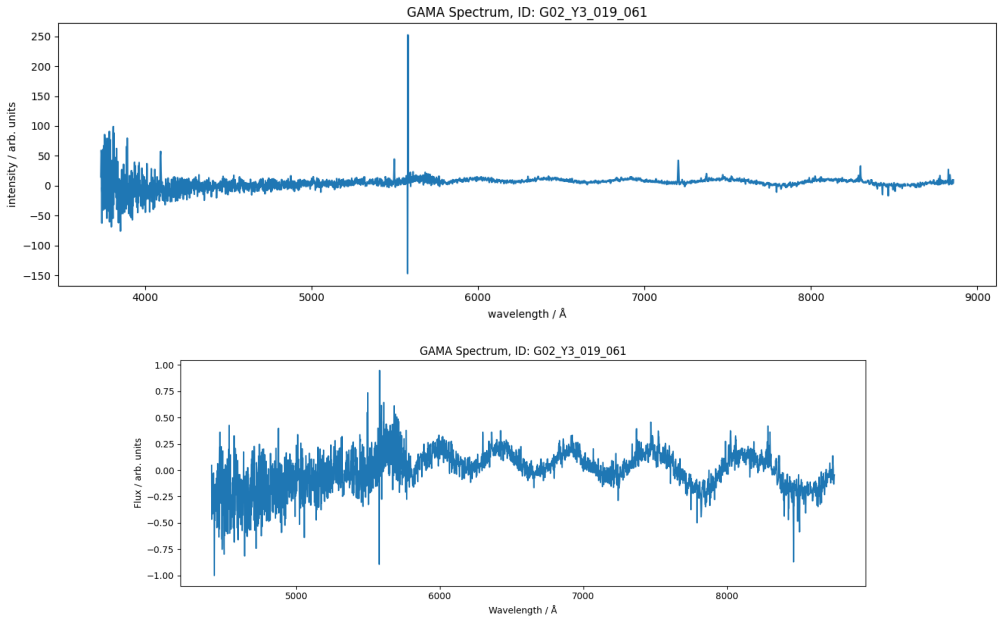


Figure 11: GAMA spectrum which is fringed: original and after data preparation for problem 'fringing'

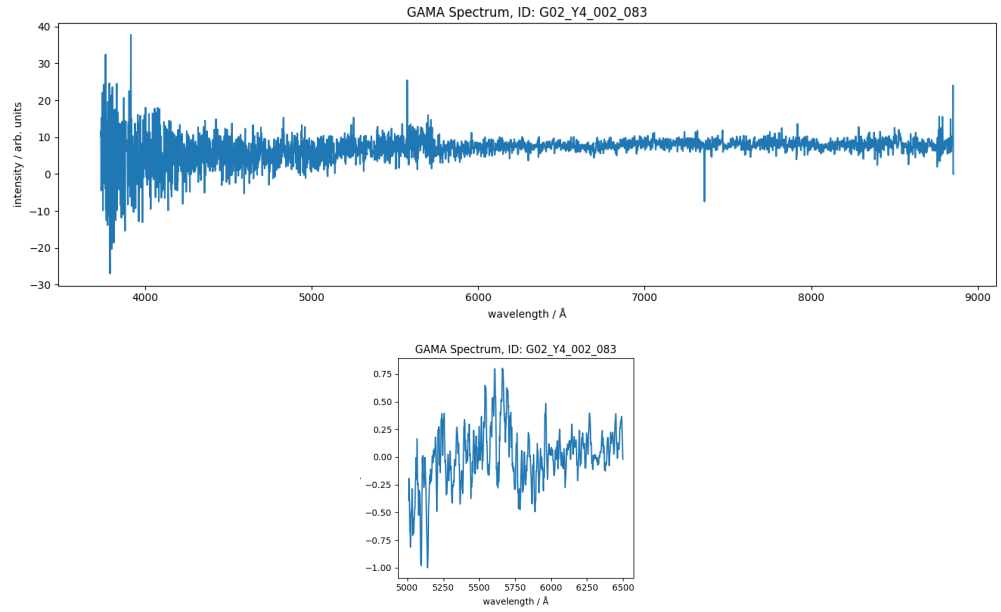


Figure 12: GAMA spectrum which has a bad splice: original and after data preparation for problem 'bad splicing'

3.3 ANN Architecture

For both problems two different ANNs are constructed using Python's deep learning framework Keras[6].

For the problem **'fringing'** we apply five independent convolution blocks of which the outputs are connected by dense layers. A schematic of the convolution blocks as well as the dimensions of the dense layers connecting these blocks can be seen in figure 13.

[6] `https://keras.io/`

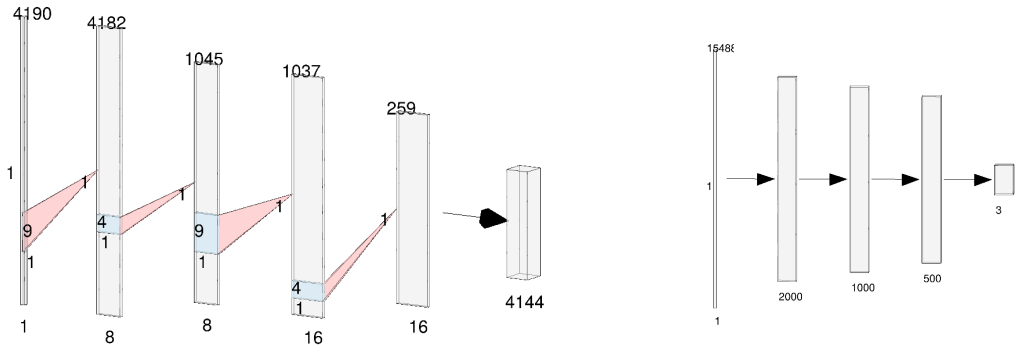


Figure 13: Schematic of the ANN for problem 'fringing': Shown are one of the five convolution blocks with convolution window of 9 pixel (left) and the dense layers connecting the convolution blocks (right), created with NN-SVG[7].

