|epcc|

Developing Rapid Classification Algorithms for Identifying Quasars in Time Series Data

Xinyue Sheng (B156924)

August 21, 2020

Abstract

Transient event and variable object classification plays an important role in astronomy. Quasars, galaxies with an actively accreting black hole at their center, are known to vary in luminosity. Of particular interest is identifying quasars with possible gravitational waves (GW) sources using data from next generation surveys, such as the Legacy Survey of Space and Time (LSST) survey. However, quasars are difficult to classify, since their light curves do not show regular, if any, variability on short timescales. In this research, we focus on the classification of quasars based purely on their light curves. We utilise and investigate survey data from the Sloan Digital Sky Survey (SDSS) and the Zwicky Transient Factory (ZTF). Due to the irregular sampling of quasar light curves, the observational cadence is considered in the classifier design. Three input formats: $Group$, $Season$ and $Simple$ were implemented and show different results in the evaluation. We implement several Recurrent Neural Networks (RNN), using three general architectures: LSTM, GRU and SimpleRNN. After testing over the range of input formats and features, network types and training parameters, we find that the g and r bands have the most obvious characteristics in distinguishing quasars and non-quasars. Additionally, LSTM networks with the $Group$ input format always gain the highest recall in combinations of different bands among all input formats. We find that the optimal network architecture is 4 GRU layers with 64 neurons for each, and the input data is the combination of g and r bands with $Group$ format. The best AUC reaches 0.893. Our research thus gives a quantifiable way forward for the observational strategy and analysis for quasars in the LSST data stream.

Contents

List of Figures

List of Tables

Acknowledgements

I sincerely thank my tutors Dr. Nicholas Ross and Dr. Darren White. Thank you all for your support, patience, trust and encouragement.

To Dr. Nicholas Ross, you have always supported and encouraged me from the beginning to the end of this project. You have taught me a wealth of astronomy knowledge and made me a beginner from a layman. I am very grateful to you for taking care of me in my study and life, and giving valuable suggestions for my future development. I feel very lucky to be able to do this interesting project under you guidance.

To Dr. Darren White, thank you very much for your key guidance on this project for eight months. During the six months when COVID-19 is raging, we have kept in contact every week and made great progress. This is inseparable from your support and encouragement.

To my personal tutor Jane Kennedy, thank you very much for your great help and encouragement in my study and life.

To my program director Ben Morse, thank you very much for your support in my study and life. Thank you for always helping me deal with many issues.

To my parents, thank you very much for your love and support.

1 Introduction

Astronomy is an old observation-based science, dating back thousands of years. In the ancient days, the wisest scholars observed the Sun, the Moon and stars by eye, recorded those objects' positions and made astrolabes (an elaborate inclinometer used to recognize stars and planets and calculate the local latitude[1]) manually. By looking at the starry sky, many scholars proposed different world system theories, such as the Ptolemaic system (the Earth is at the centre of the Universe) and Heliocentrism (the Sun is at the centre of the Universe).

The mystery of the Universe always attracts humans deeply and encourages us to explore. The invention of the astronomical telescope in the early 17^{th} Century broke the research mode of naked-eye observation and opened a new era of astronomical science. The visible light focused by the camera lens of the telescope gives scientists an insight into various objects' shapes and distances from the Earth. After two centuries of scientific development, the discovery and research of spectroscopy made astronomers realize that visible light occupies only a small place in the entire Electromagnetic (EM) Spectrum, which also includes microwaves, infrared, ultraviolet, X-rays and gamma-rays. Emission and absorption line features in a spectrum can deliver more information about distant objects, such as their chemical compositions, structures, atmospheres and redshifts. As such, spectroscopy plays a vital part of astronomical research.

With the invention of more sophisticated and diverse telescopes, astronomy has been developing rapidly. Optical telescopes, radio telescopes, gamma-ray telescopes and other types of telescopes are now all widely applied in different branches of astronomical study. They are able to detect specific wavelengths and receive information from different types of objects. The need for multiple wavelength information of objects promoted the invention of multi-band telescopes. For example, *Hubble Space Telescope* is designed to collect infrared light, visible light, and ultraviolet light. It was launched to the Low Earth Orbit to catch the broader information of celestial bodies.

To complement deep observations (as performed by *Hubble Space Telescope*), astronomy research also uses large-angle sky survey observations which cover large parts of the celestial sphere. With the rapid development of computer and network communication technologies, optical, infrared, radio, gamma-ray surveys were conducted respectively, and billions of galaxies, stars and planets' information have been detected, processed and stored. The data products of these sky surveys are a major resource for researchers in physics and astronomy today.

Starting in 1949, the 48-inch Oschin Schmit telescope on Palomar Mountain, California California carried out the Palomar Sky Survey (POSS-I). This optical survey used two sensitive square photographic plates with Kodak 103a emulsions: one red-sensitive Kodak 103a-E plate and another blue-sensitive Kodak 103a-O plate[2]. It covered the celestial sphere from North Celestial Pole to -33° declination, with declination being the Celestial Sphere equivalent of geographical latitude and the generated images can be converted into digital format[3]. After a few years, due to the considerable technological improvements on the resolution and sensitivity of photographic emulations, the quality of plates was much better than before. Therefore, the second Palomar survey, POSS-II, was carried out. This new survey took advantage of the upgraded telescope and photographic plates and utilized three passbands - blue, green and red to image the entire northern sky. The digitization of POSS-II photographic sky maps were constructed as the Palomar Digital Sky Survey (DPOSS) after being processed and sorted as catalogs. This achievement was regarded as the core astronomical atlas and referenced by researchers for many decades[4].

Compared to POSS surveys, another series of the optical surveys - the Sloan Digital Sky Surveys (SDSS), which began observations in 2000, shows better performance in many ways. This survey uses a 2.5-m wide-angle optical telescope at Apache Point Observatory in New Mexico, United States[5]. It covers half of the northern sky and utilizes the state-of-the-art Charged-Couple Devices (CCDs) collecting and converting light flux to digital signals, and then sending back the data to efficient computer software.

[1] https://en.wikipedia.org/wiki/Astrolabe
[2] https://en.wikipedia.org/wiki/National_Geographic_Society_Palomar_Observatory_Sky_Survey
[3] http://skyserver.sdss.org/dr5/en/proj/advanced/skysurveys/poss.asp
[4] https://www.astro.caltech.edu/ george/dposs/dposs_pop.html
[5] https://www.sdss.org/

Five passbands - u, g, r, i, z are designed to bring more magnitude information of the distant objects. In the initial survey, the SDSS[6] produced 8000 square degrees' deep multi-colour imaging and measured the spectra of over 0.7 million celestial objects. In the next few surveys, through continuous development and cooperation, SDSS successfully imaged the northern sky, and detected hundreds of supernova and other events. At the same time, more SDSS-based surveys were carried out, such as eBOSS[7], APOGEE[8] and MaNGA[9].

The Zwicky Transient Facility (ZTF) survey[10] started in 2017 aims to detect transients with the time domain. A new camera is set up with a 47-square-degree field of view on the telescope used for POSS. This telescope scans over 3750 square degree per hour to the faith of 20.5 mag. In this way, it is able to discover young supernovae, variable stars, binaries, AGN and asteroids.

The Large Synoptic Survey Telescope (LSST)[11], renamed as Vera C. Rubin Observatory in Chile will start to explore the Universe in 2021. Compared to any existing sky surveys, LSST will have an unique impact on the astronomical research field. This survey has four main goals: Making an inventory of Solar System, Mapping the Milky Way, Detecting transients in the optical sky, and Research on Dark Energy and Dark Matter. With up-to-date detector and photometry technologies , its effective and enormous étendue (319 m^2deg^2) of the telescope means that it is able to quickly detect the objects in the sky with the aimed depth. Six passbands - u, g, r, i, z, y with the first single exposure longer than 20s and up to 1000 times detection to the same position promise high-quality imaging through the sky, even imaging of faint events. Many surveys in different fields can dig into the scientific missions based on the production of LSST at the same time. Following the large-scale survey, the data volume of the production will also be large. It is projected that LSST will generate around 14TB of data per night. Consequently, astronomy science is facing the difficulties of taking advantage of these data effectively and efficiently and managing them properly.

The era of big data in astronomy has arrived. The research mode of astronomy has changed from demand-driven to data-driven. Previously, research methods were limited by data shortage. Due to the lack of good quality data and powerful computers with satisfying computing capability, astronomers often chose some sources to observe and applied simple algorithms to deal with the raw data manually. It often takes a few hours or even days to obtain the spectra for targeted objects, and then they would spend much time and effort on preprocessing, programming, analysing as well as using the their own professional experience to build models. This procedure was enough for dealing with astronomical research at that time[22]. However, the development of large-scale sky surveys leads the astronomy to data-intense science, which means that millions of celestial objects' data are being generated continuously per night. This gives astronomy both opportunities and challenges. Astronomers are able to utilize abundant and high-quality data. It becomes essential to have powerful high performance computing facilities and efficient data processing software to handle those data.

Modern astronomy now has a strong tie with computer science. More data scientists are working with big data coming from sky surveys. These data will have multiple bands and epochs, but they are often with noise. The vast majority of objects will be detected close to the surveys' sensitivity limit. Rapid, reliable and repeatable methods are desired to perform near real-time classification on astronomical data. Complex and rapid algorithms and systems are applied in this field. One of the most important applications is machine learning technology, such as Neural Network, Random Forest and Naïve Bayes. In recent years, Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) are often used to classify imaging data, predict light curves and coming events, and high accuracy is gained.

For this project, Recurrent Neural Network (RNN) is applied to build a classifier to recognize quasars based on their light curves. The main goal is to recognize quasars which may hold Gravitational Wave (GW) events among all possible celestial objects.

[6]https://www.sdss.org/surveys/#sdssi-ii
[7]https://www.sdss.org/surveys/eboss/
[8]https://www.sdss.org/surveys/apogee/
[9]https://www.sdss.org/surveys/manga/
[10]https://www.ztf.caltech.edu
[11]https://www.lsst.org/about

The organization of this dissertation is as follows. Chapter 2 gives further astronomical background to quasars, Gravitational Waves, RNN, and photometric classification. Chapter 3 reports on the data investigations and the motivation and decisions behind using the SDSS Stripe 82 quasar-targeted dataset. In Chapter 4, research methods are presented in detail, and in Chapter 5, the results of all tests are displayed. Chapter 6 is a discussion of the findings and Chapter 7 gives numerous ways forward. Chapter 8 gives the conclusion. The Acronyms Explanation, Astronomy Terms Explanation and Sky Survey Summary are shown in Table 1-3.

Table 1: Acronyms Explanation

Acronyms	Definition
AGN	Active Galactic Nucleus
BBH	Binary Black Holes
SNe	Supernovae
GW	Gravitational wave
EM	Electromagnetic
SMBH	Supermassive Black Hole
MJD	Modified Julian Day
Stripe 82	300 $\deg^2$ equatorial field of sky
POSS	the Palomar Sky Survey
ZTF	Zwicky Transient Facility
SDSS	the Sloan Digital Sky Survey
LIGO	the Laser Interferometer Gravitational-Wave Observatory
LSST	the Large Synoptic Survey Telescope, renamed as Vera C. Rubin Observatory
RNN	Recurrent Neural Network
GRU	Gated Recurrent Unit, which is a gating mechanism in RNN architecture
LSTM	Long Short-term Memory, which is a type of neural in RNN architecture
GPR	Gaussian Process Regression
RBF	Radial Basis Function, shown in Formula 1

Table 2: Astronomy Terms Explanation

Term	Definition
étendue	also known as throughput,the ability to collect light, which is related to the area of the source and the solid angle[12]
magnitude	a measure of brightness of a celestial body in a band[13]
passband/band	the optical filter set up on telescopes' cameras. Different band can detect different range of wavelengths
redshift	the increase of the wavelength due to Doppler' effect. Because the universe is expanding, distant celestial objects often have large redshifts
light curve	a curve produced by magnitudes of a celestial body in each band ordered by their observation time
$M_\odot$	the mass of the Sun

[12]https://en.wikipedia.org/wiki/Etendue
[13]https://en.wikipedia.org/wiki/Magnitude_(astronomy
[14]http://articles.adsabs.harvard.edu/cgi-bin/nph-iarticle_query?bibcode=1959ASPL....8..121A
[15]https://www.astro.caltech.edu/ george/dposs/dposs_pop.html

Table 3: Sky Survey Summary

Name	POSS-I	POSS-II	SDSS	ZTF	LSST
Launch year	1949	1985	2000	2017	2021
Sky area	celestial sphere from north celestial pole to -33 declination	entire Northern sky	half of Northern sky	entire Northern sky	entire sky
Bands	red, blue	blue, green, red	u, g, r, i, z	g, r	u, g, r, i, z, y
Depth(mag)	22(B)	22.5-19.5	22.5	20.5	26.5
Object num	thousands of clusters and associations of stars, 86 planetary nebulae[14]	over 50 million galaxies, half a billion stars including tens of thousands of quasars[15]	10^9	10^9	37×10^9
Duration(years)	9	10	20	2	10
Nightly alert rate	-	-	-	10^6	10^7

2 Research Background

2.1 AGN and Quasars

An Active Galactic Nucleus (AGN) is a compact region at the center of a galaxy with high luminosity[16]. AGN with extremely high luminosity are called quasars, also known as QSO.

The first recognized quasar is 3C 273, classified in the Third Cambridge Catalogue of Radio Sources (3C) in 1959[20]. 3C 273 is a bright radio source, which attracted astronomers' attention. In the 1960s, scientists had identified distant radio galaxies with large redshifts, but they did not have any observational evidence for the existence of black holes. 3C 273 was regarded as a bright nearby star or a radio galaxy because it is very luminous (absolute magnitude is up to -26.7) and large redshift (z=0.158)[19]. However, since its spectrum is different from those of any known star, astronomers named objects like it quasi-stellar radio sources.

Astronomers realized that the quasar is an Active Galactic Nucleus (AGN), and there is a massive black hole (10^6-10^{10} $M_\odot$) in the center of it with a heated accretion disk around. The reason behind its strong luminosity is that the materials such as dust and gas plunging into the black hole form a gradually heated spiral accretion disk around the center, and emit enormous radiations. Compared to a galaxy, a typical AGN has a broader range of luminosity among all wavelengths with the peak in ultraviolet due to the the disk's extremely high temperature[17], which can easily be observed by telescopes.

The X-ray emission is proved to be periodic on the order of months to years, which is called quasi-periodic oscillation (QPO)[14]. Additionally, a damped random walk (DRM) model is often used to describe the quasar's optical variability. It indicates that the quasar's light curve might be arbitrary in the short term but have a tendency to rise or fall in the long term[9]. These features can help in the recognition of quasars in the design and training process of the classifier.

2.2 Gravitational Waves

In recent years, a new window on the Universe has been opened to us. Gravitational waves (GW) are the ripples in space-time produced when objects move with high accelerations. On 14 September 2015, the first gravitational waves (GW150914) were observed by the Laser Interferometer Gravitational-Wave Observatory (LIGO)[18]. GW150914 was the result of the merger of a binary black hole (BBH). This great discover confirmed Einstein's General Theory of Relativity. More importantly, in addition to electromagnetic wave (EM), it reveals a new way of observing the Universe. Receiving the alerts of GW detections, scientists can know that there are precious transients happening, such as Binary Black Hole (BBH) mergers, Binary Neutron Star (BNS) mergers and supernovae explosions. A potential channel for GW events is for BBHs to reside and preferentially merge in quasars, in the AGN accretion disk[7]. This provides a clear path to search for GW sources.

Such events give astronomers great opportunities to observe astronomical phenomena, and understand the formation of new stars as well as their internal compositions. This information helps human beings understand the Universe and the environment around our planet to a large extent. Many observatories, including LIGO and the Virgo Interferometer[19] dare detecting gravitational waves and generating real-time alerts to astronomers, allowing follow-up observations of exciting, new GW events.

Recognizing possible quasars holding GW events with high confidence among all celestial bodies in a targeted area (such as in ZTF Skymaps shown in Lasair[20]) is urgent for further observation of BBH merger events, since such events are thought to happen rapidly. Given the case that obtaining spectra requires a great amount of labor and professional knowledge and there are billions of stars, galaxies

[16] https://en.wikipedia.org/wiki/Active_galactic_nucleus
[17] Introductory to Astrophysics Coursebook, written by Catherine Heymans and Andy Lawrence
[18] https://www.ligo.caltech.edu/
[19] https://www.virgo-gw.eu/
[20] https://lasair.roe.ac.uk/skymap/

and AGN detected, it is not possible to have all objects' spectra information provided for us to pick out quasars. On the other hand, Large Synoptic Survey Telescope (LSST) won't provide spectra, however, it will provide magnitudes in each detection and passband. **Our project therefore attempts to classify quasars based purely on their light curves, which could be vital for future GW+AGN detections.**

In order to recognize the most probable host galaxies/quasars holding the GW events quickly, the classification algorithms in the Machine Learning field can come in handy, which involves Recurrent Neural Networks. This is the basic algorithm for the classification in this research.

2.3 Recurrent Neural Networks

The Recurrent Neural Network (RNN)[18] is a type of neural network that takes sequences as input, recurses in the order of the sequence, and all recurrent units are connected in a chain[6]. It is designed for processing sequential data, and widely applied in speech recognition, text generation and machine translation.

In the initial design, an ordinary RNN unit can only focus on the correlation between data points at near steps in the sequence and the weight of transferring each step's information is fixed. Such a simple network doesn't perform well in complex applications since many real sequences don't have short-term features but some distant data points are related.

The inventions of the Long Short-Term Memory (LSTM) model[8] and Gated Recurrent Unit (GRU)[5] break the short-term limitation of traditional RNN units and are able to remember the important information in previous distant steps. An input gate, a forget gate, and an output gate are designed in a LSTM cell. All information is controlled by these gates and added into the cell state with nonlinear transformations. Such design can *remember* the important information and *forget* the less important information regardless of distances of steps in the sequence. Compared with LSTM, GRU has similar effects but it is more concise and efficient as it has no output gate, only a reset gate and an update gate. In recent years, LSTM and GRU networks show outstanding results in time-series classifications and predictions.

In the astronomy field, research on transients and variables classification and prediction also takes advantage of them to recognize new objects automatically and efficiently.

2.4 Photometric Classification

In recent years, the classification and prediction of transients and variables mainly based on their light curves are one of the popular research topics, and RNN is widely applied in this field, such as SNe, variable stars, quasars and simulated data classifications.

In the transient research field, classified objects are often various types of supernova[3] or variable stars[16, 11, 1]. These may have obvious changes in short-term observations, which is helpful for classifiers to extract apparent characteristics from them easily.

There is also research[15] on simulated light curves, such as the PLAsTiCC dataset, which is designed for a Kaggle competition[12]. The PLAsTiCC data involve many simulated variables (e.g., stars and AGN) and transient (e.g., SNe) light curves. Since it is based on model implementations instead of the actual observations, there are still some differences between the simulated and the real conditions. Due to the more complex challenges of classifying quasars, there are less studies on the classification of quasars[21].

Since astronomical time-series data are often irregularly sampled, the design of classifiers may have some special operations to deal with this issue.

[3] aims to build a deep RNN classifier for supernovae. They considered the simulated supernovae light curves from the Supernovae Photometric Classification Challenge[13]. For data preprocessing, Charnock and Moss processed the uneven light curve data into adjustable unit data, and put the g, r, i, z filter values into a single vector, which ensures that there is at most one filter-type in each vector. For the network architecture, they used two LSTM layers as the hidden layers. They also tried bidirectional RNN which

is able to pass the data forward and backward. The results shows that the best performance is the two layers network with unidirectional LSTM units. This research provides a basic method of dealing with light curve data, which is referenced by many other papers in the transient classification field.