In the shown example the first of five convolution blocks can be seen on the left. Each block takes the same input, which is a one-dimensional vector of 4190 pixel (the size spectra are cropped to during data preparation) and passes a convolution window over it which in this example has a size of 9 pixel (the convolution window size of the five independent blocks range from 9 to 729 to capture characteristics on different scales independently). The convolution layer creates 8 copies of the input each created with convolution windows containing different weights but having same size. The output is followed by a max pooling layer which passes the maximum value out of 4 onto the next layer for each of the 8 copies. This is followed by another convolution and max pooling layer. This time from the 8 copies 16 are created during convolution. After this a flattening layer stacks the 16 copies into one one-dimensional vector and a dropout layer is applied. (The dropout layer is not included in the schematic.).
As said before the ANN contains five such convolution blocks which are identical except for the convolution window size. Due to some differences in padding caused by the different convolution window sizes the outputs of the five convolution blocks have different sizes. The first convolution block has an output of size 4144 as can be seen in Figure 13, the other convolution blocks have outputs of size 4048, 3776, 2976 and 544. The outputs of all these convolution blocks are then stacked into one one-dimensional vector of size 15488(=4144+4048+3776+2976+544), followed by three dense layers, another dropout layer and a dense layer as output layer with dimension 3 for the three classes, where class 0 corresponds to 'other', class 1 to 'fringed' and class 2 to 'M-stars'. This can be seen in figure 13 on the right (Again the dropout layer is not included in the schematic.).
As activation functions for all dense and convolution layers we use the ReLU function (A.1.1), except for the output layer, where the Softmax function (A.1.2) is applied. (Pooling and Dropout layers do not require activation functions.) As loss function we use Categorical Cross Entropy (A.2) and as optimizer the Adam algorithm (A.3).

For the problem **'bad splicing'** we try a different approach and apply a more simple but deeper architecture, since 'bad splicing' does not appear with such variety in shape as 'fringing'. A schematic of the architecture can be seen in figure 14.

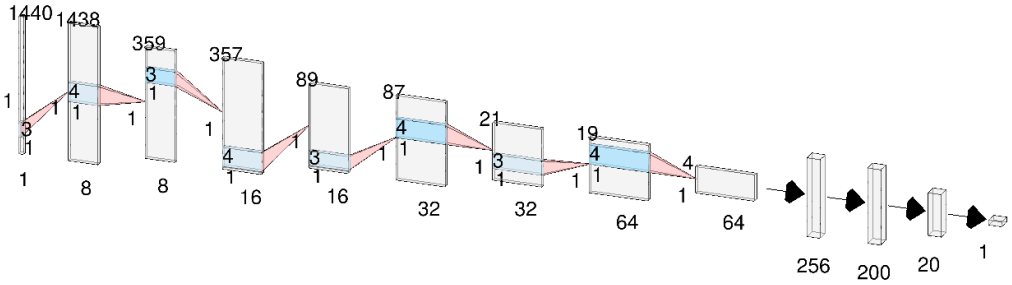


Figure 14: Schematic of the ANN for problem 'bad splicing', created with NN-SVG[7]

The input is a one-dimensional vector of 1440 pixel (the size spectra are cropped to during data preparation). We apply one convolution block consisting of four convolution layers, with a window size of 3 pixel, each followed by a pooling layer, with a window size of 4 pixel. With each convolution the number of filters increases (from 8 to 64), as can be seen by the depth of the vectors in the schematic. After the convolution block a flattening layer stacks the 64 copies into one one-dimensional vector and a dropout layer is applied followed by two dense layers and another dropout layer (Dropout layers again are not included in the schematic.). The last layer, the output layer, is again a dense layer which sorts the inputs into the two classes by binary classification, where 0 corresponds to 'other' and 1 to 'bad splicing'.
As for fringing, for all dense and convolution layers, the ReLU function (A.1.1) is applied as activation function, except for the output layer, where the Sigmoid function (binary equivalent to Softmax (A.1.2) function) is applied. As loss function we use Binary Cross Entropy (binary equivalent to Categorical Cross Entropy (A.2)) and as optimizer the Adam algorithm (A.3).

The final settings like learning rate, batch size, dropout rate and so on as well as model summaries with input and output shape for each layer can be found on our Github page[8].

3.4 Results

3.4.1 Problem: 'Fringing', Task 1

After trying different architectures for the ANN with different hyper parameters and different types of data preparation we chose the data preparation and ANN settings described above.

[7] `http://alexlenail.me/NN-SVG/AlexNet.html`
[8] `https://github.com/juliaziegler/ann_for_galaxy_classification`

First, we analyze performance of our first ANN for the problem 'fringing' as well as whether it can not only find fringed spectra but also distinguish between fringed spectra and spectra of M-stars. For this we first train and evaluate on a balanced training set and later evaluate on an unbalanced test set (with a natural class distribution) as described in 3.1. Two exemplary losscurves of training for 10 epochs can be seen below.

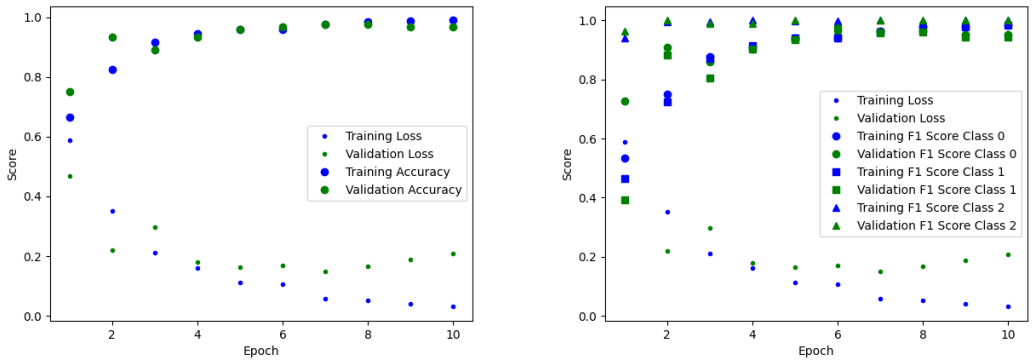


Figure 15: Losscurves: (left:) accuracy and loss, (right:) F1 scores for class 0 ('other'), class 1 ('fringed'), class 2 ('M-stars') and loss, each for training and validation data

In figure 15 the overall accuracy for all three classes and F1 scores for each of the three classes are shown as well as the results from the loss function. Results are shown for training and validation data. Results from evaluation on the test set will be presented later in this section. As one can see with each epoch accuracy and F1 scores increase while loss decreases. As said earlier ANNs try to minimize the loss to improve performance. In this example after only about 7 epochs saturation is reached. Afterwards the test loss decreases further while validation loss increases slightly. This is a sign for slight over fitting. However, still overall accuracy is very good and above 95% for both training and validation data. (For the following runs for the problem 'fringing' we will train on 10 epochs since after this time the loss curve has reached saturation.) When analyzing the F1 scores we find that performance on class 2 is especially good. After the first epoch training and test F1 scores are already above 90%. But the performance on class 0 and class 1 after about 5 epochs is also above 90%
Apparently distinguishing M-stars is fairly easy for our ANN. This is probably due to the fact that spectra of M-stars are very characteristic and look very similar. Fringed spectra on the other hand have big variances across their features, as explained in the introduction.

Since spectra of M-stars show a 'bumpy' structure which can look slightly similar to the sine-like pattern of fringed spectra, we expect that when the ANN is not specifically trained to recognize M-stars, it might mistake them for fringed spectra. To see how the ANN reacts to fringed spectra if it is not specifically trained to recognize them, we train on two more training sets: One training set contains no spectra with M-stars at all. The other training set contains ∼2% spectra with M-stars which are classified as class 0 (this

is about a natural distribution of M-stars, as usually 1-2 % of spectra from the GAMA survey contain M-stars). As a result, these training sets are smaller in total than the one we trained on first, but still contain the same amount of spectra per class.
For this we do 20 training iterations for each of the three training sets. Each training iteration starts from different random weights. This results in the ANN adjusting the random weights in a slightly different way for each training iteration. For a simpler problem or a problem with more training examples this effect might decrease. But for complex problems the loss function has many minima. So with each new training iteration the ANN might adjust more to one of the minima than to another which results in different final weights after each of the 20 training iterations. Afterwards, we evaluate every training iteration on the same test set described in section 3.1. However, when training on two classes instead of three, 'M-stars' are classified into the class 'other'. Results for accuracy of training, validation and test set, as well as F1 scores for test set can be seen in figure 16.

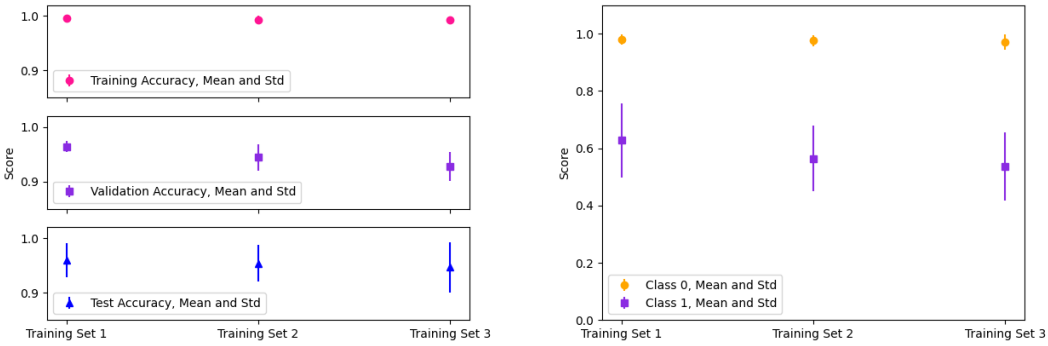


Figure 16: (left:) Training, validation and test accuracy, (right:) Test F1 scores for class 0 ('other') and class 1 ('fringed'), each for three different training sets: training set 1 (even distribution between 'other', 'fringed' and 'M-star'), training set 2 (even distribution between 'other' and 'fringed', no 'M-stars'), training set 3 (even distribution between 'other' and 'fringed', 'other' contain ∼2% 'M-stars'), shown are mean values and standard deviations of 20 training and test iterations.