In [16], the objects are variable stars. An RNN encoder-decoder architecture is considered: After feeding the light curve to the encoder with the measurement values and the differences between sampling times, the encoder outputs a fixed-length feature vector. Then the decoder translates the vector representation into output time series. The autoencoding features show better performance than expert-selected features in some cases. The main function of this architecture is to convert the real unbalanced light curves to balanced adjusted time sequences, which can be used for further classification or regression. This method cleverly converts unbalanced data into balanced data that is easier for RNN to handle, and by constructing the sequences with delta magnitudes and delta time, the input of missing values is avoided.

[1] also aims at variable stars classification. In their approach, they constructed the input data with the magnitude differences and time differences. Additionally, they divided light curves into three parts with equal data points and feed them into the input layer separately. In this way, the non-uniform cadence in light curves is ignored. This method inspired us to deal with light curves of quasars since our data also have the yearly observation cadence.

[2] used simulated data PLAsTiCC to classify 15 kinds of transients, including AGN. Chaini and Kumar applied Gaussian Process Regression to generate synthetic light curves based on real data. The GP-fitted curves makes the training set more representative, and many features can be extracted from them. The unbalanced condition can also be tackled in this way.

Due to the more complex challenges of classifying quasars, there are less studies on the classification of quasars. There are two main reasons for this. The first is that quasars' optical variability shows a Damped Random Walk (DRW) feature, which indicates that the light curve could be random on short timescales, but over longer timescales (e.g. several years) then, a trend may emerge. The second is due to the observation issue. Many observations will be taken over a short period (e.g., a few nights), but then there is a large time separation (e.g. a year) until the next set of observations. This observation cadence brings difficulties to quasar classification without the colour and redshift metadata. In this case, the classification of quasars can be both challenging and interesting. In order to investigate how these affect the classification accuracy and how to reduce their negative influence, various methods are tried. The detail work is illustrated in Chapter 4 and 5.

3 Data Selection and Investigation

The first step for our project is to choose a proper dataset that contains quasars and non-quasars' data as our training and test sets. We focused on the datasets on Lasair and SDSS released data. Lasair is an event broker digesting ZTF transient data currently, and it doesn't ingest the long-baseline SDSS light curves. Currently, there are many versions of spectroscopic quasar catalogs (such as DR14Q[21], DR16Q[22]) from SDSS, which list objects that have very high confidence to be quasars. Additionally, the SDSS Stripe 82 region has been imaged by SDSS multiple times, therefore, many known quasars have SDSS light curves from 2000 to 2008.

Comparing the data from Lasair and SDSS, my tutors and I decided to utilize a SDSS Quasar targeted dataset. In order to understand the inner structure of the dataset, a detailed investigation of the data was carried out.

[21] https://www.sdss.org/dr14/algorithms/qso_catalog/
[22] https://www.sdss.org/dr16/algorithms/qso_catalog/

3.1 Dataset on Lasair

Lasair[23] is an event broker that will digest the data stream from LSST in the future, developed jointly by the University of Edinburgh and Queen's University, Belfast. It focuses on transient and variable sources: it will receive transient alerts from LSST once LSST is regularly observing the night sky, enrich the data with existing catalogs and analysis tools, and serve them to the LSST user community. As a test before connecting to LSST, it receives the ZTF alert streams.

Initially, the quasar light curves dataset in the Lasair database was preferred. Dr. Nicholas Ross created a Stripe82 AGN WatchList[24] including 24,789 quasar objects from the MilliQuas Catalog[4] which locates on the Stripe82 region to match the existing ZTF data on Lasair dataset. However, the match results show that there are only 506 objects found. This amount is not enough for building our classifier.

The main reason is that Lasair's goal is to digest transients alerts with obvious optical changes in a short time, such as SNe. However, quasars are variables whose luminosity is often variable over long timescales, and doesn't necessarily show a short-time feature. In this case, many quasars are not registered in the Lasair database. In addition, ZTF only have 2 year observation records, this is not good for quasars classification.

Although Lasair is a UoE event broker, and originally our preferred platform, it unfortunately did not prove to be useful for our investigations. This was a key 'null' finding, with the associated time cost

3.2 SDSS Stripe 82 Quasar Targeted Dataset

The SDSS Stripe 82 quasar targeted dataset is selected from the SDSS CasJobs interface[25] using SQL queries[26] by Dr. Nicholas Ross. The SQL query is presented in the Appendix 9.1. The light curve data is from SDSS Data Release 16 and cut into only Stripe82 region, which covers 300 deg^2 of the entire sky. The total object count is 43132.

This dataset has objects': position, magnitudes, magnitude errors in the u, g, r, i, z bands; obversvation MJD and redshift. The first step is to label each object with quasar or non-quasar. I first tried to regard objects with redshift value larger than 0.1 as quasars, and the rest are non-quasars. During early testing, we concluded that this label method is not accurate enough. Therefore, we decided to use existing catalogs to match objects in our dataset. First, the SDSS Quasar Catalog 14 data release (DR14Q)[17] and the Million Quasars Catalog (MILLIQUAS) Version 6.5 are used to find all recorded quasars in our dataset. Additionally, objects with redshift larger than 2.0 are also regarded as quasars. For non-quasars, the SDSS Stripe 82 Standard Star Catalog[10] published in 2007 is referenced to find all recorded stars in the dataset. Since quasars often have very large redshifts, objects with redshift smaller than 0.001 are also considered as non-quasars.

Thus, the number of marked objects is 33417, including 27413 quasars and 6004 non-quasars. Each labeled object is saved in a JSON file.

It is worth mentioning that during the cross-matching work, it was found that there are a few objects which are not listed in DR14Q[27] and coming DR16Q[28], however, they have a high possibility to be quasars. For example, object SDSS J000046.36+000827.5 (R.A. = 0.193201 digress; Decl=0.14098 degress) with a large redshift 2.086 is a quasar, it is not listed in the DR14Q Quasar catalog.

[23] https://lasair.roe.ac.uk
[24] https://lasair.roe.ac.uk/watchlist/155
[25] http://casjobs.sdss.org/CasJobs/
[26] https://www.sdss.org/dr16/spectro/targets/
[27] https://www.sdss.org/dr14/algorithms/qso_catalog/
[28] https://www.sdss.org/dr16/algorithms/qso_catalog/

3.3 Data Investigation

The investigation regarding the features and distributions of the SDSS dataset benefits the classifier design to a large extent. For each object, apart from its id, position and redshift, there are 5 bands u, g, r, i, z magnitudes and corresponding magnitude errors at a time. Objects with less than 10 observations are removed, and a processed csv file for all selected objects is generated.

3.3.1 Data Distribution

A heat map (Figure 1) is constructed to show the observation cadence and g band magnitudes of all objects for 8 years. The plot shows that observations are grouped together, with approximately a year between each group. Due to the upgrades of the SDSS telescope, in recent years, the time it takes to observe the target sky once is gradually reduced. Therefore, compared with early years, more observations of the same object can be made in a short time. The later the time, the denser the annual observations, which means that the last three sets of observations contain much more information than the previous five years. By estimation, the time gap between each set is about 260 days, and the average annual observation length is around 68 days. For RNN input sequences, in order to reduce the number of steps with null values and improve the efficiency of the classifier, it would be wise to skip those huge gaps and choose valid observations.

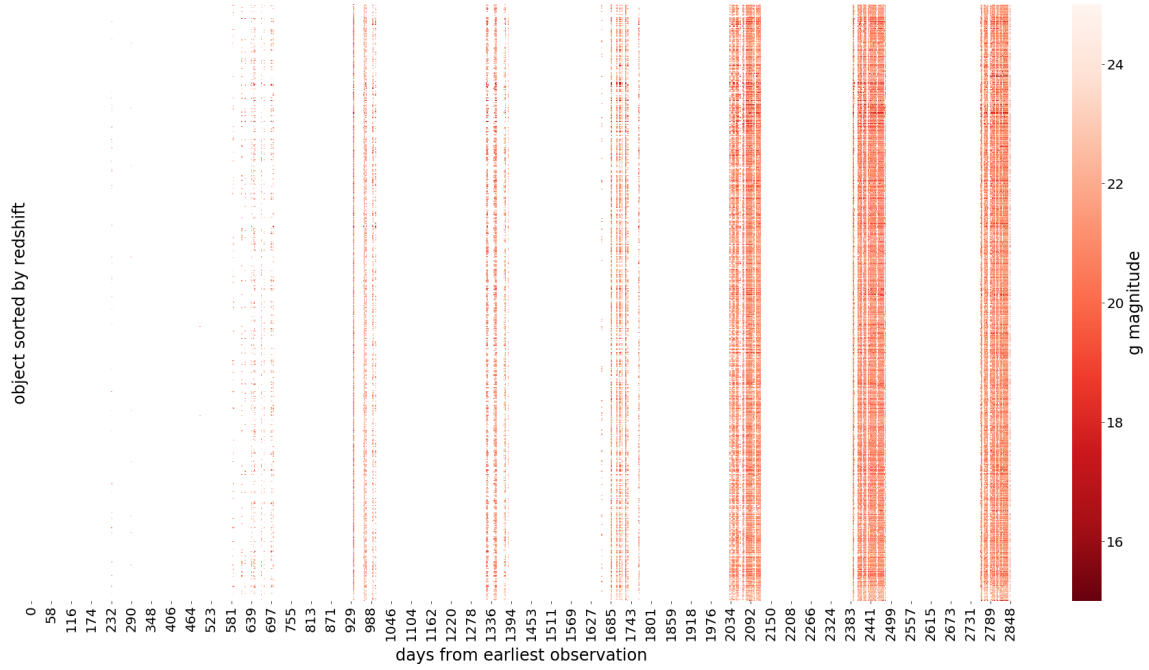
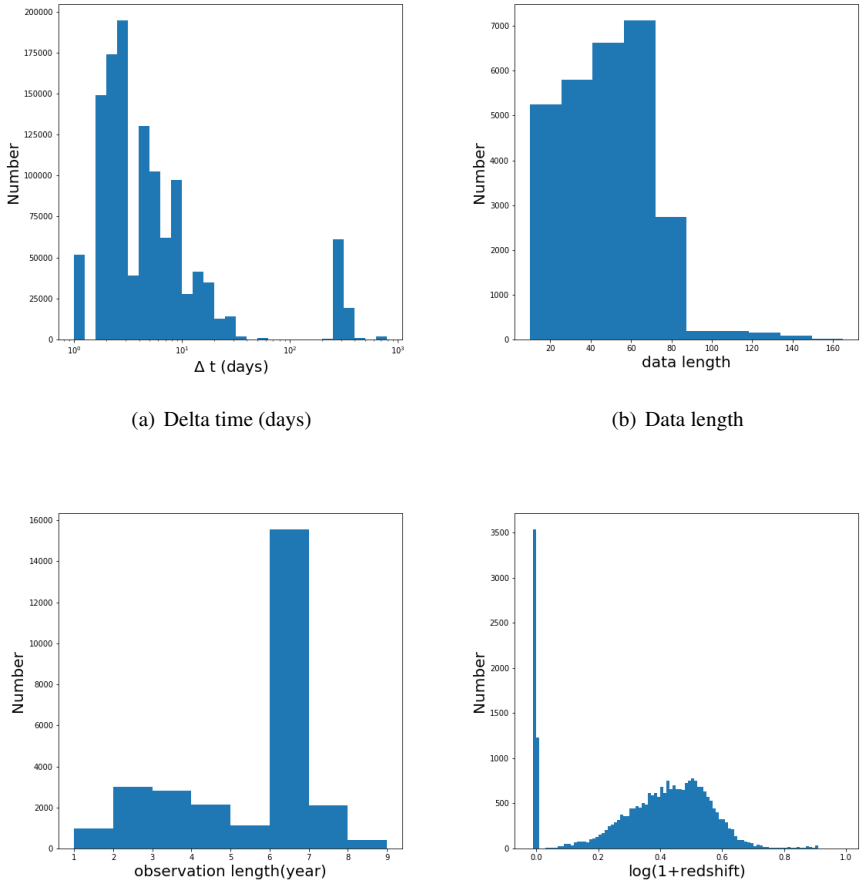


Figure 1: A heat map of the dataset represented by 4,000 objects, which is selected randomly from the entire dataset. It shows the g-band magnitudes during an 8-year observation period. Objects are ordered according to the redshift from small to large, And the redder the area in the colour bar, the brighter. For the x-axis, 1029 represents the latest observed MJD, and 0 represents the earliest observed MJD, which can be traced back to 8 years ago.

The four graphs in Figure 2 show the distributions of the dataset from the perspective of time and redshift. In Figure 2(a), high values are concentrated between 0 and 10 days, which indicates the small time differences in a round of observations. The delta time larger than 100 days means the time differences between two neighboring rounds of observations. Figure 2(b) depicts the distribution of observation times for each object. Most objects have 20-70 observations, and a few have over 100 observations. In



(a) Delta time (days)

(b) Data length

(c) Observation length (years)

(d) redshift

Figure 2: (a) is the distribution of time differences between neighboring MJDs for each object. (b) is the distribution of detection number for each object. (c) is the distribution of time differences between the earliest and the latest observations.(d) is the redshift distribution with the log values as the x-axis.

general, more observations an object has, the more complete its light curve is. As we know the mean observation times, the range of the length of the time sequence in the RNN design can be evaluated. Figure 2(c) shows the observation length with the unit of year. It is calculated by the difference between the earliest observation time and the latest one. Therefore, it happens that some objects don't have one or two years' data, though its overall observation time is over 8 years. In the redshift graph (Figure 2(d)), the non-quasars' redshifts are often less than 0.01, and the others are mostly quasars.

3.3.2 Data Correlation

In addition to data distribution investigation, analyzing the correlations between different data is also necessary, which helps us to have an insight into the data structure and inspires us how features are combined can achieve a classifier with high-confidence predictions.

Figure 3 and 4 are the Covariance Plot and Correlation Matrix of the SDSS Stripe 82 dataset respectively. The Covariance Plot shows that the error in each band increases as the magnitude increases. This means that the fainter the celestial body, the greater the observation error. In addition, magnitudes of different bands are all positively correlated. Especially for g, r, i bands shown in the Correlation Matrix, the

correlation value between any two of them is greater than or equal to 0.89. The correlation between redshift and bands are comparably weak.

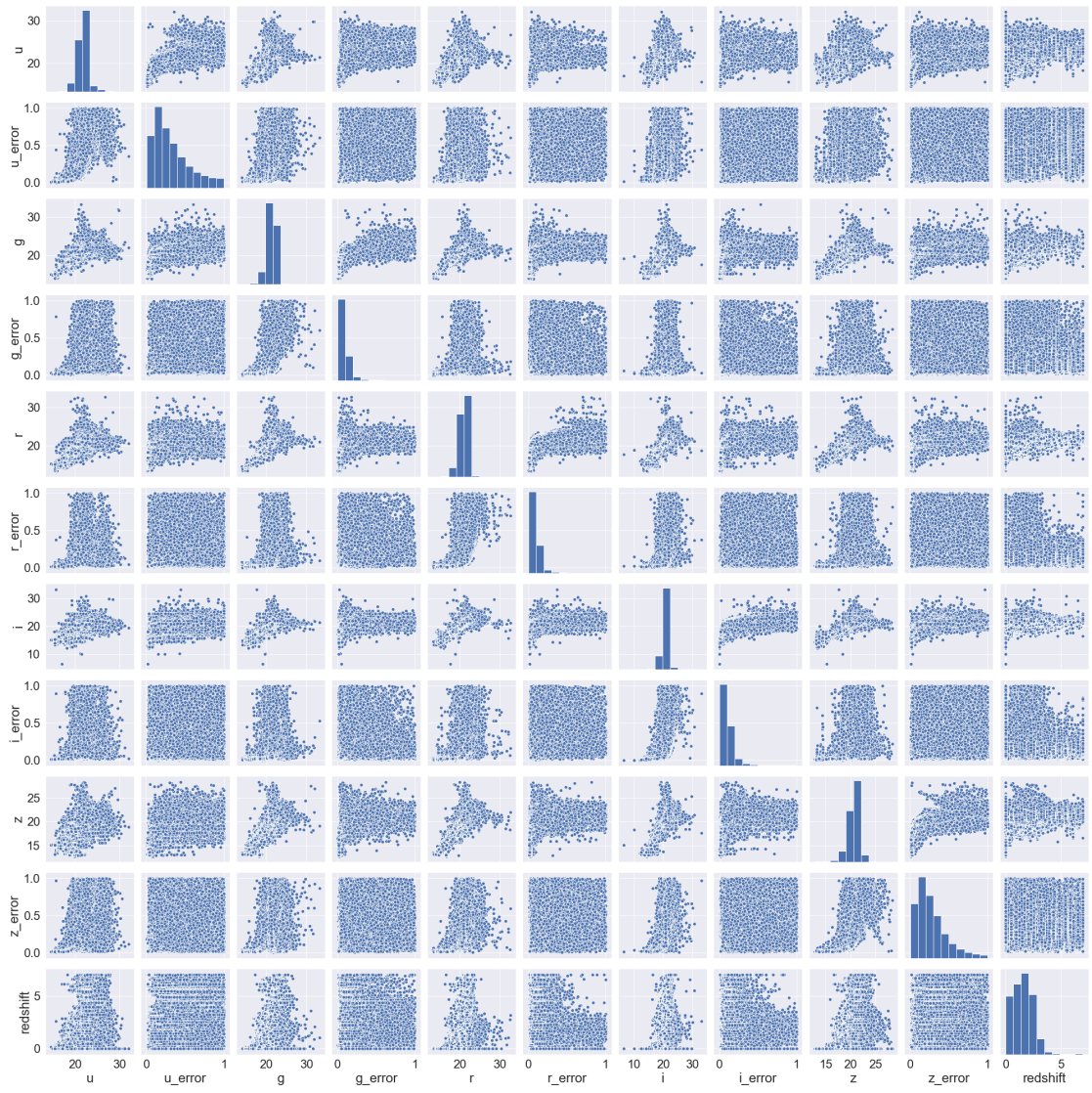


Figure 3: A Covariance Plot of the SDSS Strip 82 dataset. The compared columns are u, u_error, g, g_error, r, r_error, i, i_error, z, z_error and $redshift$.

3.3.3 Colour-Redshift and Colour-Colour Plots

Figure 5 depicts the colour distribution of all objects in the SDSS Stripe 82 dataset. $u - g$ colour is the difference of magnitudes in u and g bands. If $u - g$ is positive, it means the magnitude in g band is brighter than that in u band. If it is negative, the magnitude in u band is brighter. The graph shows that objects with a large redshift (>1.8) are greener, compared with those with a small redshift (<1.5).

In Figure 6, it reveals the relationship between redshift and colours that represent the difference between adjacent bands. It can be known that with the increase in redshift, the $u - g$, $g - r$ and $r - i$ rise slightly, which means that objects with big redshift are often greener and redder than those with a small redshift.