As one can see, there is no significant difference between training on the three different sets. Mean values for training set 1 are slightly higher than for training set 2 and training set 3, but given the standard deviations this difference is not significant. However, when analyzing which spectra are wrongly labelled as 'fringed', we find that when training on no M-stars at all, between 40-60% of M-stars are labelled as 'fringed' when they should be labelled as 'other'. One reason can be that the ANN does find the 'bumpy' shape of their absorption bands similar to the sine-like shape of fringed spectra. Another reason might be that their spectra do not look similar enough to what the ANN considers as 'other' and that the ANN does not know to which class they belong to and randomly predicts them as either 'other' or 'fringed'. However, when 'other' contain ∼2% 'M-stars' this does not happen and almost all spectra of 'M-stars' are predicted to be 'other'. The conclusion

of this is that to distinguish 'fringed' spectra from 'other', we do not specifically need to train the ANN to also distinguish 'M-stars'. A normal distribution of 'M-stars' among 'other' is sufficient to prevent the ANN from mistakenly labelling 'M-stars' as 'fringed'.
We also analyze spectra wrongly classified as 'fringed', when the training set does contain 'M-stars'. In this case many wrongly classified spectra are spectra which do look slightly fringed. On the other hand 'fringed' spectra, where the fringing is hard to detect because the wavelength is very short or the amplitude of the sine-patterns is low in combination with low signal-to-noise ratio, are often wrongly classified as 'other'. Apart from these transition cases also some spectra are classified wrong where it is not obvious why they were labelled wrongly. But the majority of wrongly labelled spectra is somewhat understandable.
One big problem while training are varying results. Whenever a new training iteration is started and the ANN starts off with new random weights, the performance at the end of training is slightly different. One reason for this probably is that the training set is very small, another reason is that also the test set is quite small considering that due to class imbalances the test set contains only 92 'fringed' spectra. Especially, when analyzing performance on this class via precision, recall and F1 score big variations can be seen. This is expectable since a few wrong classifications in the test set will not have a big impact on the larger class 'other' but on the smaller class 'fringing'. Especially precision varies strongly with each training iteration. The results for precision are strongly dependent on wrong classifications of 'other'. When performance on this class changes slightly a few more or less percent of this bigger class have a big impact on the test set of the smaller class 'fringing'. Results for precision and recall for the 'fringed' class when evaluating on the test set can be seen in figure 17. However, some differences in performance when doing a new training run and starting with new random weights is normal since there are many weights in an ANN which can settle at different values to minimize the loss.

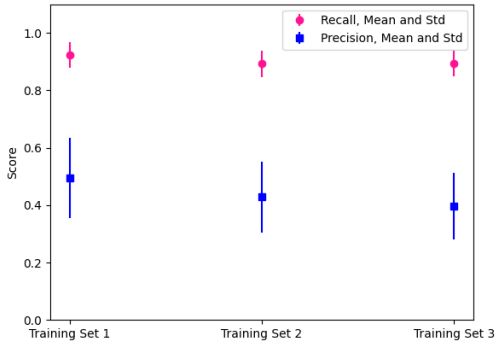


Figure 17: Test precision and recall for class 1 ('fringed'), each for three different training sets: training set 1 (even distribution between 'other', 'fringed' and 'M-star'), training set 2 (even distribution between 'other' and 'fringed', no 'M-stars'), training set 3 (even distribution between 'other' and 'fringed', 'other' contain ∼2% 'M-stars'), shown are mean values and standard deviations of 20 training and test iterations.

As examined before, there are again no significant differences among the three training sets. Precision values seem slightly higher for training set 1 but given the high standard deviations this difference can be due to random fluctuations. For further analysis of the ANN we go on using training set 3, where the class 'other' contains a natural distribution of 'M-stars' and classify for two classes, namely 'other' and 'fringed' instead of three. As can be seen in figure 16, the values for test accuracy (ACC), F1 score for class 0 ($F1_0$) and F1 score for class 1 ($F1_1$) (of the run with highest value for F1 score for class 1) are:

$$\begin{aligned} ACC &\approx 0.9788 \\ F1_0 &\approx 0.9890 \\ F1_1 &\approx 0.6937 \end{aligned}$$

The F1 scores for class 1 seem low as they are still close to 0.5. However, for an unbalanced test set an F1 score of 0.5 does not mean predictions are random, as described in the introduction. F1 score for a random classifier in this case (where the ratio of fringed spectra in test set is about $\sim 2.9\%$) would be $F1_{1,random} = \frac{0.029}{0.029+0.5} \approx 0.055$). This means that a value of $F1_1 \approx 0.6937$ is much better than random chances. To analyze the overall performance instead of performance on a minority class we need to take a look at the ROC curve for evaluation on test data, shown in figure 18.

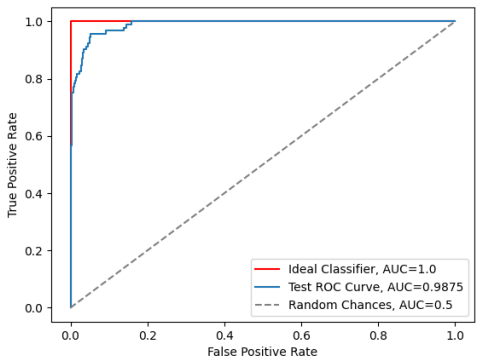


Figure 18: ROC curve for test data

As can be seen in the ROC curve, performance is quite good and the curve is close to the ideal line. This shows that the ANN is in fact working and makes far better predictions than random. To analyze how the performance on the minority class can be improved, specifically how the F1 scores of class 1 can be maximized, we take a look at the effect of the class distribution in the next section.

3.4.2 Problem: 'Fringing', Task 2

In this section we analyze how the class distribution in the training data affects the performance when evaluating on the extremely unbalanced test set. Since we find that for performance it does not make a difference whether we train the ANN specifically to recognize 'M-stars' or use a natural distribution of 'M-stars' included in the class 'other', we decide to change for the latter option as training data and go on with classifying for two classes ('other' and 'fringed') instead of three.
From the pool of the training samples described in section 3.1, we create new training sets of fixed size but with varying class distribution. The total training set size is 400. We start from a balanced training set containing 200 'other' and 200 'fringed' spectra and vary up to a distribution with 390 'other' and 10 'fringed' spectra. The test set again is the same one as described in 3.1.
Once more we do 20 iterations and plot mean values and standard deviation against $1-R$, where R is the ratio of fringed spectra in the training set. Results can be seen in figure 19.