The colour distribution of all objects with quasar and non-quasar labels is shown in Figure 7. The large overlap of two types of objects means that many quasars and non-quasars have similar colours in u, g, r,

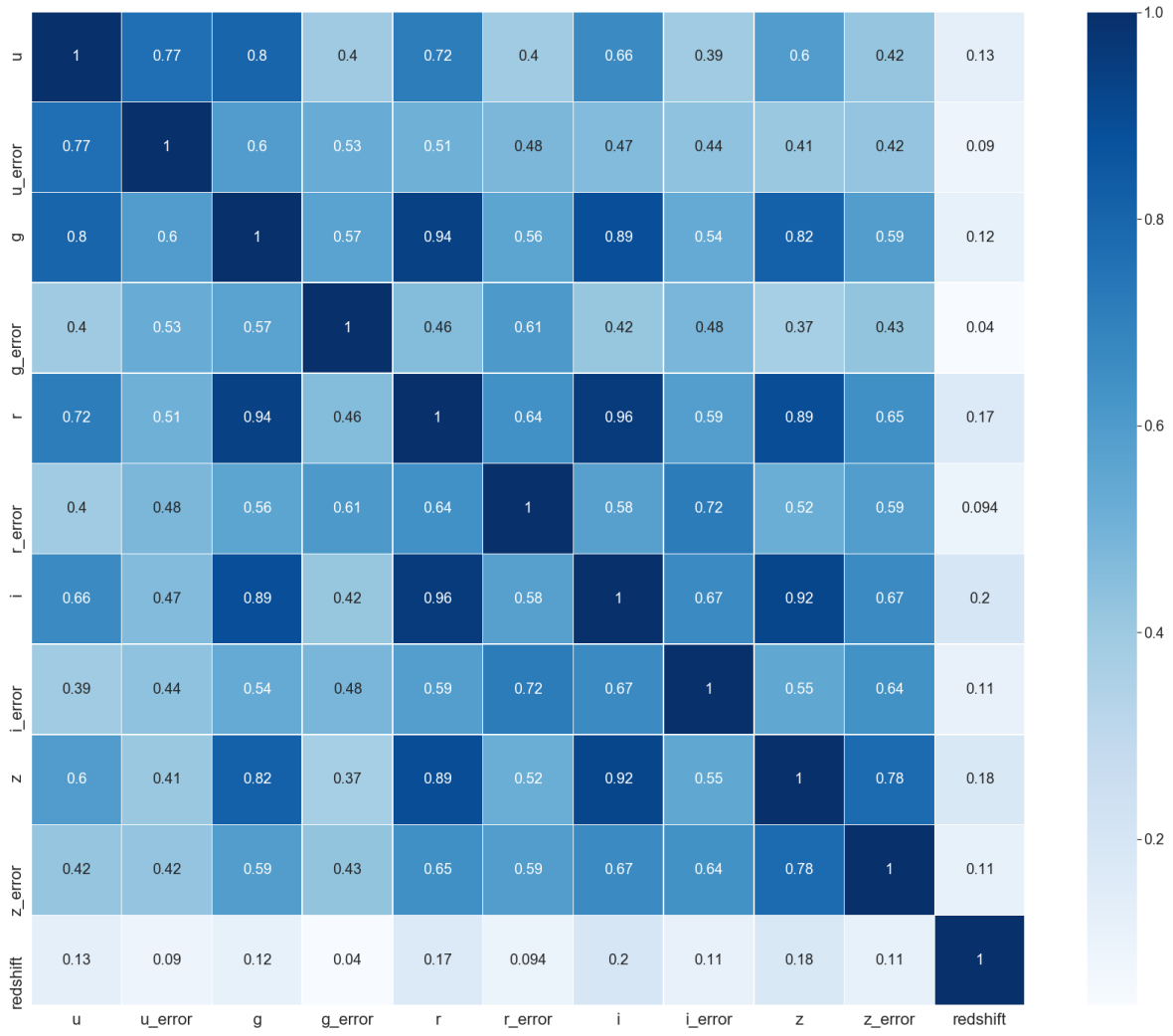


Figure 4: A Correlation Matrix of the SDSS Strip 82 dataset. The compared columns are u, u_error, g, g_error, r, r_error, i, i_error, z, z_error and $redshift$.

i and z bands. Therefore, classifying them with their imaging difference can be tough.

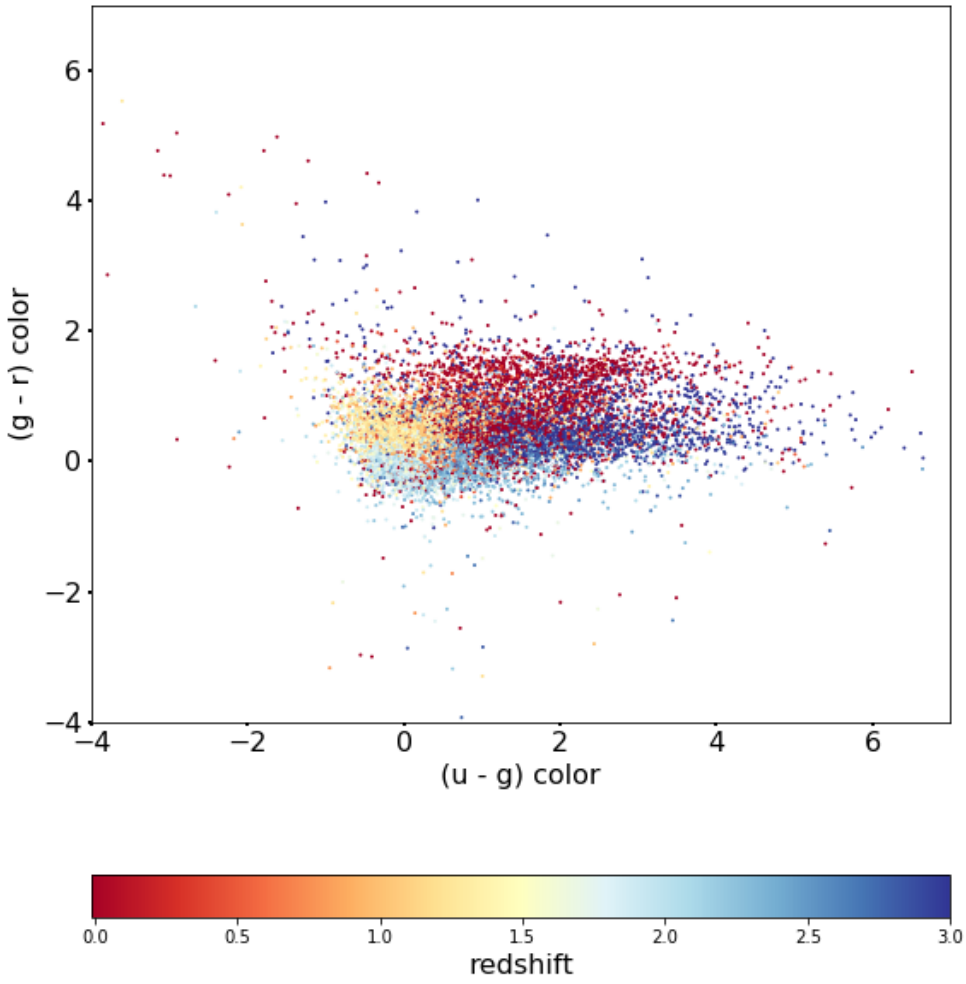


Figure 5: The colour distribution of objects in the dataset with redshift. Objects at the bottom left are bluer and hotter than objects at the top right.

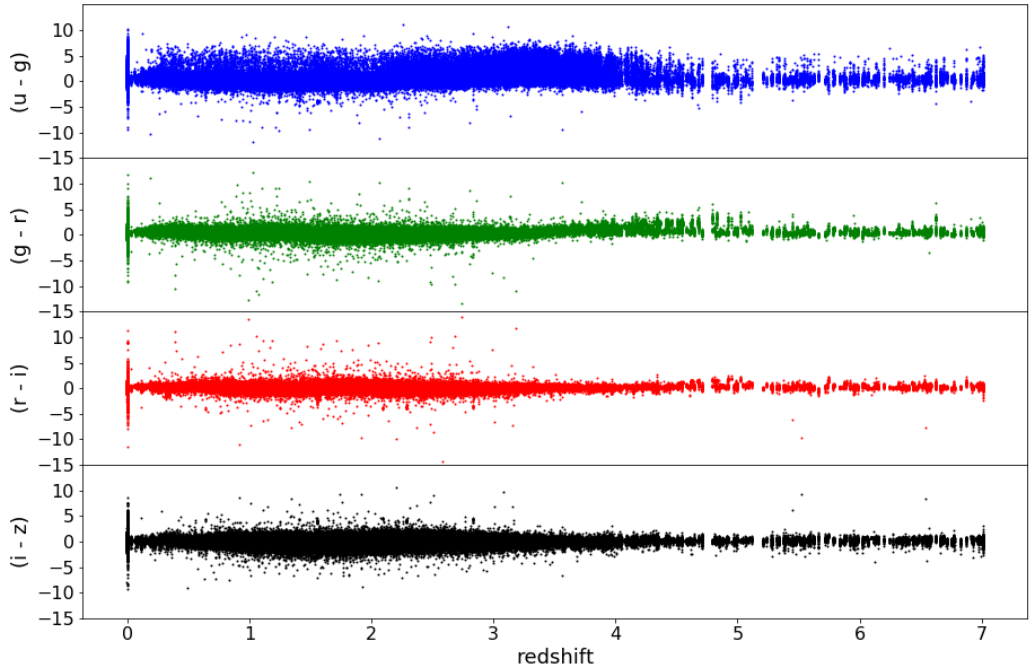


Figure 6: Object colour as a function of redshift. Here there are four colours: $(u-g)$, $(g-r)$, $(r-i)$ and $(i-z)$. In each case, the more positive the colour, the redder the object is. Note quasar colour, in all four panels changes with redshift. This is a additional complication for the training and test datasets. The stack of objects at redshift=0 are stars.

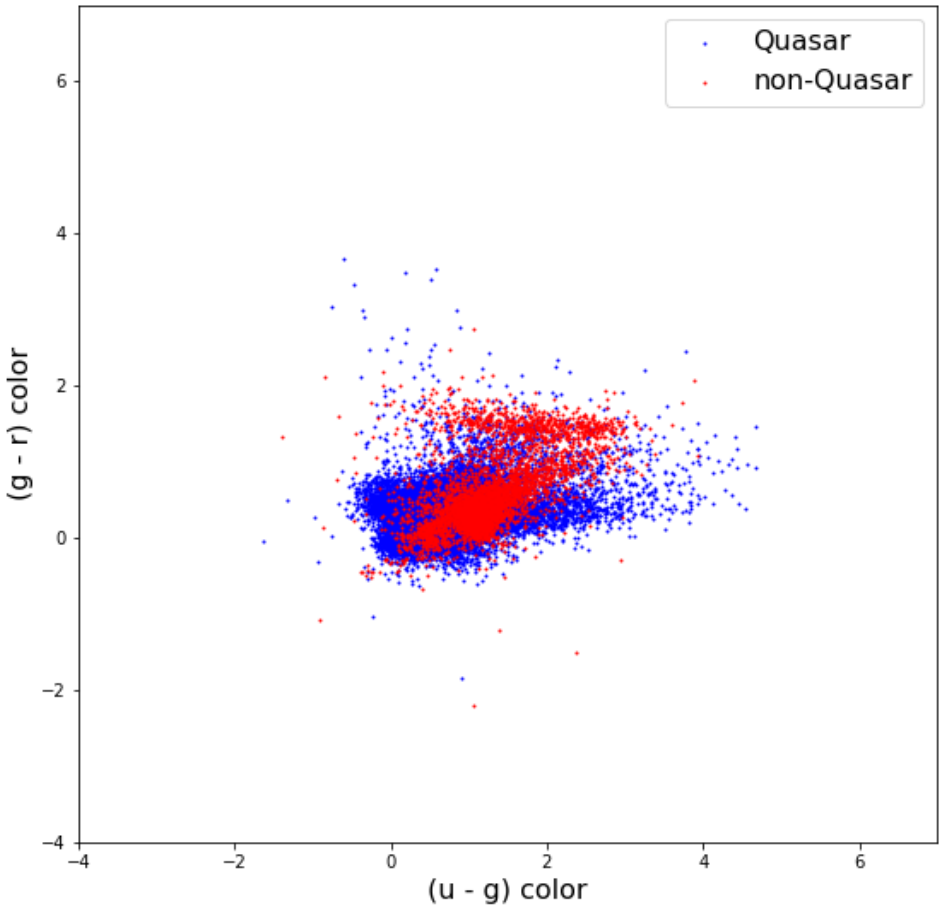


Figure 7: The colour distribution of labeled quasars and non-quasars in the dataset. Objects at the bottom left are bluer and hotter than objects at the top right. The number of quasars is much larger than that of non-quasars. It can be seen that the colour of non-quasars and quasars overlap greatly.

4 Research Methods

In this chapter, the key research methods of this project are illustrated. For data preprocessing, some common methods are applied, such as removing outliers, training and test set separation and data normalization. But there are also some novel methods: grouping observations by their cadence, using the Gaussian Process Regression algorithm to simulate light curves and replacing the real magnitudes with the delta magnitudes. In addition, three different input data formats are designed for the RNN input layer. The model architecture is explained in detail. Then a training plan is presented. For testing, two kinds of tests are planned to test the classifier's generalization ability and the prediction results with objects having different numbers of observations.

4.1 Data Preprocessing

To make our dataset usable for classification, there are several steps needed to be performed. First, all JSON files corresponding to each object need to be merged and transferred to a single file for input preparation. Then, data augmentation, grouping, and normalization to modify and reshape the input data.

4.1.1 Remove Outliers

The removing noise procedure is made during the generation of the preprocessed files. The *preprocess.remove_noise function* of an astronomy package - *feets*[29] is called. The magnitude values greater than 5 standard deviations from the mean of the object's magnitudes are considered as noise and will be removed.

This method is often used in stars' light curves, but may not be suitable to quasars. Due to the quasars' highly unpredictable luminosity, some true and extreme magnitudes may be regarded as noise and be eliminated, thus destroying the original shape of the light curve.

After consideration, this function is not used for the final preprocessed file.

4.1.2 Training Set and Test Set

The preprocessed file is divided into a training set and a test set. For the training set, it contains 22783 objects (20399 quasars and 2384 non-quasars). The unbalanced classes condition can be improved by the design of loss function in the RNN architecture, though it still has negative impact to the performance of the classifier.

The test set has 5396 objects, including 2402 non-quasar and 2994 quasar light curves. In order to evaluate the generalization capability of our classifier, the magnitude in each band in the test set is added with noise whose range is referenced by its magnitude error value. The detailed illustration is in Chapter 4.5.

4.1.3 Group Observations by Year

The light curve can be regard as time-series sequence. All objects' sequence will be padded equally based on the real MJD and fed into the input layer of RNN. There are two disadvantages: The first is that the sequence length can be extremely long, up to 3352 data points. The computation of RNN is very time-consuming. The second is that large gaps between each round of observations mean a considerable number of data points are set as zero, which will reduce the quality of the input data and training.

Inspired by the observation cadence shown in Figure 1, I tried to divide each light curve into multiple groups based on its MJDs. Each round of observations is set as a group. Each group size is adjusted to

[29] https://pypi.org/project/feets/

be the same and the number of groups is fixed to ensure that every object has the same number of groups and group size, which satisfies the RNN input requirements. In this way, the time gap between each year can be ignored. This method is applied into both the training set and test set. The detailed input formats are illustrated in Chapter 4.2. There are several steps for grouping observations.

Step 1 Remove isolated data points

In an object's light curve, the number of observations is not always evenly distributed across time. A small number of observations are clustered together, but they are far from other observations. One reason may be that after the telescope has detected an object several times, it was planned to observe other areas of the sky or focus on other objects. Another reason is that in the early years, the imaging speed was low, so one object can only be observed a few times a year. After photographic technology is upgraded, the telescope can image objects rapidly and efficiently. Therefore, there were more observations produced in later years.

For example, in Figure 1, it can be discovered that for objects with 8-year records, the number of observations in the first year (first column in the graph) is much less than that in the second year. In this case, if the former forms a group, the information contained in it is much less than the latter group. In order to keep the equal group size, most of data points in the former group will be set as zero. This doesn't help the neural network to learn the characteristics from their sequences. Therefore, the small observation clusters should be removed. In this function, the *remove check delta* (the maximal tolerated time difference between neighboring MJDs) and the *min size* (minimal number of observations in a group) are user-defined parameters.

We found that the most suitable *remove check delta* and *min size* are 233 and 3 respectively. Algorithm 2 in the Appendix shows the ideas of achieving this function.

Step 2 Merge multiple observations per night

SDSS may scan the same region multiple times in one night and observe the same source several times. For the convenience of input generation, the unit of the timestep in the sequence is set as one day. Therefore, it is necessary to merge observations made during the same night. In this function, the *check delta time* is a user-defined parameters, and set as 0.7. When two observations whose MJD's difference is less than 0.7 day, the early observation will be saved and the other will be removed.

Step 3 Find all groups in each object

For each object, all observations need to be divided into several groups by comparing each time difference between its adjacent observations with the *remove check delta* (233 days).

During this process, grouping condition contain a lot of information: the maximal or minimal group count, the maximal, minimal and mean group size. This gives us convenience to understand the dataset's structure better. Table 16 in the Appendix displays all relevant information.

Step 4 Calculate the suitable time difference between neighboring groups

After all groups are found for each object, it is necessary to find the common time gap that can properly separate every object's light curve into groups. This means that the suitable gap time should be short enough to consider all group gap times, but also long enough to separate all groups.

First, a variable *suit delta* is defined with the initial value equal to the maximal time differences among all objects' groups. Then, for each object with more than one group, the *suit delta* is reduced gradually until it can divide each object's light curve into the correct groups. After calculation, the suitable delta time for our dataset is 233 days. It is worth noting that *remove check delta* in Step 1 refers the *suit delta*. A parameter adjustment process is required.

The specific implementation can be seen in Algorithm 1. Figure 8 shows how light curves are divided into groups.

Algorithm 1 Find Suitable Gap Length

```
n = 0
suit_delta = max_delta
while n < len(id_list) do
    group = num_group[n]
    obj_delta_time = delta_time[n]
    if group > 1 then
        while len([x for x in obj_delta_time if x>=suit_delta) < group -1 and suit_delta > 0 do
            suit_delta -= 1
        end while
    end if
    n += 1
end while
return suit_delta
```

Step 5 Calculate the boundary MJDs of each group

This step is to calculate the boundary MJDs for each group according to the real MJD and *suit delta*. All light curves are ordered by the real time. Backtracking from the most recent MJD, by calculating the earliest and latest records of all observed objects each year, the boundary MJDs of all groups can be found. Sequentially, the right boundary of each group is set as the benchmark, and the modified left boundary MJD is calculated by reducing the user-defined sequence length from the benchmark. As a result, the boundary MJDs with the same length for each group can be obtained. This is called *model group bound*. Algorithm 3 in the Appendix explains this function's implementation.

Step 6 Reconstruct groups with identical lengths

As the *model group bound* is obtained, the next step is to let all groups of observations fit the model. It is implemented in different input format structures (see chapter 4.2) in different ways. The key is that for a group bound in the model and for one object, the object's actual observation date should correspond to the date in the model bound. When there are some dates in the model that cannot be found in the real observation records, the values on them should be stored as zero. In this way, it can be assured that each group in each object has the same group size.

The *model group bound* for this dataset is: [[52519, 52586], [52519, 52586], [52904, 52971], [53285, 53352], [53638, 53705], [54001, 54068], [54366, 54433]]. The numbers in each list are the boundary MJDs for each group.

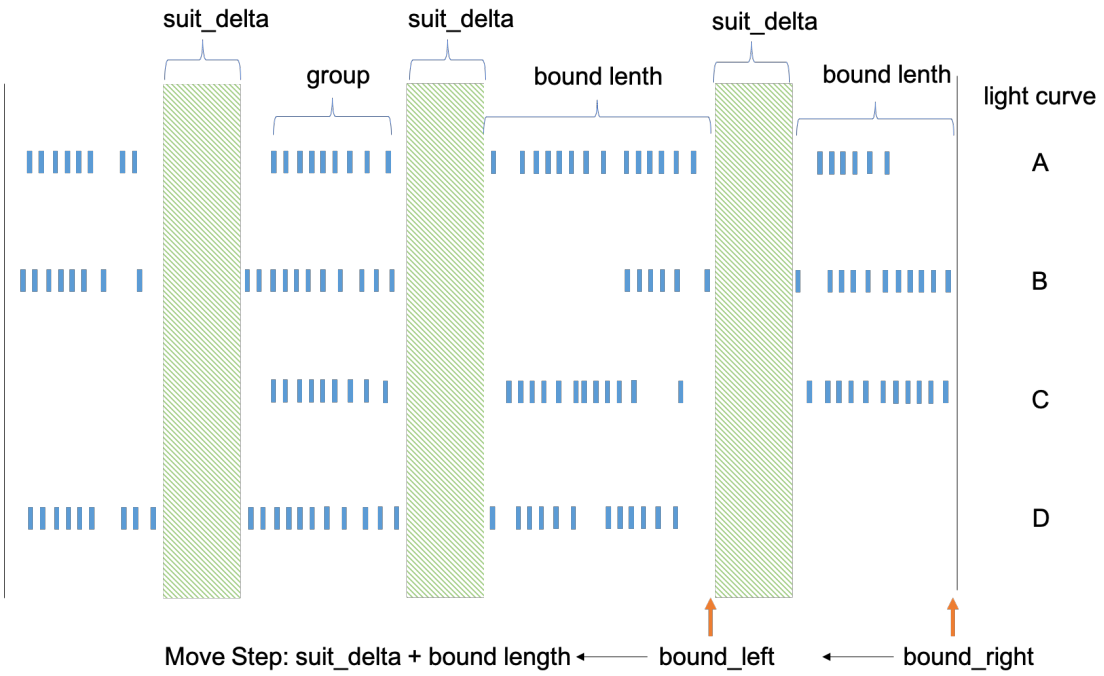


Figure 8: How the light curves are divided into groups. Here 4 examples of light curves-A, B, C and D are shown. Each small blue vertical line represents an observation. The *suit delta* is identical in each group gap, and *bound right* and *bound left* move from right to left with the step of *suit delta +bound length*. This is for determining the final boundary MJDs of each group for all objects.

4.1.4 Gaussian Process Regression

For this research, Gaussian Process Regression (GPR) is investigated as a method to achieve data augmentation for the light curve data. The goal is to predict the missing magnitudes of an object in each group. In this way, each MJD in a group corresponds to a magnitude, thus enriching the information contained in a group. Figure 9 shows the light curve in a group after regression.

GPR is a non-parametric model based on Bayesian approach. It can fit a function to the data (called regression) and make predictions. Multivariate Gaussian distribution is the basis for GPR. It is defined by the mean vector μ and the covariance matrix Σ. μ shows the expected value of the distribution, and Σ models the variance of each dimension and determines the correlation between different random variables[30]. The correlation depends on the distance between random variables. Different kernels can be used to generate the covariance matrix Σ.

Given a dataset with random variables X and Y, GPR firstly assumes that both variables obey the joint probability distribution $P_{x,Y}$ and then applies Bayes inference method to update current assumptions. Sequentially, the prior distribution f$\sim(\mu, \Sigma)$ is gained. This can be used for prediction. When a new random variable x* is given, the posterior distribution $P_{y/x*}$ can be generated by the application of Bayes Theorem. This algorithm is often used in time-series forecasting.

The package *sklearn.gaussian_process* is applied to achieve this function. We chose the Radial Basis Function (RBF), shown in Formula (1), which calculates the squared Euclidean distance between two vectors. X and X' are two vectors, and σ is a parameter that controls the radial range of the function. For its parameters, length_scale is 10, and length_scale_bounds is [1e-1, 1e1)].

$$K(x, x') = exp\left(-\frac{\|x - x'\|}{2\sigma^2}\right), \ \sigma > 0 \tag{1}$$

[30]https://distill.pub/2019/visual-exploration-gaussian-processes/

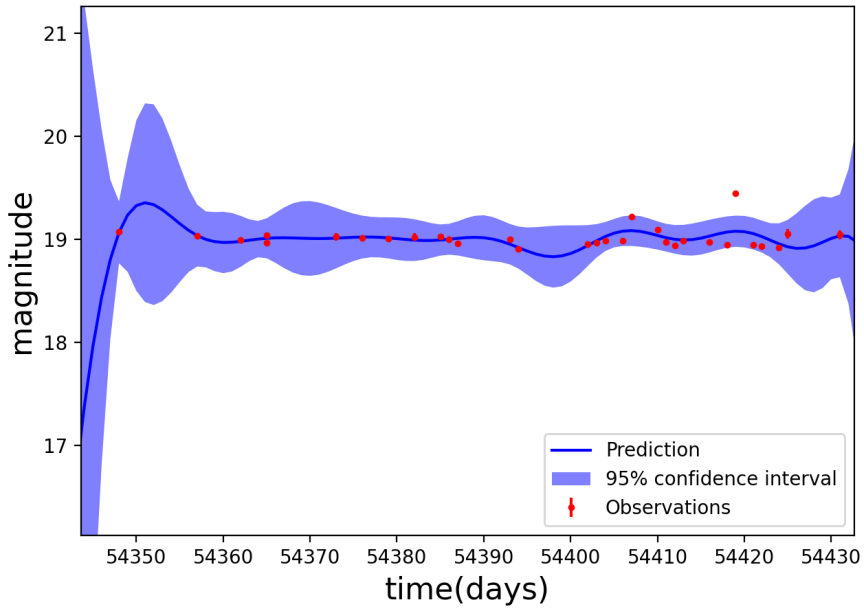


Figure 9: The real and simulated light curves in one group, the blue prediction line is the simulated light curve generated by GPR, the red points are original observation values. The magnitude error for each magnitude is involved in the calculation.

Two methods are designed for data augmentation. One is to only add predicted values to vacant positions and leave other data points as usual, shown in Figure 10-A, the other is to use GPR to generate a new light curve based on the old light curve and replace all values with predicted values, shown in Figure 10-B.

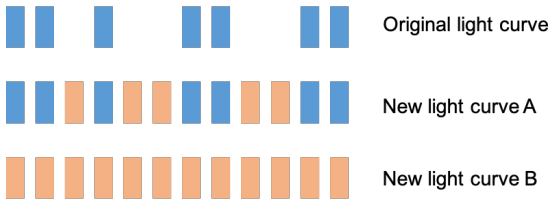


Figure 10: Depiction of two methods of generating a new light curve for a group. Each blue vertical line represents an observation's value per day. The orange vertical lines are predicted values.

4.1.5 Difference between Neighboring Observations

This method aims to classify objects based on light curve trends, rather than measured values. The magnitude difference between neighboring observations (Formula 2) is used instead of measured values. Experiments have proved that this method performs poorly due to unbalanced observation dates. All null values are set to 0, which will affect the classifier's learning light curve trend. It was planned to use GPR to call this method, but as shown in Chapter 5.2.5, GPR is not suitable for group design, it still cannot improve the performance of the classifier. If the time difference is also considered, the performance is expected to be better. Due to lack of research time, this method is planned to be implemented in the

future.

$$x^* = x_{t+1} - x_t,\ t \in a\ light\ curve's\ MJDs \tag{2}$$

4.1.6 Data Normalization

Data normalization is most often considered as the linear transformation of data. It does not change the numerical order of the original data, and can speed up the convergence speed of the loss function during backpropagation. In addition, it can eliminate the differences in the influence of features with different dimensions on neuron parameters during training. For this project, two methods are applied.

1 Scaling (min-max normalization)

Scaling compresses the data to values between 0 and 1, using equation 3. It is only related to the maximum and minimum in the dataset, therefore, it is often used for stable data without extreme high or low values. When new data are added, all values need to be updated.

$$x^* = \frac{x - min(x)}{max(x) - min(x)} \tag{3}$$

2 Standardization (Z-score normalization)

Standardization (as shown in equation 4)) calculates the mean and standard deviation of the data, and converts the data to a standard normal distribution. The value range of the data is unlimited.

$$x^* = \frac{x - \mu}{\sigma} \tag{4}$$

4.2 Input Data Format

For the RNN, the input data should be a three-dimensional array - {*batch size, sequence, vector*}. All sequences should have identical lengths. For data points in a sequence, they are vectors with specific dimensions sorted in a certain order. The construction of sequences determines the performance of the training process as well as the test results. In this work, three formats are designed specifically for given light curves: the *Simple* format, the *Group* format and the *Season* format.

4.2.1 Simple Format

In the *Simple* format, a sequence is the selected multi-band light curves for one object. The vector in the sequence is the combination of different bands' magnitudes. Figure 11 shows the sequence's structure. In this case, the time differences between observations are ignored. The order of the sequence is simply the order of MJDs from early to late. Since the sequence length for each object is different, the longest sequence length is regarded as the standard length for every object. The sequence that is shorter than it will be padded with zero and placed in the front of its original sequence.

4.2.2 Group Format

According to current literature, the light curve of quasars has an arbitrary trend in the short term, but has a more gradual trend over long time frames, while the luminosity of stars should be relatively stable for years.

Inspired by this information, the group format, as shown in Figure 12, is designed to put the information of each group in a vector in MJD order, and let the classifier learn the trend of the value at each position

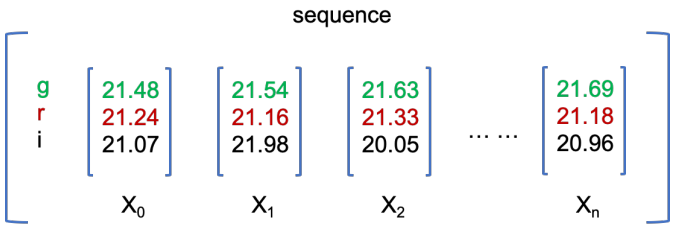


Figure 11: An example of the *Simple* input format. Take three bands' combination (g, r, i bands) as an example, for each vector, these three bands' magnitudes are set in a specific order. For convenience, the values in the figure are not normalized.

of each group in multiple consecutive groups. It is hoped that we can use this to explore whether the quasar data is periodic or there is an overall change in luminosity every year.

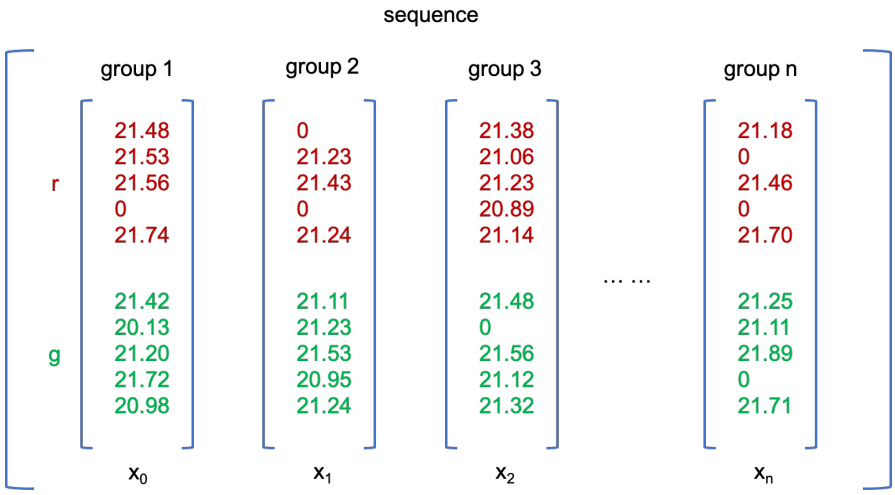


Figure 12: An example of the *Group* input format. Take two bands' combination as an example, the group size is assumed to be 5 days. Each vector contains a group information for each band. For convenience, the values in the figure are not normalized.

4.2.3 Season Format

The *Season* format, as shown in Figure 13, regards each group of all objects as a sequence. It breaks objects' light curves into sequences whose lengths are equal to the group size. It is designed for testing whether the classifier can extract enough characters from a group of information to recognize quasars. The advantage of this method is that the sequence can be selected. The groups with less than 10 measured observations are supposed to be removed since they have too many missing values. This method can speed up the training process and improve the quality of input data.

4.3 Model Design

The classifier model architecture for this project is determined by the user. A configuration file is provided for users to modify many parameters. Therefore, the RNN architecture is not fixed but still has some basic blocks. Different parameter combinations are tested for comparison and seeking for the suitable settings for the dataset.

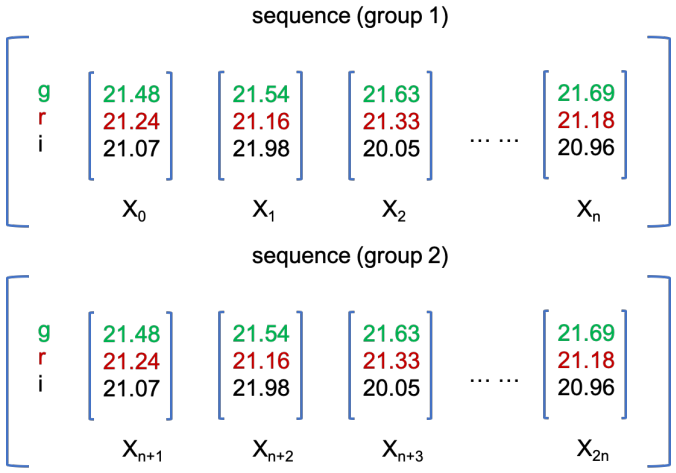


Figure 13: An example of the *Season* input format. Take three bands' combination as an example, the group size is assume to be n. For each vector, it contains three bands' magnitudes. The sequence length is equal to the group size. In this case, one object's light curve is divided into several sequences (groups).

4.3.1 User-defined configuration

There are many user-defined parameters in the configuration, as shown in Table 4. The advantage of this method is that users can adjust the construction, training and test of the classifier easily according to the volume and structure of the dataset. In order to prevent confusion of the results generated by different parameter combinations, a deep file tree is generated when the program is running to save the results in the corresponding location.

4.3.2 RNN Architecture

A deep Recurrent Neural Network is trained to map each light curve into the predicted possibilities of being a quasar and a non-quasar. Equation 5 describes this classification function that maps a sequence length τ of a variable v with an input matrix M^{vt} at each time step t to the output probability 2-D vector y^{v}. θ means all weight and bias parameters involved in the neural network. The class of the variable will be predicted as the one with a probability greater than 0.5.

$$y^{v} = f(M^{vt}; \theta),\ t \in \tau \tag{5}$$

The implementation of the RNN architecture draws on Charnock's research[3] and is simplified based on the project demand. Different types of RNN neurons - LSTM, GRU and SimpleRNN are tried test their abilities in quasar recognition. A powerful Python API *Keras* built on *Tensorflow* is used to achieve the building and training of the classifier. Figure 14 shows a basic classifier model with two LSTM layers. The explanations for this model are below.

1 Input layer

At a time step X, the input is an $n \times d$ matrix M , where n is the batch size, d is the dimensions for a vector in the sequence. The neuron number of the input layer is equal to d. In Figure 14, the vector has three dimensions - g, r and i bands, which will be fed into D1, D2, and D3 respectively. Figure 15 depicts the method of feeding input data. At each time step X_t, for each batch with batch size n, the vectors at the same time step of n sequences are formed as matrix M and fed into the input layer. The output shape of the input layer is (n, τ, d), where τ is the sequence length.

2 Masking

Table 4: Configuration Setting

Type	Parameter	Definition	Example
input	train_path	the path of the training set file	train_set.csv
	test_path	the path of the test set file	test_set.csv
	save_path	the folder that saves all results	results
	seed	the seed for generating random number	1
	features	the bands/features used in training. All features: u, g, r, i, z	g,r,i
	format	the input format for the training. Three formats are provided: simple, group, season	group
	processed	the preprocess method for the input data. Three methods are provided: s: standardization; n: normalization; d: difference between neighboring data points	s
	set_GPR	whether to use the Gaussian Process Regression. This method will generate a new regressed light curve for each group of light curve of an object	True
	group_size	the number of days in all groups	67
	group_num	the number of groups for each object	7
	cut_fraction	For prediction, the fraction of data drop from each group. This parameter is used for testing the improvement of accuracy with more complete data	0.1
network	rnn_type	the type of RNN layers. Three types are provided: LSTM, GRU, SimpleRNN	LSTM
	hidden_layers	a list of hiddren layers' neuron numbers	[64,64,64]
	dropout	the fraction of objects dropped before being fed into the next layer to avoid overfitting	0.35
	plot_model	plot the model architecture	True
training	batch_size	the number of sequences fed into the layer for each time	512
	num_epochs	the number of cycles through the full training set	10
	test_fraction	the fraction of validation data in training set	0.2
	optimizer	the optimization method for the loss function. Two functions are provided: Adam, SGD	Adam
	learning_rate	a tuning parameter in an optimization algorithm that determines the step size at each iteration while moving toward a minimum of a loss function	0.0001
	decay	if the learning rate will decrease with the increasing number of epochs. If true, decay_value = learning_rate/num_epochs	True
	metrics	the metrics during training for testing the performance of the classifier	accuracy,AUC

Considering that there are many padded zero values in the input data which will negatively impact the training effect, the *keras.layers.masking* is called to mask them in subsequent calculations.

3 Hidden layer

In this work, the type of the hidden layer could be LSTM, GRU and Simple Unit. Compared to other types of Neural Networks, the key difference for RNN is that its algorithm expands in the time dimension. The output value of each neuron (also called unit) at time step t is influenced by the value at time step $t-1$. Equation 6 (taken from [6], chapter 10, equation 10.5) presents the state s of a typical RNN unit h, where x is the input vector from the input layer, θ means all weights and bias parameters.

$$s^{(t)} = f(s^{(t-1)}, x^{(t)}; \theta) \tag{6}$$

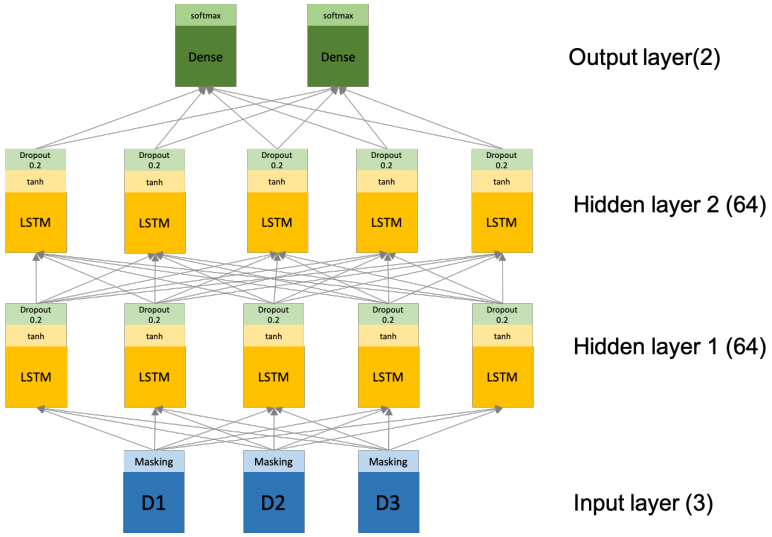


Figure 14: This is a RNN model with specific configuration. This model is designed for the three-bands data with *simple* format. After masking, the data are sent to the first hidden layer. There are two LSTM layers with 64 neurons and a dropout 0.2 for each. The output layer has two neurons with the softmax activation, and it produces the prediction probability for each object.