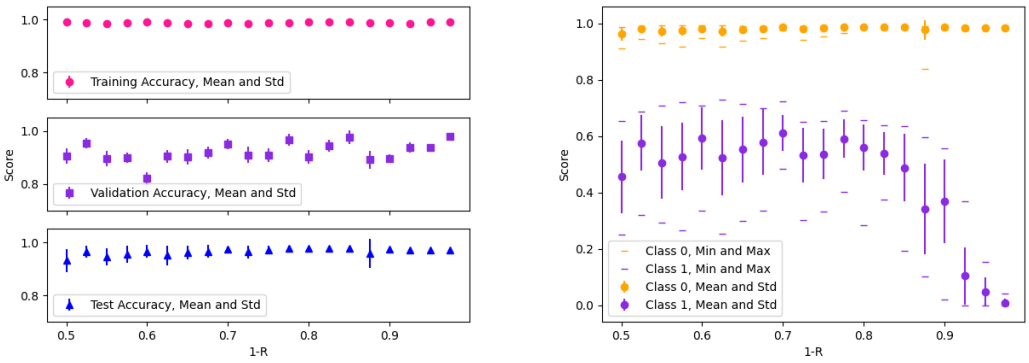


Figure 19: (left:) Training, validation and test accuracy, (right:) Test F1 scores for class 0 ('other') and class 1 ('fringed'), both plotted against $1-R$, where R is the ratio of 'fringed' spectra in training set, shown are mean values and standard deviations of 20 training and test iterations.

As one can see, validation accuracy, which has the same class distribution like the training set jumps around a constant value of $\sim 92\%$, while test accuracy does not show significant changes and always stays around a value of $\sim 96.5\%$. Overall its mean value seems to increase slightly while the standard deviation decreases, but this effect is barely noticeable and not significant.
As mentioned earlier, accuracy is not always the best way to evaluate performance. So when looking at the plot on the right we see that test F1 scores for the 'other' class constantly stay at a high level, while the standard deviation decreases with a lower ratio of 'fringed' (so a higher ratio of 'other') in the training set. This is similar to what we see in the plot on the left. Main changes are visible on the right when evaluating F1 scores for the 'fringed' class. Mean values first seem to increase slightly, which is difficult

to interpret correctly since the standard deviation on this class is a lot higher due to it being much smaller compared to the 'other' class. The highest F1 score was reached at $1-R=0.625$ and is $\sim 73.1\%$, which is a good value given the high class imbalances. At values where $1-R>0.8$ performance on the 'fringed' spectra starts breaking down. At this point the class imbalance in training data apparently is too high to still learn correctly how to distinguish 'fringed' spectra or the total number of fringed spectra is too low to learn their characteristics. To analyze test F1 scores for the class 'fringed', test precision and recall are plotted in figure 20.

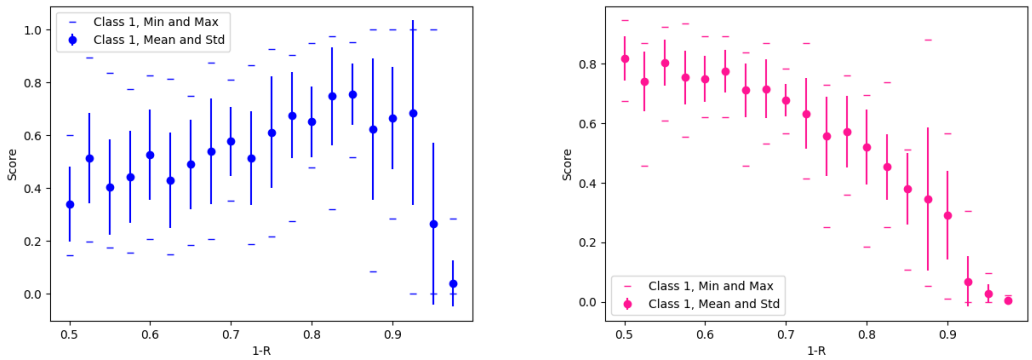


Figure 20: (left:) Test precision, (right:) Test recall, both for class 1 ('fringed'), plotted against $1-R$, where R is the ratio of 'fringed' spectra in training set, shown are mean values and standard deviations of 20 training and test iterations.

As can be seen in the figure on the left, test precision increases when the ratio of 'fringed' spectra in training data decreases. This is due to the fact that when more 'other' than 'fringed' are in the training set, the ANN is biased towards 'other' and tends to classify more spectra as 'other'. As a result, there are not many spectra wrongly classified as 'fringed', so there are fever false positives which increases precision. On the other hand, as can be seen on the right, recall decreases with decreasing ratio of fringed spectra, because an opposite effect shows. When the ANN is biased towards 'other' fewer spectra in general are predicted as 'fringed' which results in fewer true positives and more false negatives which in return lowers recall.
To sum it up, ANNs train best on balanced data, as explained in the introduction. However, when test data is heavily imbalanced, a slight imbalance in training data might improve performance on the minority class, or at least does not decrease performance until a certain degree of imbalance in training data, which in this case is $1-R>0.8$ for a test set containing $\sim 3\%$ examples of the minority class. As said before, the highest score was reached at $1-R=0.625$. Results for test accuracy (ACC), F1 score for class 0

($F1_0$), and F1 score for class 1 ($F1_1$) are:

$$ACC \approx 0.986$$
$$F1_0 \approx 0.9928$$
$$F1_1 \approx 0.7305$$

Furthermore, it depends on the application and whether a higher precision or recall is desired, if a slight imbalance in training data is useful or not.
It remains to be said that we could not get rid of the performance deficit on the minority class as this is mainly due to the class imbalance.