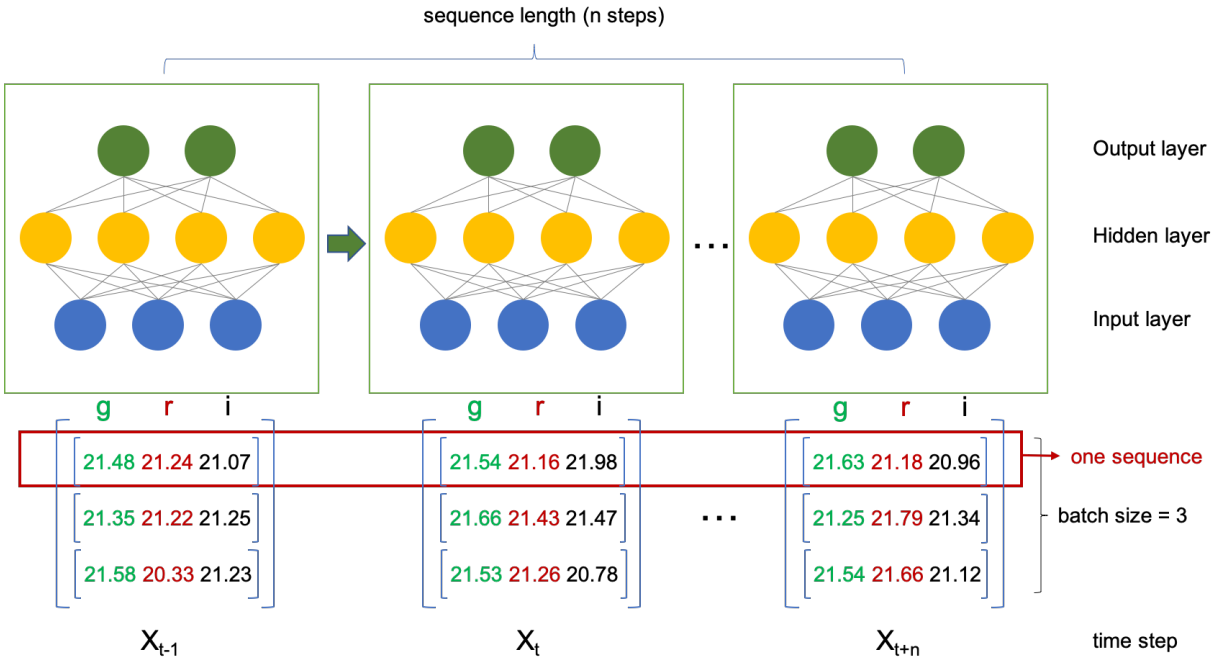


Figure 15: This figure shows the way of feeding input data in an RNN classifier. It is assumed that the vector for each step has three bands' values. The sequence length is n and the batch size is 3. For each time step X_t, a 3-d input tensor is fed into the input layer, which contains three sequences' corresponding data points at the same time step. The value of each RNN neuron in the hidden layer at this time step t does not only depend on X_t, but also depend on its value at time step X_{t-1}.

Simple RNN units can only learn the short-term trend of sequences. In many cases such as quasars' light curve sequences, the magnitudes in a short time could be random but there might be a clear trend in a long time. Therefore, Long Short-Term Memory (LSTM) unit and Gated Recurrent Unit (GRU)

are also applied in the model.

For the results of the classifiers with different layers, LSTM and GRU perform much better than the leaky SimpleRNN units.

The activation function for hidden layers is *tanh* (Formula 7), which compresses the output of each neuron into the value between 0 and 1. It is able to transfer the linear function to non-linear function to enhance the expressive ability of the model.

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{7}$$

The initial tests show that for the training set, the loss function is small and the accuracy is high, however, for validation test, the loss is considerable and the accuracy is relatively low. This means that the classifier has an overfitting issue. In order to alleviate the degree of overfitting in training, dropout is applied after each hidden layer. It will ignore a certain proportion of the output of previous neurons and enhance the generalization capability of the classifier.

4 Output layer

The output layer is a full-connected layer with 2 neurons and it is the final layer in the neural network. It receives all outputs from the neurons in the previous hidden layer and produces a 2-D vector. *Softmax* activation function (Formula 8) is applied to squash the values in the 2-D vector into two values between 0 and 1 and the sum of them is 1. These two values are the predicted possibilities of the two classes for the input object.

$$softmax(x_i) = \frac{e^{x_i}}{\sum_j e^{x_i}} \tag{8}$$

5 Loss function:

The loss function is used to estimate the difference between the predicted value $f(x)$ produced by the classifier and the true value Y. The smaller the loss function, the better the classification. The loss function is binary cross-entropy, which is specially designed for binary classification training. Additionally, since the classes of objects in the training set are extremely unbalanced (quasars: non-quasars≈17:2), during backpropagation, the loss function will be biased towards the side with more samples, resulting in a small loss function at the expense of low accuracy for the small class. Therefore, the loss function for the small class should be given high weight, and the big class low weight. It is called weighted binary cross-entropy. In Formula 9, M is the number of samples with the quasar class, m is that with the non-quasar class. y is the true class for a given object X.

$$L(X, y) = -\frac{m}{M+m} y \cdot log(p(Y = 1|X)) - \frac{M}{M+m}(1-y) \cdot log(p(Y = 0|X)) \tag{9}$$

6 Optimizer

The optimizer is a function to update the weights and biases of each neuron in the network for minimizing the loss function during backpropagation. The key parameter of the optimizer is the learning rate α, which determines the convergence speed of the loss function. A large learning rate might always skip the local optima and the loss cannot converge, and too small learning rate will slow the convergence speed and will need a longer time for training.

In this work, Adaptive Moment Estimation (Adam) is the optimizer. The exponential rates for the first moment estimates and second moment estimates remain the default values in *Keras* setting. Different learning rates are tested for results comparisons. In addition, the learning rate decay is also used in Adam, which decays the learning rate with the increasing epochs.

7 Metrics

During training, the metrics for each batch and epoch can be shown. There are three metrics for the classifier: loss, accuracy and AUC (Area Under Curve). After each batch, these metrics are calculated for the training set, and after each epoch, they are also applied to the validation set. After training, the validation set's ROC graph, Confusion Matrix, Recall, Specificity, Precision, F1_score, TN, FP, FN, TP will be produced. Table 5 shows the definitions of all metrics.

Table 5: Metric Definition

name	definition
TN	True Negative. Predicted class as non-quasar and the true class is non-quasar
FP	False Positive. Predicted class as quasar and the true class is non-quasar
FN	False Negative. Predicted class as non-quasar and the true class is quasar
TP	True Positive. Predicted class as quasar and the true class is quasar
accuracy	$\frac{TP+TN}{TN+FP+FN+TP}$
recall	Also called sensitivity, true positive rate. $\frac{TP}{TP+FN}$
specificity	Also called true negative rate. $\frac{TN}{FP+TN}$
precision	Positive predictive value. $\frac{TP}{TP+FP}$
f1_score	$\frac{2\times precision\times recall}{precision+recall}$
loss	The output of loss function
ROC	Receiver Characteristic Operator. In this graph, the x-axis is $(1-specificity)$, and the y-axis is $sensitivity$. When the curve is close to the upper left corner, it means the classifier has high sensitivity and low false positive rate.
AUC	Area under the ROC curve.
confusion matrix	A table that involves TN, FP, FN, TP values.

8 Callbacks

The *Keras.Callbacks* package is used for tracing the metrics during training. *Callbacks.Callback* function is called to record all metrics values in training, and *Callbacks.EarlyStopping* will stop training when the validation loss no longer decreases in three epochs.

4.4 Training Plan

Since there are many user-defined parameters for constructing and training the classifier, there are several potential tests designed to discover the performance with different conditions. All tests are conducted on Cirrus, which is a state-of-the-art supercomputing HPC system[31]. One GPU node is used for each training process.

Table 6 shows all combinations of input formats and RNN types. All tests are based on these combinations, and all metrics are recorded for each test for further analysis. The tests can be divided into two parts: network and training parameters comparison as well as bands and RNN types comparison.

For network and training parameters comparison, shown in Table 7. The tests are divided into 4 parts:

(1) Number of layers and neurons

(2) Dropout value

(3) Learning rate

(4) Batch size

[31] https://www.cirrus.ac.uk/

Discovering the impact of different network parameters on the training performance can help in finding the optimal classifiers in different cases.

In bands and RNN types comparison, shown in Table 8, the identical training parameters and the basic network architecture are applied to all tests. We look forward to knowing the impact of different bands, different input methods and RNN types on the classification results.

Table 6: Configuration Setting

RNN type	format	GPR	preprocess	codename
LSTM	group	False	standardization	LSTM-gfs
	season	False	standardization	LSTM-sfs
	season	True	standardization	LSTM-sts
	simple	False	standardization	LSTM-pfs
GRU	group	False	standardization	GRU-gfs
	group	False	difference	GRU-gfd
	group	False	normalization	GRU-gfn
	season	False	standardization	GRU-sfs
SimpleRNN	group	False	standardization	Simple-gfs

Table 7: Network and Training Parameters Comparison

Parameters	layer design	dropout	learning rate	batch size	epochs	Architecture codename
LSTM-gfs+g,r,i	[32,32]	0.35	0.0001	512	100	ARC-32-2
LSTM-gfs+g,r,i	[64,64]	0.35	0.0001	512	100	ARC-64-2
LSTM-gfs+g,r,i	[32,32,32]	0.35	0.0001	512	100	ARC-32-3
LSTM-gfs+g,r,i	[64,64,64]	0.35	0.0001	512	100	ARC-64-3
LSTM-gfs+g,r,i	[32,32,32,32]	0.35	0.0001	512	100	ARC-32-4
LSTM-gfs+g,r,i	[64,64,64,64]	0.35	0.0001	512	100	ARC-64-4
LSTM-gfs+g	[32,32]	0.35	0.0001	512	100	ARC-32-2-g
LSTM-gfs+g	[64,64]	0.35	0.0001	512	100	ARC-64-2-g
LSTM-gfs+g	[32,32,32]	0.35	0.0001	512	100	ARC-32-3-g
LSTM-gfs+g	[64,64,64]	0.35	0.0001	512	100	ARC-64-3-g
LSTM-gfs+g	[32,32,32,32]	0.35	0.0001	512	100	ARC-32-4-g
LSTM-gfs+g	[64,64,64,64]	0.35	0.0001	512	100	ARC-64-4-g
LSTM-gfs+g,r,i	[64,64,64,64]	0.20	0.0001	512	100	ARC-64-4-do-1
LSTM-gfs+g,r,i	[64,64,64,64]	0.35	0.0001	512	100	ARC-64-4-do-2
LSTM-gfs+g,r,i	[64,64,64,64]	0.50	0.0001	512	100	ARC-64-4-do-3
LSTM-gfs+g,r,i	[64,64,64,64]	0.50	0.00001	512	100	ARC-64-4-lr-1
LSTM-gfs+g,r,i	[64,64,64,64]	0.50	0.00001	512	100	ARC-64-4-lr2
LSTM-gfs+g,r,i	[64,64,64,64]	0.35	0.0002	512	100	ARC-64-4-lr3
LSTM-gfs+g,r,i	[64,64,64,64]	0.35	0.0005	512	100	ARC-64-4-lr4
LSTM-gfs+g,r,i	[64,64,64,64]	0.35	0.001	512	100	ARC-64-4-lr5
LSTM-gfs+g,r,i	[64,64,64,64]	0.35	0.001	64	100	ARC-64-4-bs-1
LSTM-gfs+g,r,i	[64,64,64,64]	0.35	0.001	128	100	ARC-64-4-bs-2
LSTM-gfs+g,r,i	[64,64,64,64]	0.35	0.001	256	100	ARC-64-4-bs-3
LSTM-gfs+g,r,i	[64,64,64,64]	0.35	0.001	512	100	ARC-64-4-bs-4

Table 8: Bands and RNN types comparison

codename	u	g	r	i	z	g,r,i	u,g,r	g,r	u,g,r,i,z
LSTM-gfs	✓	✓	✓	✓	✓	✓	✓	✓	✓
LSTM-sfs	✓	✓	✓	✓	✓	✓	✓	✓	✓
LSTM-sts	✓	✓	✓	✓	✓		✓		✓
LSTM-pfs	✓	✓	✓	✓	✓	✓	✓	✓	✓
GRU-gfs	✓	✓	✓	✓	✓	✓	✓	✓	✓
GRU-gfd	✓	✓	✓	✓	✓		✓		✓
GRU-gfn	✓	✓	✓	✓	✓				
GRU-sfs	✓	✓	✓	✓	✓	✓	✓	✓	✓
Simple-gfs	✓	✓	✓	✓	✓	✓	✓	✓	✓

4.5 Test Methods

4.5.1 Test Sets with Added Noise

Because of the influence of weather or the limitation of detecting different levels of luminosity for telescope's cameras, the observation values are often not accurate, and their errors are also recorded. A good classifier is expected to be able to recognize quasars even though their sequences involve many sources of noise.

In order to test the classifier's generalization ability, two levels of test sets with added noise are designed. For the first one, the magnitudes in the test set are added with errors, which are the random values with the range between the positive and negative the magnitude errors for each band. The second test set includes the objects that have less than 10 observations and some extreme values that haven't been cleaned.

4.5.2 The Performance with Increasing Observations in Each Group

For testing the relationship between the number of true observations in each group and the classifier's ability in recognition, a function is designed to cut a fraction of values in each group in the test sets. In this way, it can be found that the minimum of observations in each year that the classifier can use to recognize quasars with high accuracy.

5 Results

5.1 Network and Training Parameters Comparison

In this section, the experiment results of different combinations of architecture parameters are displayed. All trained classifiers are tested by the test set without added noise. Table 9 shows all metrics calculated for each type. The AUC value in bold is the highest value in a set of comparisons. The detailed comparisons regarding to layers, dropout, learning rate and batch size are below.

1. Hidden Layers

   The number of layers and neurons defines the depth and width of the neural network. The number of neurons determines the number of features extracted in one layer, and the next layer extracts them to higher levels. For the same input data, the performance of different hidden layers and neurons will vary. Apart from the real features contained in the training set, a network that is too deep may easily learn features in the noise of the data as well, resulting in overfitting. However, a shallow network is not able to extract enough features from objects, called underfitting. Therefore, it is necessary to compare the different numbers of hidden layers and their neurons to discover the principles of building good RNN architectures.

Table 9: Result metrics for various parameter

Architecture	AUC	TP	FN	FP	TN	Precision	Recall	F1-score
ARC-32-2	0.806	2771	223	1215	1187	0.695	0.926	0.794
ARC-64-2	**0.845**	2795	199	1260	1142	0.689	0.934	0.793
ARC-32-3	0.799	2764	230	1189	1213	0.699	0.923	0.796
ARC-64-3	0.841	2798	196	1327	1075	0.678	0.934	0.786
ARC-32-4	0.698	2752	242	1228	1174	0.691	0.919	0.789
ARC-64-4	0.842	2787	207	1293	1109	0.683	0.931	0.788
ARC-32-2-g	0.798	2539	455	995	1407	0.718	0.848	0.778
ARC-64-2-g	0.810	2623	371	1161	1241	0.693	0.876	0.774
ARC-32-3-g	**0.822**	2438	556	893	1509	0.732	0.814	0.771
ARC-64-3-g	0.816	2561	433	1028	1374	0.714	0.855	0.778
ARC-32-4-g	0.813	2472	522	843	1559	0.746	0.826	0.784
ARC-64-4-g	0.765	2577	417	1067	1335	0.707	0.861	0.776
ARC-64-4-do-1	**0.856**	2777	217	1252	1150	0.689	0.928	0.791
ARC-64-4-do-2	0.842	2787	207	1293	1109	0.683	0.931	0.788
ARC-64-4-do-3	0.691	2778	216	1251	1151	0.689	0.928	0.791
ARC-64-4-lr1	0.837	2773	221	1225	1177	0.694	0.926	0.793
ARC-64-4-lr2	**0.842**	2787	207	1293	1109	0.683	0.931	0.788
ARC-64-4-lr3	0.825	2797	197	1207	1195	0.699	0.934	0.799
ARC-64-4-lr4	0.815	2786	208	1241	1161	0.692	0.931	0.794
ARC-64-4-bs-1	**0.870**	2780	214	1178	1224	0.702	0.929	0.799
ARC-64-4-bs-2	0.857	2765	229	1195	1207	0.698	0.923	0.795
ARC-64-4-bs-3	0.855	2786	208	1296	1106	0.683	0.931	0.787
ARC-64-4-bs-4	0.826	2795	199	1324	1078	0.679	0.934	0.786

Figure 16 displays AUC trends of validation sets in training. Different bands are selected for the input dataset. The RNN unit for all architectures is LSTM. It is shown that the overall AUC of g,r,i bands data reaches the peak earlier than that of g band data, and the architecture - 4 LSTM hidden layers with 32 neurons for each performs the worst among all architectures in both subplots. The others show similar trends in each subplot.

The loss trends (shown in Figure 17) reveal more differences between various architectures. The loss of g,r,i bands drops quickly, reaches the bottom, and rebounds, while the loss of g band goes down and rises slowly. In both plots, the loss trends of layers with 64 neurons grow more quickly and unstable than that with 32 neurons. The increase in loss is because the classifier learns more noise from the training set, resulting in less loss of the training set but more loss of the validation set. This is due to overfitting. When overfitting appears, the loss of a shallow network is bigger than that of a deep network. This is because the deep network has more parameters and generalization ability (reducing noise in its calculations) is better.

Figure 18 plots the ROC curve of each architecture for two input data types. The results show that a 3-layer or 4-layer architecture with 32 neurons performs well for g-band input data, but g, r, and i-band input data is more suitable for multiple layers with 64 neurons. The reason is that g, r, i-band data contains more features than g-band data, so layers with 64 neurons can extract many features from it. For g-band data, there are fewer useful features. Therefore, more neurons means more noise will be learned, resulting in poor AUC.

Overall, for different data sets, the number of neurons in each layer has a greater impact on the training effect than the number of layers. A small number of layers with 32 neurons is suitable for single-band dataset. For multi-band dataset, a deeper network with 64 neurons for each layer is preferred.

2. Dropout

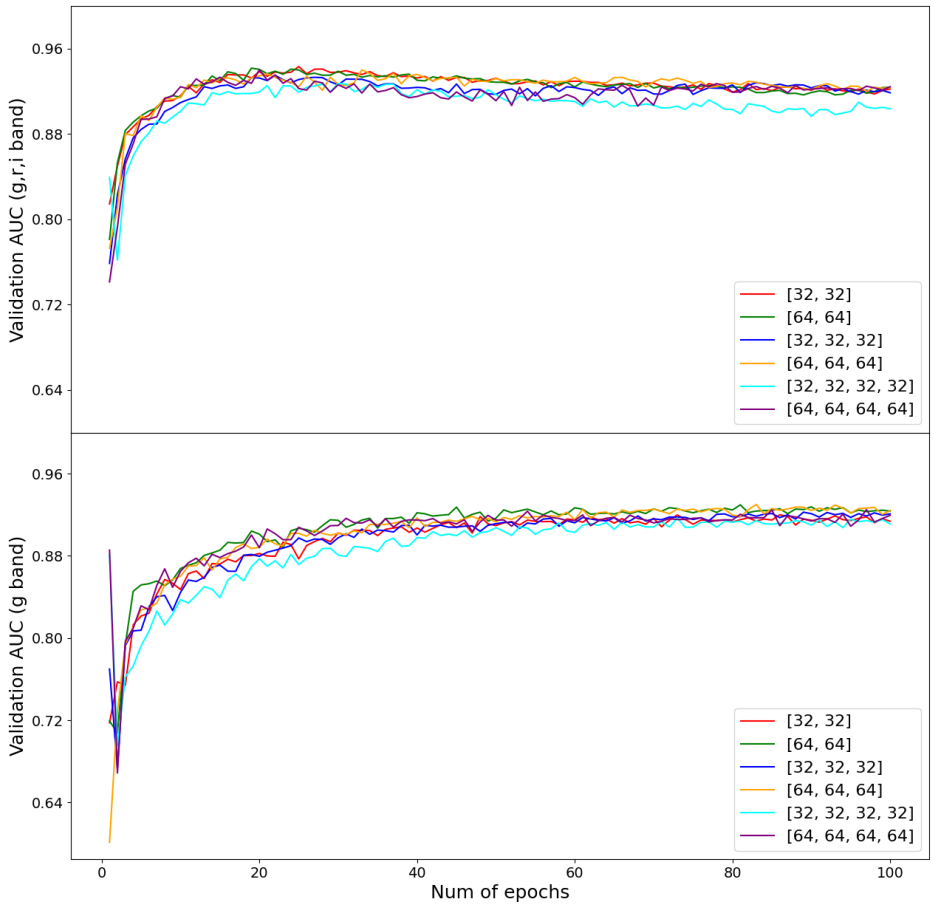


Figure 16: Depiction of trends of AUC with an increasing number of epochs for g,r,i bands and g band in the validation set respectively. Six combinations of LSTM layers and neurons are shown. For example, [32,32] means two LSTM layers with 32 neurons for each.

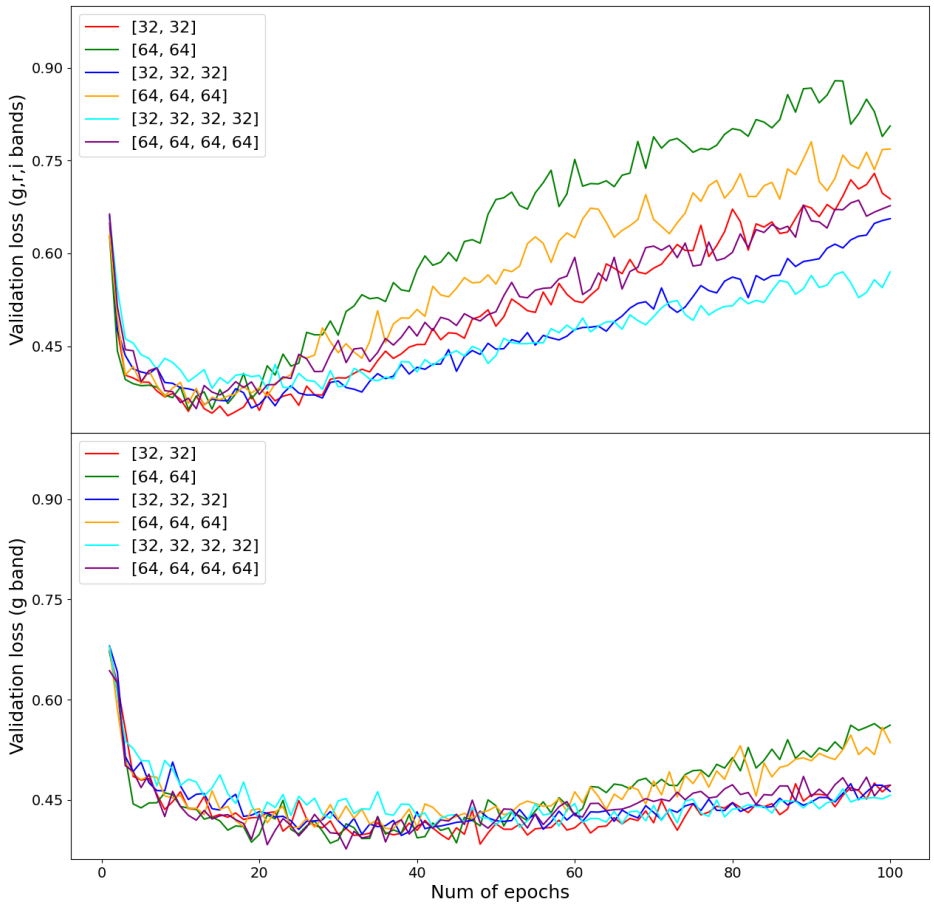


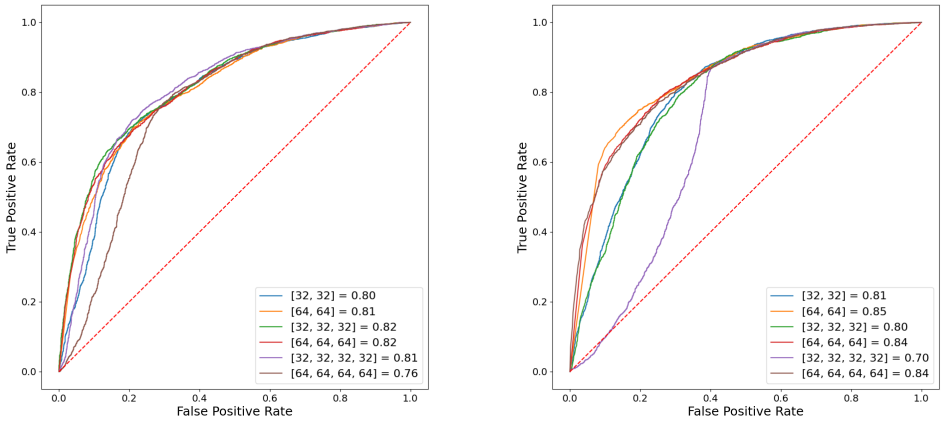
Figure 17: Depiction of trends of loss with increasing number of epochs for g band and g,r,i bands in the validation set respectively. Six combinations of LSTM layers and neurons are shown.

Dropout can ignore a proportion of neurons' outputs after each hidden layer. This method can reduce the interaction between feature detectors (hidden layer neurons). Therefore the overfitting problem is reduced. In Figure 19, the training loss of neural networks without dropout (dropout0) converges quickly on the training set during training, however, the loss on the validation set grows rapidly after around 10 epochs. Comparably, the loss trends of architectures with dropouts rises slower. This suggests that dropout can effectively reduce interactions between features which are possibly noise.

However, a large dropout is not good for training. Some essential features may also be ignored. In Figure 20, after 100 epochs, the network with dropout=0.5 gains the lowest AUC value. The optimal value for dropout is between $0.20 \sim 0.35$.

3. Learning Rate

The learning rate is the degree of adjustment of the network's parameters by the derivative of the loss function during each backpropagation process. The optimal learning rate can reach minimal loss and gain high accuracy. Figure 21 shows the validation loss and AUC for different learning



(a) g band ROC

(b) g,r,i band ROC

Figure 18: (a) and (b) are ROC plots for g band and g,r,i bands in the test set respectively. All classifiers are trained for 100 epochs. 6 combinations of LSTM layers and neurons are shown.

rates. As we can see that the loss of learning rate 10^{-5} is bigger than that of any other condition. The main reason is that the extremely small learning rate easily makes the loss function drop into a local minimum, which is still bigger than the global minimum. It is worth noting that learning rate 2e-4 reaches the lower loss minimum than learning 10^{-3}. This is because the large learning rate cannot reach the lowest loss but oscillate around it, thus influencing its training performance. For the AUC graph, the trends are generally similar, but the AUC of the minimum learning rate is always lower than other learning rates.

From Figure 22, it is shown that for g,r,i-bands dataset, learning rates around e-4 show the highest AUC.

4. Batch Size

   The batch size determines the number of training sequences fed into the input layer at a time. After a batch of data is fed and processed, the network will update all parameters according to the deviation of the loss function in the backpropagation process. A small batch size means in one epoch, all parameters will be updated frequently, resulting in longer training time. A big batch size can process data rapidly but require more memory capacity. Therefore, the correct choice of batch size is to find the best balance between memory efficiency and memory capacity.

   In Figure 23, the loss of batch size 64 shows a very unstable trend, which is caused by the high gradient oscillations between adjacent batches. For losses of large batch sizes of 256 and 512, their trends are more stable due to smoother gradient changes. However, too large batch size (512) may easily fall into the local minimum. From the AUC graph, we can also see that training with large batch size results in poor accuracy. The trend of the median batch size 256 keeps stable after reaching the peak.

   In summary, for a g, r, i band data set with an architecture with 4 LSTM layers and 64 neurons in each layer and a learning rate of e-4, the suitable batch size is 256.

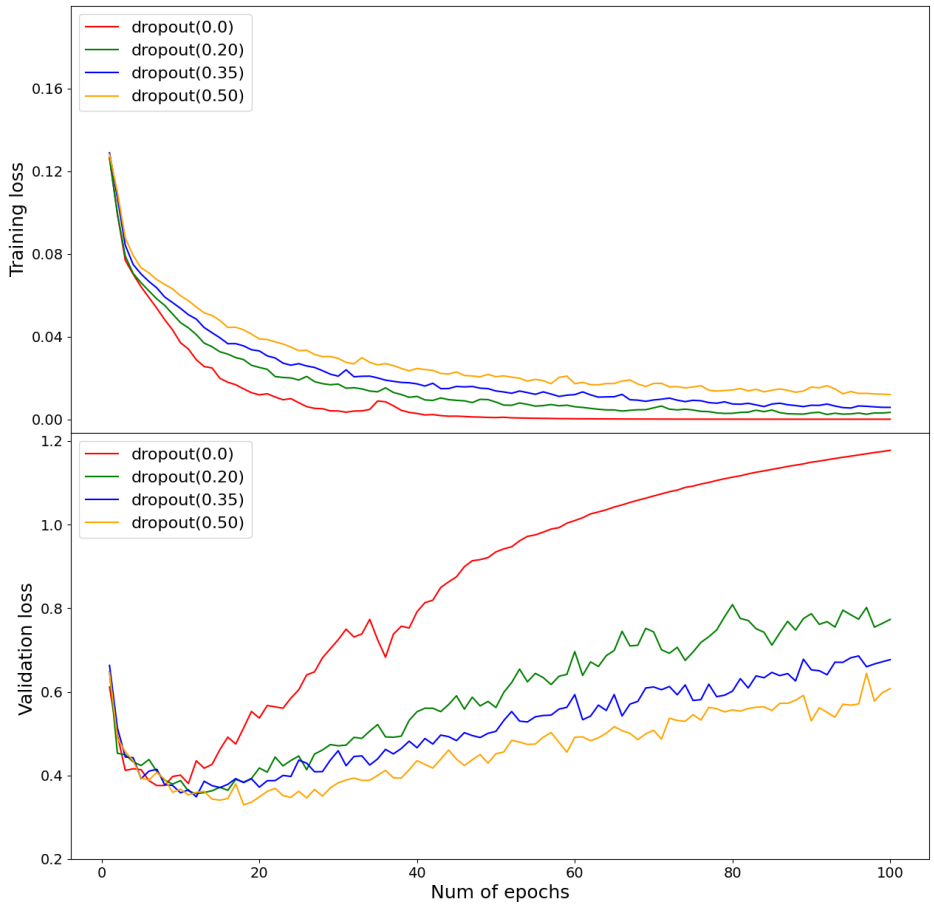


Figure 19: Trends of validation loss and training loss in training with different levels of dropout after each layer. The architecture is LSTM-gfs, g,r,i bands. The dropout occurs after each hidden layer.

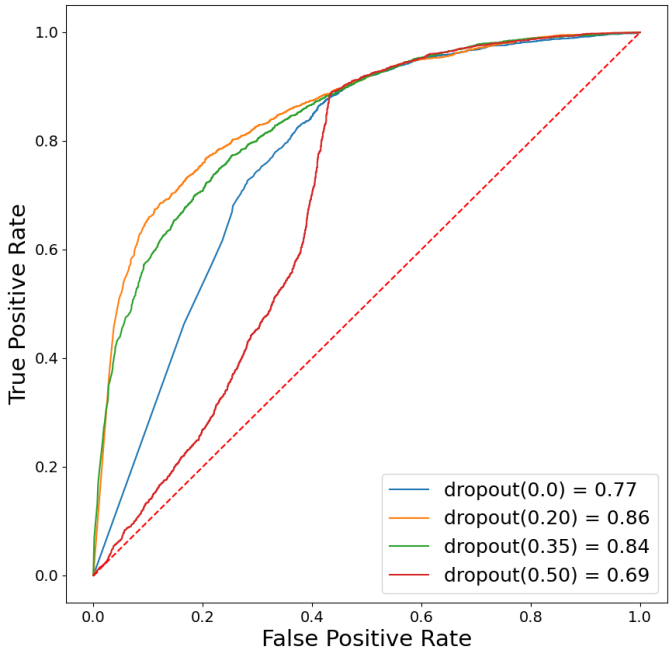


Figure 20: Test ROC for different dropout conditions after 100 epochs. The architecture is LSTM-gfs, *g*,r,i bands.

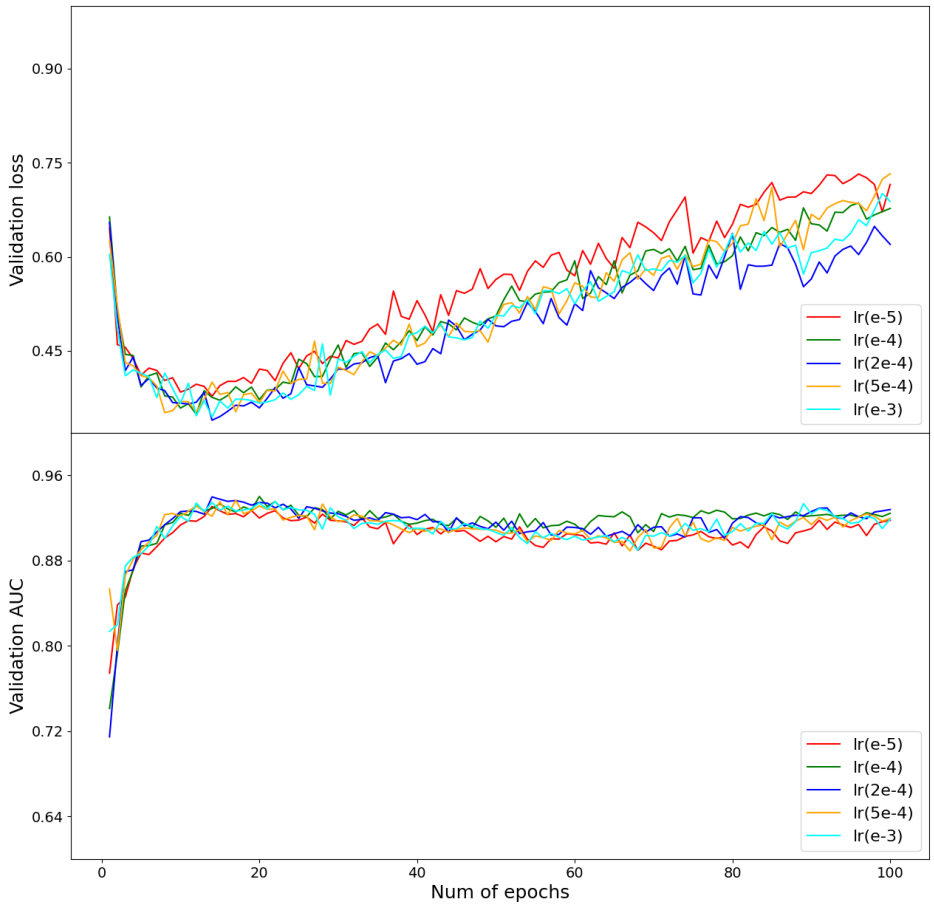


Figure 21: The validation loss and AUC trends with different learning rates. The architecture is LSTM-gfs, *g*,r,i bands.

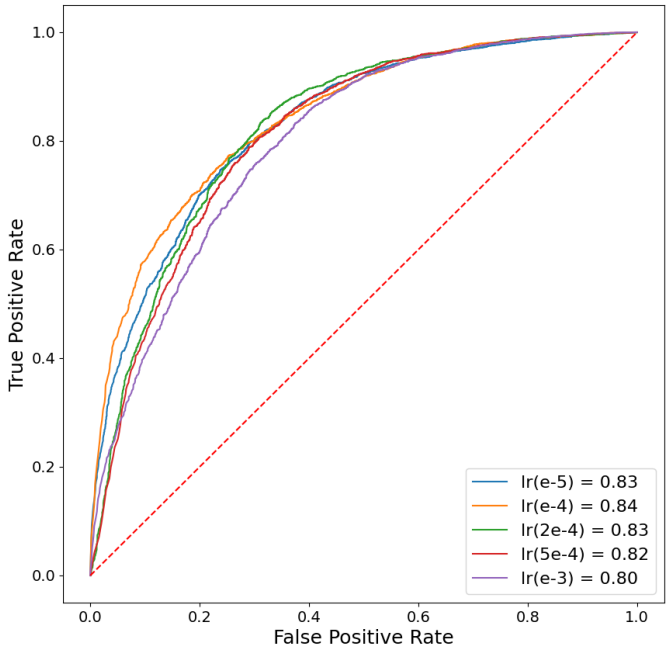


Figure 22: Test ROC for the same architecture with different learning rates. The architecture is LSTM-gfs, g,r,i bands.

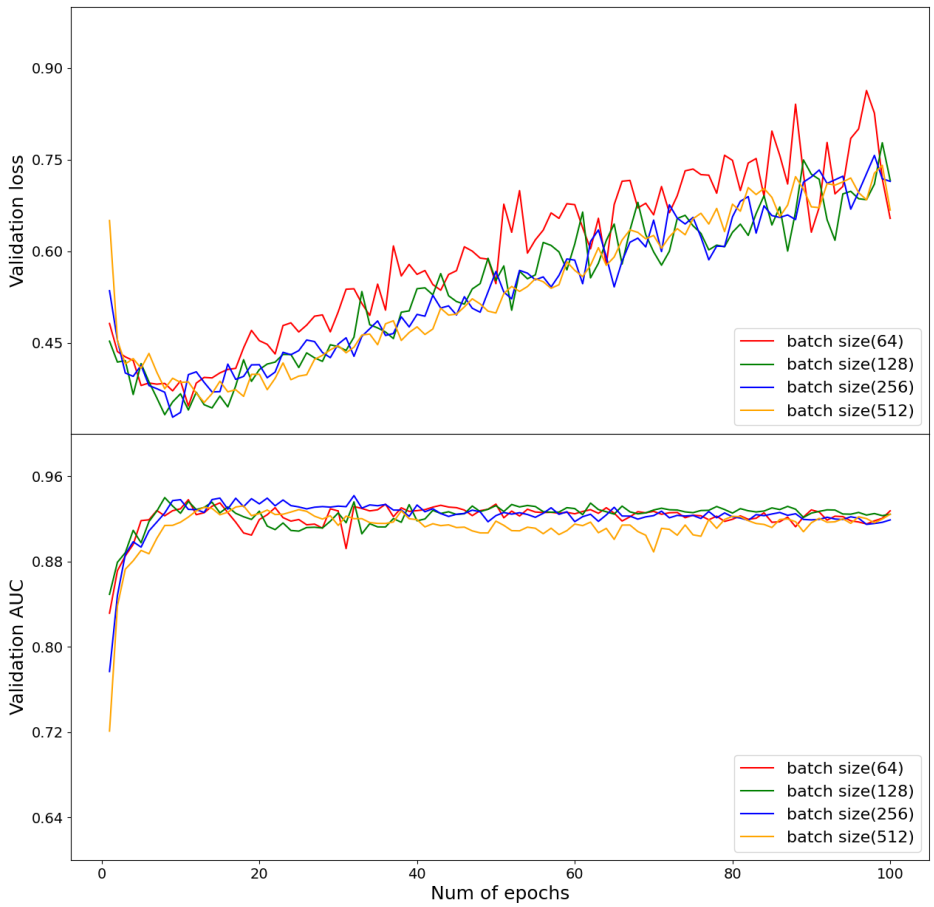


Figure 23: Validation loss and AUC for ARC-64-4bs-1,2,3,4 architectures

5.2 Bands and RNN Types Comparison

In this section, results of band comparisons with different RNN types (LSTM, GRU and SimpleRNN) and input formats are presented. In order to compare the association and difference between different bands and RNN types, we keep the key parameters of the architecture identical, which is shown in Table 10. All experiments use *Callbacks.EarlyStopping* method to stop the training progress when the loss no longer decreases in 3 epochs. It is worth noting that fully optimal architectures for each dataset with different bands combination are not applied in this case, since finding the relative performance of various configuration setups with the same architecture is the main goal.