3.4.3 Problem: 'Fringing', Task 3

Finally, we analyze how training set size affects performance of our ANN for 'fringing'. To do this we use a balanced training set starting with only 20 examples per class and go up to 400 examples per class. These spectra are again selected from the training set pool described in 3.1. The test set still is the one described in 3.1.
Again we do 20 iterations and plot mean values and standard deviation against N, where N is the number of spectra per class in the training set. The results for accuracy and test F1 scores can be seen in figure 21.

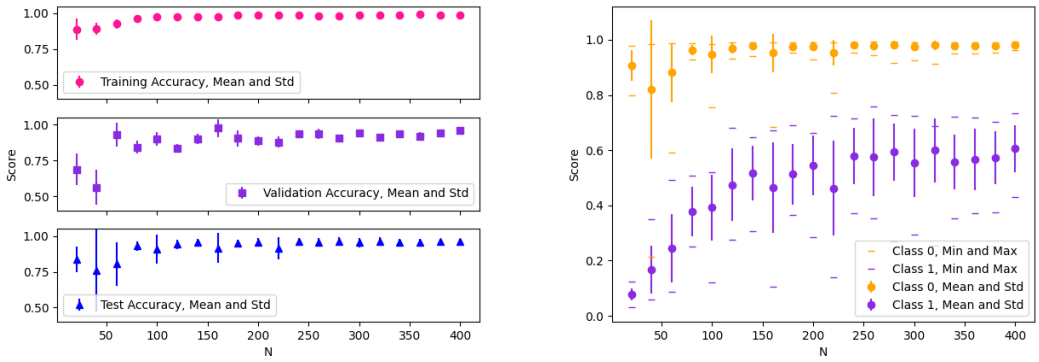


Figure 21: (left:) Training, validation and test accuracy, (right:) Test F1 scores for class 0 ('other') and class 1 ('fringed'), both plotted against N, where N is the total number of examples per class in training set, shown are mean values and standard deviations of 20 training and test iterations.

As can be seen on the left, performance completely breaks down for training sets with less than 60 examples per class. In the figure on the right one can see that performance on the minority class 'fringed' already breaks down when trained on less than 150 examples per class. For $N = 220$ there are still some results of the 20 iterations with very low performance while some tries deliver results with a good performance. Overall, for

more than 200 examples per class the performance does not increase dramatically, we expect that we need a much bigger training set to get significantly better results. Furthermore, for such big standard deviations as in task 1, 2 and 3 it is necessary to do more than 20 iterations for each setting, but this would increase computation time dramatically.

3.4.4 Problem: 'Bad Splicing', Task 4

In this section we analyze performance of the second ANN for the problem 'bad splicing'. We first train and evaluate on a balanced training set and later evaluate on an unbalanced test set (with a natural class distribution) as described in 3.1. Two losscurves of training for 12 epochs can be seen below.

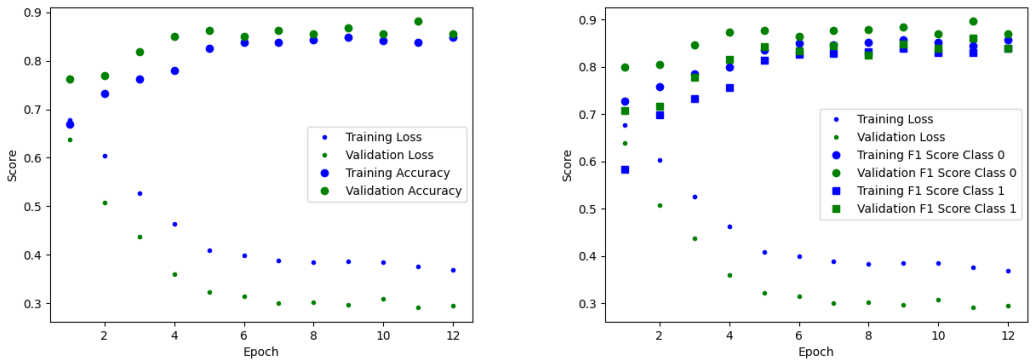


Figure 22: Losscurves: (left:) accuracy and loss, (right:) F1 scores for class 0 ('other') and clsss 1 ('fringed'), each for training and validation data

In figure 22 the overall accuracy for both classes and F1 scores for each of the two classes are shown as well as results from the loss function. Results are shown for training and validation data. Results from evaluation on the test set will be presented later in this section. As can be seen in the losscurves saturation is reached quickly and after the first two epochs performance increases very slowly. In this case we use a different approach to define when to stop training, namely when validation loss does not increase more than 0.01 for more than 5 epochs, which in this case is reached after 12 epochs.
Results for validation data jump slightly but are close to results of training data. This indicates that there is not much overfitting. After the last epoch results for accuracy and F1 scores are above 83% for both training and validation data. A better performance is desired but could not be achieved for this problem. This is probably due to the fact that distinguishing between spectra, which show bad slicing and spectra which do not, is quite difficult since there are many transition cases.
Next, we present results of evaluation on the test data. For this problem again we evaluate performance mainly by accuracy and F1 scores but also show the ROC curve as a way to measure and compare performance.