Table 10: Key Parameters

Name	Value
Hidden layers	[64,64,64,64]
Dropout	0.35
Learning rate	0.0001
Batch size	512

5.2.1 Metrics Results

TableS 11-13 are metrics results tables for LSTM, GRU and SimpleRNN respectively. The numbers in bold are the best results in each case. In general, classifiers using the training set including g and r bands show high values in AUC, Precision, Recall and F1-score.

5.2.2 Bands Comparison

The characteristics of quasars behave differently in their light curves of different bands. From the literature, we know that for quasars, the temperature of the disk near the central black hole is very high, and a lot of high-energy EM waves are generated, which can be detected more easily in u, g and r bands.

From the upper two plots in Figure 24, it can be seen that g and r bands' curves are closest to the upper left corner among all bands. The z band classifier shows the worst accuracy. Surprisingly, u band does not perform well in classification. The AUC order from high to low is g, r, i, u, z. In the lower two plots, multiple bands are combined as one training set. It is worth noting that g,r-bands and g,r,i-bands classifiers gain the highest AUC values. Due to the negative effect of the u and z bands, the full-bands classifications with each RNN type show relatively bad results. It is obvious that for the group format, the best classifier is the g,r-band classifier of GRU layers.

5.2.3 Input Format Comparison

Apart from bands, the input format also impacts a classifier's performance. Figure 25 presents all Confusion Matrices in the *Group*, *Season* or *Simple* format with different band combinations.

Recall is an essential metric in this work since identifying more possible quasars is more critical than identifying non-quasars for rapid follow-up. For classifiers with each input format, the recall increases with more bands chosen. Compared with season and simple-format classifiers with the identical band combination, the group-format classifiers always gain higher recall. The simple-format classifiers present the worst recall (0.5).

Additionally, it can be seen that negative predictive values for season and simple-format classifiers are much higher than those in group-format classifiers. For the full-bands classifiers with the simple format, 91% of non-quasar objects are recognized.

Table 11: LSTM Results Metrics

codename	band(s)	AUC	TP	FN	FP	TN	Precision	Recall	F1-score
LSTM-gfs	u	0.696	1549	1445	535	1867	0.743	0.517	0.610
	g	0.862	2115	879	348	2054	0.859	0.706	0.775
	r	0.852	1974	1020	278	2124	**0.877**	0.659	0.753
	i	0.814	1931	1063	440	1962	0.814	0.645	0.720
	z	0.642	1807	1187	1009	1393	0.642	0.604	0.622
	g,r	0.870	2519	475	688	1714	0.785	0.841	**0.812**
	u,g,r	0.832	2546	448	894	1508	0.740	0.850	0.791
	g,r,i	**0.875**	2507	487	681	1721	0.786	0.837	0.811
	u,g,r,i,z	0.853	2637	357	948	1454	0.736	**0.881**	0.802
LSTM-sfs	u	0.643	3055	3612	1307	4018	0.700	0.458	0.554
	g	0.786	3960	2707	747	4578	0.841	0.594	0.696
	r	0.792	4013	2654	721	4604	0.848	0.602	0.704
	i	0.744	3437	3230	720	4605	0.827	0.516	0.635
	z	0.622	2525	4142	1063	4262	0.704	0.379	0.492
	g,r	0.832	4351	2316	645	4680	0.871	0.653	0.746
	u,g,r	0.852	4587	2080	630	4695	0.879	0.688	0.772
	g,r,i	0.846	4111	2556	342	4983	**0.923**	0.617	0.739
	u,g,r,i,z	**0.857**	4666	2001	641	4684	0.879	**0.700**	**0.779**
LSTM-sts	u	0.579	3352	5535	1896	5107	0.639	0.377	0.474
	g	0.660	4354	4533	1988	5015	0.687	0.490	0.572
	r	0.666	4408	4479	1908	5095	0.698	0.496	0.580
	i	0.623	3274	5613	1427	5576	0.696	0.368	0.482
	z	0.543	4466	4421	3103	3900	0.590	0.503	0.543
	u,g,r	**0.711**	5744	3143	2574	4429	0.691	**0.646**	**0.668**
	u,g,r,i,z	0.694	3606	5281	1038	5965	**0.776**	0.406	0.533
LSTM-pfs	u	0.610	907	2087	410	1992	0.689	0.303	0.421
	g	0.745	1520	1474	350	2052	0.813	0.508	0.625
	r	0.734	1355	1639	228	2174	0.856	0.453	0.592
	i	0.656	1015	1979	234	2168	0.813	0.339	0.478
	z	0.571	1718	1276	1157	1245	0.598	0.574	0.585
	g,r	0.806	1783	1211	291	2111	0.860	0.596	0.704
	u,g,r	0.826	1854	1140	274	2128	0.871	0.619	0.724
	g,r,i	0.809	1679	1315	190	2212	**0.898**	0.561	0.691
	u,g,r,i,z	**0.858**	1982	1012	235	2167	0.894	**0.662**	**0.761**

For the training time and epoch number (shown in Table 14), it is worth noting that the group format always takes the shortest training time (less than 40 seconds) than other types, and its epoch numbers are also bigger than others. The simple format often takes 4-5 minutes to train a classifier. The season format's training time is at the median level, but always spends a small number of training epochs.

5.2.4 Data Preprocessing Comparison

The classifiers with data preprocessed by different methods are also tested, which is shown in Figure 26. For each band, the AUC of classifier with standardized input data is always greater than any other formats. In the u band ROC, the differences between preprocessing methods are smaller than that in g and r bands.

Table 12: GRU Results Metrics

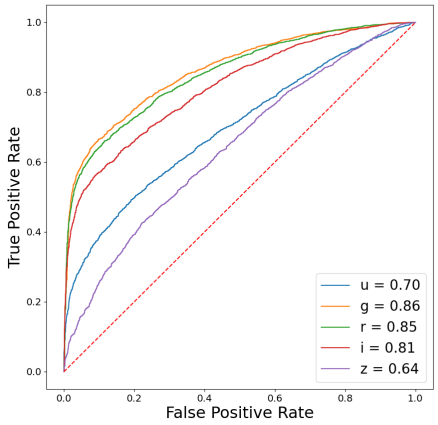
codename	band(s)	AUC	TP	FN	FP	TN	Precision	Recall	F1-score
GRU-gfs	u	0.691	1595	1399	618	1784	0.721	0.533	0.613
	g	0.856	1948	1046	239	2163	**0.891**	0.651	0.752
	r	0.855	2045	949	351	2051	0.854	0.683	0.759
	i	0.818	1928	1066	440	1962	0.814	0.644	0.719
	z	0.660	1567	1427	735	1667	0.681	0.523	0.592
	g,r	**0.893**	2358	636	407	1995	0.853	0.788	**0.819**
	u,g,r	0.879	2430	564	610	1792	0.799	0.812	0.805
	g,r,i	0.885	2508	486	640	1762	0.797	0.838	0.817
	u,g,r,i,z	0.872	2513	481	792	1610	0.760	**0.839**	0.798
GRU-gfd	u	0.664	2204	790	1158	1244	**0.656**	**0.736**	**0.694**
	g	0.593	2027	967	1301	1101	0.609	0.677	0.641
	r	0.557	2003	991	1399	1003	0.589	0.669	0.626
	i	0.564	1968	1026	1364	1038	0.591	0.657	0.622
	z	0.528	1328	1666	926	1476	0.589	0.444	0.506
	u,g,r	0.667	2097	897	1081	1321	0.660	0.700	0.680
	u,g,r,i,z	**0.668**	2157	837	1138	1264	0.655	0.720	0.686
GRU-gfn	u	0.616	1541	1453	845	1557	0.646	0.515	0.573
	g	0.607	1122	1872	564	1838	**0.665**	0.375	0.479
	r	**0.632**	1738	1256	955	1447	0.645	**0.580**	**0.611**
	i	0.619	1392	1602	709	1693	0.663	0.465	0.546
	z	0.597	1443	1551	809	1593	0.641	0.482	0.550
GRU-sfs	u	0.648	3026	3641	1304	4021	0.699	0.454	0.550
	g	0.809	4129	2538	693	4632	0.856	0.619	0.719
	r	0.795	4109	2558	777	4548	0.841	0.616	0.711
	i	0.754	3742	2925	899	4426	0.806	0.561	0.662
	z	0.611	2876	3791	1436	3889	0.667	0.431	0.524
	g,r	0.843	4169	2498	402	4923	0.912	0.625	0.742
	u,g,r	0.849	4225	2442	366	4959	**0.920**	0.634	0.751
	g,r,i	0.848	4301	2366	436	4889	0.908	0.645	0.754
	u,g,r,i,z	**0.858**	4561	2106	504	4821	0.900	**0.684**	**0.778**

Table 13: SimpleRNN Results Metrics

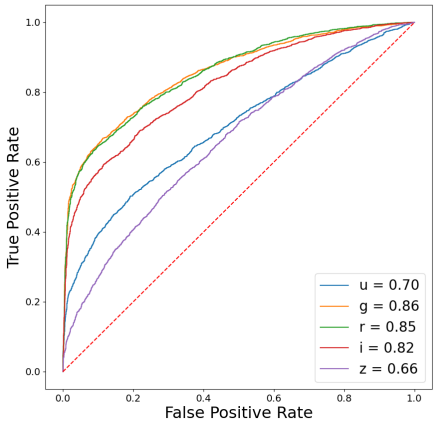
codename	band(s)	AUC	TP	FN	FP	TN	Precision	Recall	F1-score
Simple-gfs	u	0.553	1640	1354	1121	1281	0.594	0.548	0.570
	g	0.602	1652	1342	1013	1389	0.620	0.552	0.584
	r	0.596	1520	1474	899	1503	0.628	0.508	0.562
	i	0.573	1546	1448	1033	1369	0.599	0.516	0.555
	z	0.550	1559	1435	1114	1288	0.583	0.521	0.550
	g,r	0.619	1783	1211	1028	1374	0.634	0.596	0.614
	u,g,r	**0.629**	1786	1208	1008	1394	**0.639**	**0.597**	**0.617**
	g,r,i	0.588	1759	1235	991	1411	0.639	0.587	0.612
	u,g,r,i,z	0.619	1742	1252	1029	1373	**0.629**	0.582	0.604

5.2.5 Gaussian Progress Regression Tests

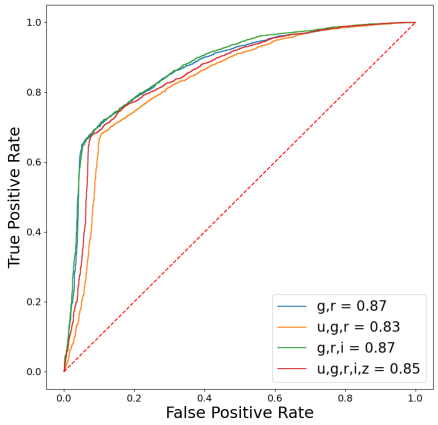
Figure 27 indicates that when the dataset is preprocessed by GPR, the classifier's accuracy drops dramatically. There are several reasons for this. The primary reason is that for each group, there are many continuous vacant positions clustered on the left part of the group due to the group design. These positions



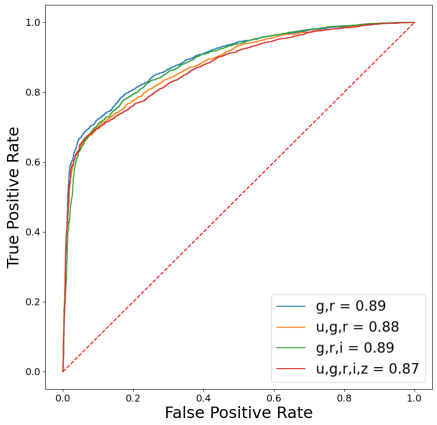
(a) LSTM-gfs: single band ROC



(b) GRU-gfs: single band ROC



(c) LSTM-gfs: multiple bands ROC

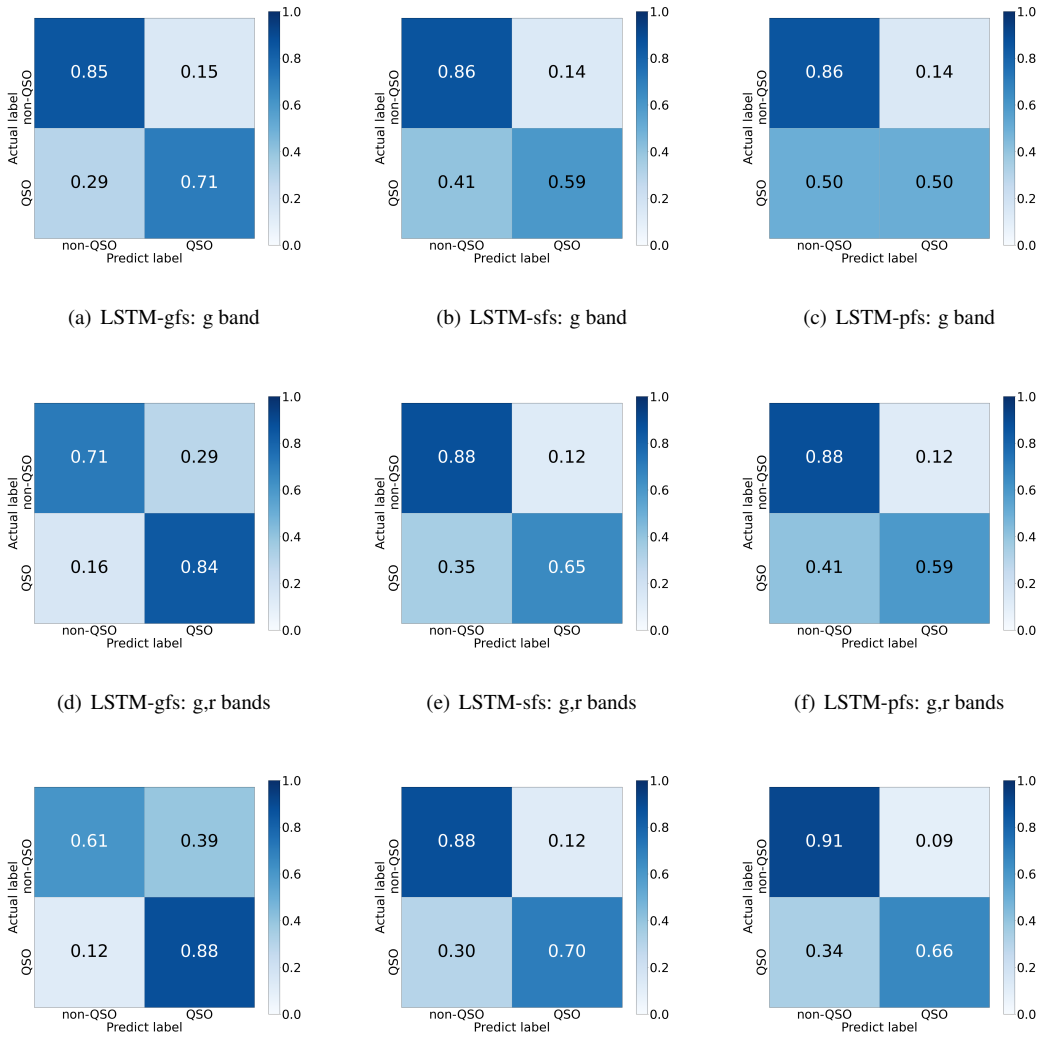


(d) GRU-gfs: multiple bands ROC

Figure 24: ROC plots for single-band and multi-band training set for LSTM and GRU networks with the group format

Table 14: Input Formats' Training Time and Epochs Comparison

input format	bands	training time(s)	epochs
group	g	32.851	12
	g,r	40.345	17
	u,g,r,i,z	33.982	12
season	g	137.562	5
	g,r	140.028	5
	u,g,r,i,z	265.192	9
simple	g	247.599	10
	g,r	327.974	9
	u,g,r,i,z	225.179	9



(a) LSTM-gfs: g band (b) LSTM-sfs: g band (c) LSTM-pfs: g band

(d) LSTM-gfs: g,r bands (e) LSTM-sfs: g,r bands (f) LSTM-pfs: g,r bands

(g) LSTM-gfs: u,g,r,i,z bands (h) LSTM-sfs: u,g,r,i,z bands (i) LSTM-pfs: u,g,r,i,z bands

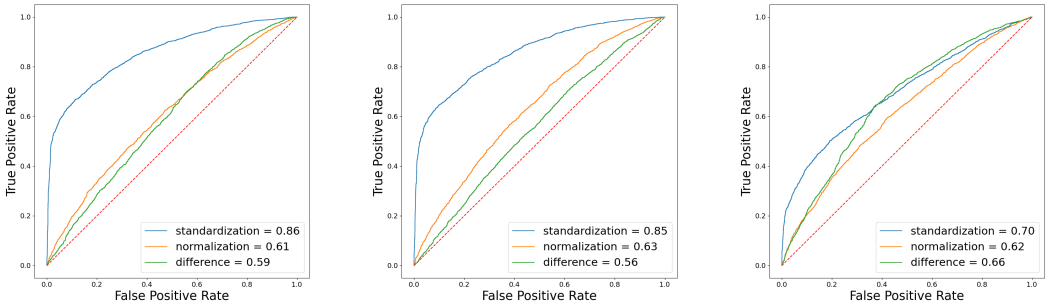
Figure 25: Confusion matrixes in group, season and simple formats for g band, (g, r) bands, (u, g, r, i, z) bands classifiers.

can easily have extremely low values (such as 0) according to the GPR principle which is unrealistic. Another reason could be the parameter setting issue for GPR. RBF kernel may not be suitable for quasars' light curves since their light curves tend to be quasi-random. Due to the calculation method of RBF, because the positions of some continuous missing values are close to surrounding training values, their predicted values will also be close to those values and relatively stable, which doesn't match typical data.

5.2.6 RNN Type Comparison

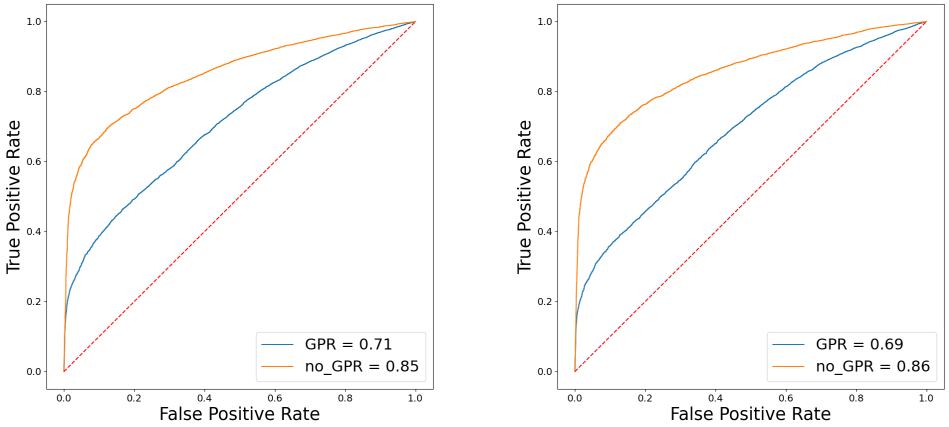
It is important to know which type of RNN suits the quasar/non-quasar classification better. In Figure 28, it can be found that GRU and LSTM networks show much better AUC than SimpleRNN networks. It proves that in the light curves of quasars and non-quasars, multiple distant observations are potentially correlated. We can also see that a GRU network is slightly better than LSTM for each band combination.