For evaluation on the test set we receive the following values for accuracy (ACC), F1 score for class 0 ($F1_0$) and F1 score for class 1 ($F1_1$):

$$ACC \approx 0.8880$$
$$F1_0 \approx 0.9354$$
$$F1_1 \approx 0.5765$$

The F1 score for class 1 seems especially low. This is due to the class imbalance and the impact of the false positives, as explained earlier, and still above random ($\sim 11\%$ of test data belong to the class 'bad splicing' so F1 score for a random classifier would be $F1_{1-random} = \frac{0.11}{0.11+0.5} \approx 0.18$). However, still a better overall performance is desired. The ROC curve for evaluation on the test set can be seen in figure 23.

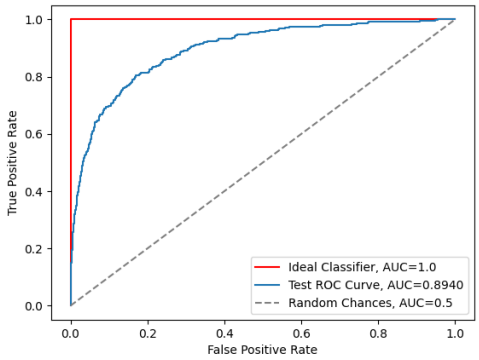


Figure 23: ROC curve for test data

As can be seen by the ROC curve, performance on the test set is far better than random but also quite far from the performance of an ideal classifier. This also confirms that the poor F1 scores are due to the imbalance in training set and that the overall performance is not that bad.
For this problem, we also check wrongly classified spectra. Many of wrong classifications are cases where even by eye it is difficult to decide whether the spectrum does have a bad splice or not. However, some spectra where bad splicing is clearly visible are still classified wrong. On the other hand, this might be understandable considering that in training data also a clear distinction was difficult and some spectra which could have a bad splice maybe were not correctly classified as those.
To achieve better performance, more time would be needed to try different data preparation techniques or a completely different architecture for the ANN and different parameters. Also as said earlier the data might not be ideal for a classification problem. In the discussion we will suggest other steps that could be taken to increase performance, which could not be done in this work due to time limitations.

4 Discussion

In this work we have tested two different ANNs for classification of galaxy spectra.
The first ANN classifies spectra for three classes namely 'other', 'fringed' and 'M-star', which later was changed to two classes namely 'other' and 'fringed'. As it was believed, that training the ANN specifically to distinguish 'M-stars' from 'fringed' spectra would increase performance. This, however, was not the case, but as a side effect the ANN performed very well on distinguishing 'M-stars'. We tested the ANN for how the class distribution in the training set would affect the performance if the ANN is evaluated on a heavily imbalanced test set. The results show that for overall performance it is not necessary to train on imbalanced data, but depending on the application, if for example precision of the minority class should be maximized, it might be useful to train on slightly unbalanced training data. However, as other works have shown, ANNs perform best when trained and evaluated on balanced data. If evaluation can not be done on balanced data, as usually when the ANN is applied to a real problem, one does not know beforehand which example belongs to which class. A slight performance deficit on minority classes might have to be accepted as this is due to statistics. Finally, we tested how training set size affects the performance. However, the maximum training set size was probably still too small, as performance increased very slowly with larger training sets.
Overall performance of the ANN is quite good given that we trained on a training set with only 400 examples per class. The highest value for AUC when trained on balanced data was $AUC = 0.9875$.
The second ANN classifies spectra for two classes namely 'other' and 'bad slicing'. Performance of this ANN was about average and much better than a random classifier, however, performance was worse than for the first ANN. The highest value for AUC when trained on balanced data was $AUC = 0.8890$.
One main problem for both ANNs is that there is a continuous transition between the classes, and defining clear boundaries is difficult when sampling training and test data by eye. As a result, the ANN cannot learn properly what the boundaries are, which limits performance. Another limitation might be that for both ANNs the class 'other' contains spectra of all kinds which do not belong to the class 'fringed' (or 'bad splicing'). So the ANN also learns boundaries for how 'other' spectra should look like when in reality this is not desired as these spectra can show wide varieties. One possible approach to solve this is one-class-learning where an ANN is trained on only one class and does not learn characteristics of examples which belong to other classes. However, for these kind of ANNs strong boundaries for the training class are needed for good performance. Whether this is a better approach than the one tried in this work needs to be tested in the future.
Another approach to increase performance is by increasing the size of the training set. This could be done by simply sampling more spectra, which is time consuming. Another solution is data augmentation, where real data is manipulated slightly, for example by

mirroring or shifting the data or rotating it by an angle, for example by multiplying the flux values with a linear function. This could be an interesting project for the future as well as creating synthetic data to train on. However, in both cases manipulated and synthetic data still need to represent real data in a realistic way.
There are also other ways to sort out 'fringed' spectra and those with a 'bad splice'. For example, random forests or variational encoders are interesting solutions to detect outliers. This was already tested on GAMA spectra and worked well to detect 'fringed' and 'bad spliced' spectra [9].
Another interesting approach to applying machine learning is automated machine learning. This is especially interesting for non-experts as it automates the process of choosing a model and creating an architecture. As a non-expert it can be difficult to decide what model of machine learning to use and how to construct it. Currently there is no such thing as a 'recipe' for machine learning or ANNs, so having an automated process might be very useful for non-experts. However, to learn how ANNs work it is a good practice to design the network yourself to increase performance by trial and error. When engaging which ANNs one can quickly learn to understand how the ANN works and it is not a complete black box as some people might think. Furthermore, there are in fact some standard approaches even if every problem is different and requires a different solution.

5 References

[1] Gama webpage. `http://www.gama-survey.org/`.

[2] Bodo Baschek Albrecht Unsöld. *Der neue Kosmos, Einführung in die Astronomie und Astrophysik*. Springer, Berlin, Heidelberg.

[3] Lutz Wisotzki Alfred Weigert, Heinrich J. Wendker. *Astronomie und Astrophysik : Ein Grundkurs*. John Wiley Sons, Incorporated, 2009.

[4] F. Allard, M. Scholz, and R. Wehrse. Molecular opacities in M-star atmospheres. , 23:203–215, March 1992.

[5] Andrew P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[6] Andries P. Engelbrecht. *Computational Intelligence: An Introduction*. John Wiley Sons Ltd, 2007.

[7] Andrew Estabrooks, Taeho Jo, and Nathalie Japkowicz. A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence*, 20(1):18–36, 2004.

[8] Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861 – 874, 2006. ROC Analysis in Pattern Recognition.

[9] Job Formsma and Teymoor Saifollahi. In search of the weirdest galaxies in the universe, 2020.

[10] A. M. Hopkins, S. P. Driver, S. Brough, M. S. Owers, A. E. Bauer, M. L. P. Gunawardhana, M. E. Cluver, M. Colless, C. Foster, M. A. Lara-López, and et al. Galaxy and mass assembly (gama): spectroscopic analysis. *Monthly Notices of the Royal Astronomical Society*, 430(3):2047–2066, Feb 2013.

[11] C Y Hui, Jongsu Lee, K L Li, Sangin Kim, Kwangmin Oh, Shengda Luo, Alex P Leung, A K H Kong, J Takata, and K S Cheng. Searches for pulsar-like candidates from unidentified objects in the third catalog of hard fermi-lat sources with machine learning techniques. *Monthly Notices of the Royal Astronomical Society*, 495(1):1093–1109, Apr 2020.

[12] Jr. Kennicutt, Robert C. The Integrated Spectra of Nearby Galaxies: General Properties and Emission-Line Spectra. , 388:310, April 1992.

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[14] Jorma Laurikkala. Improving identification of difficult small classes by balancing class distribution. In Silvana Quaglini, Pedro Barahona, and Steen Andreassen, editors, *Artificial Intelligence in Medicine*, pages 63–66, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[15] J. Liske, I. K. Baldry, S. P. Driver, R. J. Tuffs, M. Alpaslan, E. Andrae, S. Brough, M. E. Cluver, M. W. Grootes, M. L. P. Gunawardhana, and et al. Galaxy and mass assembly (gama): end of survey report and data release 2. *Monthly Notices of the Royal Astronomical Society*, 452(2):2087–2126, Jul 2015.

[16] David Masko and Paulina Hensman. The impact of imbalanced training data for convolutional neural networks, 2015.

[17] Matthew A. Petroff, Graeme E. Addison, Charles L. Bennett, and Janet L. Weiland. Full-sky cosmic microwave background foreground cleaning using machine learning, 2020.

[18] Fjodor van Veen. Neural network zoo prequel: Cells and layers, 2017.

[19] Joseph Yacim and Douw Boshoff. Impact of artificial neural networks training algorithms on accurate prediction of property values. *Journal of Real Estate Research*, 40:375–418, 11 2018.

A Appendix

A.1 Activation Functions

A.1.1 Rectified Linear Unit Function (ReLU)

One on the activation functions used in this work is the ReLU function. It is defined as:

$$f(x) = max(0, x) \tag{17}$$

Where x is the input of the neuron. So a neuron only 'fires' when the input is larger than 0.

A.1.2 Softmax Function

Another activation function being used is the Softmax function. It applies the logistic (or Sigmoid) function to multiple dimensions.
The Sigmoid function is defined as:

$$f(x) = \frac{1}{1 + \exp x} \tag{18}$$

Where x is the input. For higher dimensions the Softmax function is defined as:

$$f(\vec{x})_i = \frac{\exp x_i}{\sum_{j=1}^{K} \exp x_j} \quad \vec{x} = (x_1, x_2, ..., x_K) \tag{19}$$

Where $\vec{x}$ is the input vector. As one can see the sum of all components $f(\vec{x})_i$ equals 1.
A bias can be applied to all activation functions however in this work the bias equals 0 in all cases.

A.2 Categorical Cross Entropy

When using categorical cross entropy as loss function the cross entropy between the labels and predictions are calculated. The cross entropy is defined as:

$$Loss = -\sum_{i=1}^{N} y_i \cdot \log \hat{y}_i \tag{20}$$

Where N is the dimension of the output (the number of classes in our case), $\hat{y}_i$ is the i-th value of the predicted output and y_i is the i-th value of the corresponding label.
In our case the labels are one hot vectors. This means one component has value 1 and all other components have value 0, the position of the value 1 indicates the class. The predicted output should match the label but in reality contains floating point values in the range $0 < y_i \leq 1$. The position of the maximum value indicates the class.

A.3 Adam Algorithm

The optimizer used in this work is based on the Adam algorithm. This is a gradient-based optimization algorithm which is computationally efficient and works well on problems with many parameters as well as problems with noisy gradients [13].
The settings applied here are the ones recommended in Kingma et al., 2014 [13] except for the learning rate/step size which is recommended to be one 0.001, whereas we received better results with 0.0001 for our problem.
Within the framework of this Bachelors Thesis we can not go into detail, explaining the Adam algorithm. However this is not necessary for the understanding of this work.

Eidesstattliche Versicherung

Ich versichere hiermit ehrenwörtlich, dass ich meine vorliegende Abschlussarbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Die Arbeit wurde vorher nicht in einem anderen Prüfungsverfahren eingereicht und die eingereichte schriftliche Fassung entspricht derjenigen auf dem elektronischen Speichermedium. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht. Ich bin damit einverstanden, dass die Bachelorarbeit veröffentlicht wird.

Ort, Datum

Unterschrift