For the training time and epoch number for each RNN (shown in Table 15, the LSTM and GRU are



(a) GRU-group: g band (b) GRU-group: r band (c) GRU-group: u band

Figure 26: The ROC of g/r/u band GRU classifiers with different data preprocessing methods (standardization, normalization, difference).



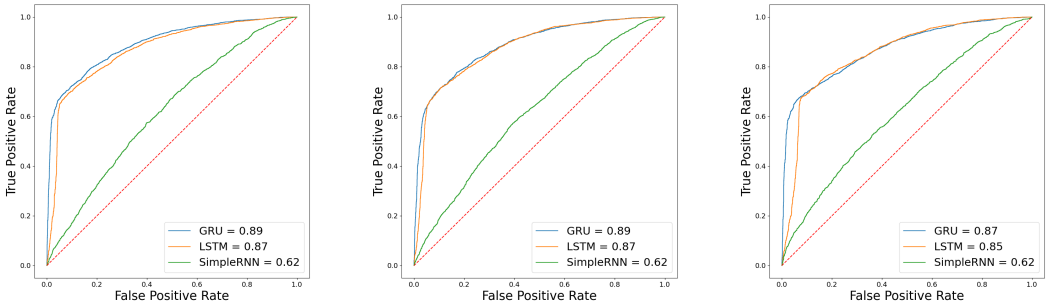
(a) LSTM-season: u,g,r bands (b) LSTM-season: u,g,r,i,z bands

Figure 27: The ROC of LSTM classifiers with GPR or not. The method of applying GPR is to replace all data points to regressed data

similar in general, however, SimpleRNN takes the shortest training time and smallest numbers of epochs due to its simple calculations.

Table 15: Different RNNs' Training Time and Epochs Comparison

RNN type	bands	training time(s)	epochs
LSTM	g,r	40.345	17
	g,r,i	34.686	13
	u,g,r,i,z	33.982	12
GRU	g,r	36.979	13
	g,r,i	40.898	17
	u,g,r,i,z	34.278	12
SimpleRNN	g,r	19.911	9
	g,r,i	16.794	5
	u,g,r,i,z	11.967	7

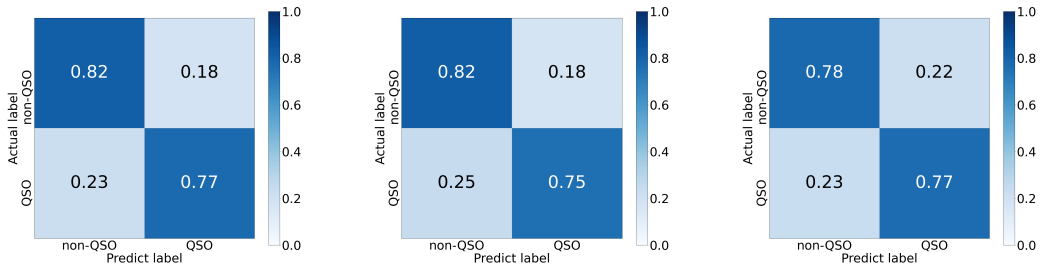


(a) group: g,r bands (b) group: g,r,i bands (c) group: u,g,r,i,z bands

Figure 28: ROC of three RNN types of classifiers with different band combinations training data

5.2.7 Generalization Tests

The best classifier in all tests is the GRU g, r-band classifier with the group input format. To test its generalization ability, the classifier is evaluated with three different levels of test sets. In Figure 29, the classifier gains a good recall for the original test set. The recall gets slightly lower when it is measured by the test set with magnitude errors. For the test set with raw data, a little more non-quasars are classified wrongly, however, the quasar recognition remains stable.

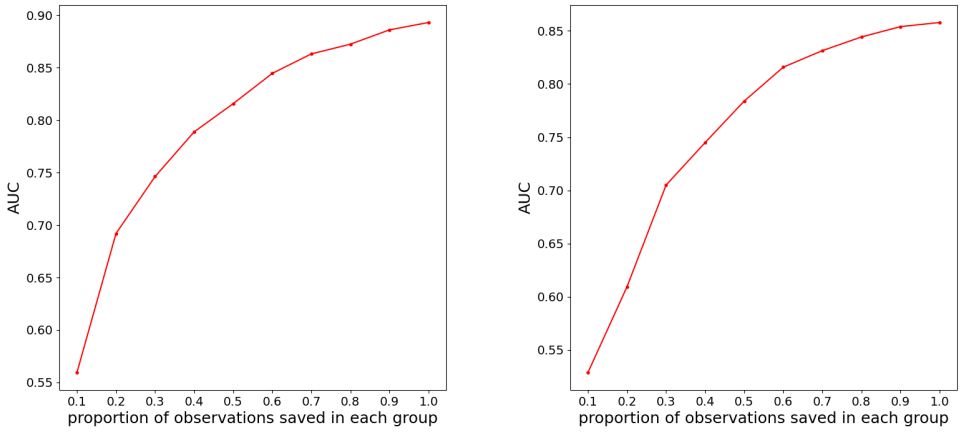


(a) original test set (b) test set with magnitude errors (c) test set with raw data

Figure 29: Confusion Matrices of the optimal classifier with three test sets.

5.2.8 Prediction Ability with Different Numbers of Observations

It is interesting to know how the effect of the number of observations in each group on the accuracy of the classifier. In Figure 30, the two classifiers trends rise quickly and then gradually level off. When over 60% observations (for the original test set, it is about 40 observations) are saved, the AUC of both classifiers are over 0.8.



(a) GRU-gfs: g,r bands

(b) GRU-sfs: u,g,r,i,z bands

Figure 30: AUC trends of two optimal classifiers with the increasing proportion of observations in each group

6 Discussion

From the results discussed in Chapter 5, it is shown that g and r bands make the biggest impact to the accuracy of our classifiers, and the group input format shows good recall in different cases and can efficiently train our classifiers. The optimal classifier is the GRU network (4 GRU layers with 64 neurons in each layer) with the group input format of g,r-band training set.

During labeling the SDSS Strip data with standard catalogs, we found that there are many objects which are very possibly quasars but not listed in the DR14Q and DR16Q catalogs. Before the development of classifiers, those objects were labelled as *unknown*. Now we are able to use the best classifier to predict their labels, which are expected to be considered for SDSS managers and astronomers to add new quasars in the next released version quasar catalog.

This research can be applied for ZTF datasets and upcoming LSST data streams to attempt quasar classification. The key to this research is the special input format design. The unique observation cadence for our quasars from SDSS is utilized in the program development. In addition, the group input format is proved to improve the accuracy and efficiency. Other parameters can be adjusted in different cases. For the ZTF dataset with g and r bands observations, the optimal classifier (which only needs g,r band data) is a perfect match. As for the future LSST data with 6 bands, our classifier only needs two bands to achieve high accuracy in recognizing quasars. In addition, the relatively high accuracy for the season format shows that when a new quasar is detected and has over 40 observations during a year, the season format classifier can recognize it with over 80% confidence.

The era of astronomical big data has arrived. Data from transient and variable objects needs to be processed, stored, and classified efficiently and effectively. The rich information contained in these data will bring interesting new discoveries and theories to the astronomy world. In addition, when a GW event is detected, a precious event on the sky is happening, which could be BBH, SNe, etc. It is essential to find the GW source in the targeted sky area quickly, and this will be an interesting application of this research.

This work also has some limitations. The main issue is the unbalanced training data. Because of the big difference in proportions of quasars and non-quasars in our dataset, the light curves of our non-quasars may not represent all characteristics of existing objects in the sky. Additionally, the non-quasars in our datasets are assumed to be stars. In reality, galaxies, SNs, neutron stars, and more are all possibly contained in the test set. Therefore, the classifier's performance may be poor when the test set involves too many different labelled non-quasars. This problem can be solved when new training data with different types of celestial objects is provided. Another problem is that the grouping method may cause many useful objects' observations to be dropped and many data points left empty. This can be adjusted by the user-defined parameter *group size*. However, it is always a trade-off issue. If the *group size* is big, more empty data points occurs; if it is small, more useful values are removed.

For this project, there were two disturbances that affect work efficiency and results: COVID-19 isolation and Cirrus upgrade. COVID-19 has affected this research in many ways. All discussions about this work with my tutors are online, and the library and Bayes Center are closed. Studying in a small dormitory can easily become difficult. I often felt unable to direct my research effectively. With the encouragement of my tutors, I was mostly able to complete the work required. However, I think if I was able to study in the library or EPCC and communicate with teachers and classmates in person, my research efficiency would have been greatly improved. The downtime of Cirrus also slowed the progress of the research. As the volume of astronomical data is big, GPU on Cirrus is required to be applied for this research. Before the Cirrus update, I spent over a week trying to install the Tensorflow-gpu image with the correct version matched with the Cirrus system in singularity. It is very tricky to look for the the correct Tensorflow since a lot of unexpected errors appeared. After Cirrus' downtime, the CUDA version was updated, and singularity was no longer working as before. Therefore, it took me another three days to create a new virtual environment and install the Tensorflow-gpu successfully. In addition, all previous tests had to be reprocessed to ensure sensible comparisons for this work.

7 Future work

There is still a lot of work that could take this research further.

The first is to construct new dataset with various types of non-quasars data and train new classifiers. The optical classifier is supposed to recognize quasars with high confidence from different celestial objects, which includes galaxies, stars, neutrons, etc.

The grouping method also needs to be improved. In the current design, there are a lot of empty positions filled with 0 in each group, which has a negative impact on classifiers' training performance. The idea of filling them by applying Gaussian Progress Regression with the RBF kernel has proved to be not appropriate. In the further research, it is expected that the GPR with more suitable kernels can be tested again. In addition, replacing the real observation values with combinations of time difference and magnitude difference is also worth investigating.

Well-trained classifiers could then be tested against the ZTF datasets on Lasair. They will be compared with other classifiers, such as Sherlock[32]. We also expect that our classifiers can recognize GW sources when the GWs are held in quasars. When a new GW alert is received by Lasair (shown on Lasair's Skymap), all its most probable galaxies are listed on its information page[33]. Our classifier could predict the possibility of being a quasar for each source using their observation records. The predictions can help astronomers narrow the search area and objects.

Finally, when the classifier is proved to gain excellent accuracy in many tests, we expect that it can deal with the coming LSST data streams on Lasair. To digest the big data efficiently, it is necessary to build a preprocessing system and databases connected with the classifier.

[32] https://lasair.roe.ac.uk/sherlock
[33] https://lasair.roe.ac.uk/skymap/S200316bj_3/

8 Conclusions

Astronomical big data from sky surveys is important in many research fields, especially transients and variables classification. Machine learning techniques are widely applied in this field, which includes RNN that use time-series data to classify objects. In recent years, research objects in the astronomical classification field are often various SNe, variable stars and simulated data. There are less classifiers which focus on identifying quasars only based on their light curves. The reason could be quasars' special features: the light curves are unpredictable in a short term but converge in a large timescale. In this research, we tried to build an effective classifier to identify quasars.

The observation cadence in light curves is considered in the classifier design. Three input formats: $Group$, $Season$ and $Simple$ were implemented. The $Group$ input format is suitable for predicting objects with observations of years; The $Season$ is able to only use over 40 observations of objects to achieve high-accuracy classification. The $Simple$ simply regards original light curves as input sequences and performs as a benchmark for other two input formats. Additionally, we investigated three RNN types LSTM, GRU and SimpleRNN with different input formats and bands combinations. In this project, the key results are:

- g and r bands have the most obvious characteristics in distinguishing quasars and non-quasars. This is because the unstable energy emissions of quasars are mainly on the ultraviolet and very blue part of the EM spectrum, where g and r bands are the most sensitive.
- Results of LSTM networks with $Group$ format always gain high recall in combinations of different bands. $Season$ and $Simple$ formats show better ability in recognizing non-quasars. However, $Simple$ format often gains lower recall.
- Classifiers with $Group$ format only take less than 41 seconds training time, which are the most efficient among all input formats.
- GRU network performs slightly better than LSTM network.
- The AUC of the optimal classifier (4 GRU layers with 64 neurons for each, group format, g, r-bands) is up to 0.893.

We expect that our research can be applied in identifying quasars in all GW sources as well as coming LSST data streams. There are still some work that we would like to do. The primary is to find a new training set which contains different types of objects' light curves, such as SNe, galaxies and neutron stars. Another is to improve the accuracy of the classification, which involves further investigation in data augmentation (GPR) and more suitable network architectures.

9 Appendix

9.1 SQL Queries for Generating the Quasar-targeted Dataset

```
select m.objid, a.ra, a.dec

into MyDB.QuasarMatchHeads

from MyDB.Stripe82_DR16_QuasarTargets_unique a
  CROSS APPLY dbo.fGetNearestObjEq(a.ra, a.dec, 1.) b
  JOIN MatchHead m ON m.objid = b.objid


select h.ra, h.dec,
  m.objID1, m.objID2,
  f.mjd_u,       f.mjd_g,       f.mjd_r,       f.mjd_i,       f.mjd_z,
  p.psfMag_u,    p.psfMag_g,    p.psfMag_r,    p.psfMag_i,    p.psfMag_z,
  p.psfMagErr_u, p.psfMagErr_g, p.psfMagErr_r, p.psfMagErr_i, p.psfMagErr_z
INTO MyDB.QuasarMatches_wErr
from MyDB.QuasarMatchHeads h
  JOIN Match m    ON m.objID1=h.objID
  JOIN PhotoTag p ON p.objID = m.objID2
  JOIN Field f    ON p.fieldID=f.fieldID
```

9.2 Algorithms and Related Tables

Table 16: Grouping-related Information

	list	definition
Each object	obj_delta_time	a list of one object's time differences between neighboring MJDs
	obj_num_group	the number of groups in one object
	idx_group_gap	a list of indexes in obj_delta_time which the delta time is bigger than remove_check_delta (233 days)
	obj_group_size	a list of group sizes for one object. The size means the time differences between the earliest and latest MJDs in one group
	obj_group_mjd	a list of boundary MJDs in each group for one object. For example, [[0,45],[264,278],[500,532]]
All objects	id_list	a list of all objects' id
	size_group	a list of all objects' obj_group_size lists
	mjd_group	a list of all objects' obj_group_mjd lists
	num_group	a list of all objects'obj_num_group values
Useful values	max_group	the maximal group number among all objects
	min_group	the minimal group number among all objects
	max_group_size	the maximal group size among all objects
	mean_group_size	the mean group size among all objects
	base_mjd	the latest observation MJD among all objects

Algorithm 2 Remove Alone MJD

```
1:  procedure <REMOVE_ALONE_MJD>(object_list, data, remove_check_delta , min_size)
2:      Initialize record_index list
3:      for Every object in object_list do
4:          Generate the list of indexes of this object in data dataframe: ind_list
5:          Generate the list of all MJDs
6:          Initialize warn_index list
7:          for Each time difference between two neighboring MJDs do
8:              if time_diff > remove_check_delta  then
9:                  Add the early MJD index to warn_index list
10:             end if
11:         end for
12:         if len(warn_index) != 0 then
13:             for Each idx in warn_index do
14:                 if idx == 0 and warn_index[0] - ind_list[0] <= min_size then
15:                     Add list(range(ind_list[0],warn_index[0]+1)) to record_index list
16:                 end if
17:                 if warn_index[idx+1] - warn_index[idx] <= min_size then
18:                     Add list(range(warn_index[idx]+1, warn_index[idx+1]+1)) to record_index list
19:                 end if
20:             end for
21:         end if
22:     end for
23:     In data dataframe, delete the indexes in record_index and generate the new_data dataframe
24:     return new_data
25: end procedure
```

Algorithm 3 Calculate Boundary MJDs

Initialize a empty list $group\_bound$
if sequence_length == None **then**
    sequence_length = max_group ▷ max_group is the largest number of groups among all objects
end if
bound_left = base_mjd - mean_group_size - suit_delta
bound_right = base_mjd ▷ base_mjd is the latest MJD among all objects. mean_group_size is the mean size of all groups
n = 1
while n <= sequence_length **do**
    Initialize empty lists: obj_bound_pre, obj_bound_post
    for obj_group_mjd in mjd_group **do**
        Initialize an empty list $match$
        for each group_mjd in obj_group_mjd **do**
            if bound_left <= group_mjd[left_mjd] <= bound_right **then**
                Append group_mjd to match list
            end if
        end for
        if len(match) == 1 **then** ▷ if only one group_mjd in the bound range
            obj_bound_pre.append(match[0][0])
            obj_bound_post.append(match[0][-1]) ▷ Append the group_mjd's left mjd and right mjd to obj_bound_pre and obj_bound_post respectively
        end if
    end for
    if len(obj_bound_pre) > 0 and len(obj_bound_post) > 0 **then**
        group_bound.append([min(obj_bound_pre),max(obj_bound_post)]) ▷ Combine the minimal value in obj_bound_pre and the maximal value in obj_bound_post as a list, and append it to group_bound list
        gap = max(obj_bound_post)-min(obj_bound_pre)
        bound_left = bound_left - gap - suit_delta
        bound_right = bound_right - gap - suit_delta
    else
        bound_left = bound_left - mean_group_size - suit_delta
        bound_right = bound_right - mean_group_size - suit_delta
    end if
    n += 1
end while
Initialize an empty list model_group_bound
if group_size == None **then** ▷ group_size is a user-defined parameter
    group_size = max_group_size ▷ max_group_size is the largest group size among all groups
end if
for mjd_bound in group_bound **do**
    new_left_bound = int(mjd_bound[1]-group_size) ▷ The new left bound value is an integer that is the result of reducing group _size from the right bound
    model_group_bound .append([new_left_bound,int(mjd_bound[1])])
end for
return model_group_bound

9.3 LSTM and GRU Formulas

(1) LSTM unit

forget gate:

$$f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}\right) \tag{10}$$

external input gate:

$$g_i^{(t)} = \sigma\left(b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)}\right) \tag{11}$$

cell internal state:

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma\left(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)}\right) \tag{12}$$

output gate:

$$q_i^{(t)} = \sigma\left(b_i^o + \sum_j U_{i,j}^o x_j^{(t)} + \sum_j W_{i,j}^o h_j^{(t-1)}\right) \tag{13}$$

output :

$$h_i^{(t)} = tanh\left(s_i^{(t)}\right) q_i^{(t)} \tag{14}$$

(2) GRU unit

update gate:

$$u_i^{(t)} = \sigma\left(b_i^u + \sum_j U_{i,j}^u x_j^{(t)} + \sum_j W_{i,j}^u h_j^{(t-1)}\right) \tag{15}$$

reset gate:

$$r_i^{(t)} = \sigma\left(b_i^r + \sum_j U_{i,j}^r x_j^{(t)} + \sum_j W_{i,j}^r h_j^{(t-1)}\right) \tag{16}$$

output:

$$h_i^{(t)} = u_i^{(t-1)} h_i^{(t-1)} + (1 - u_i^{(t-1)}) \sigma\left(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} r_i^{(t-1)} h_j^{(t-1)}\right) \tag{17}$$

References

[1] I. Becker, K. Pichara, M. Catelan, P. Protopapas, C. Aguirre, and F. Nikzat. Scalable end-to-end recurrent neural network for variable star classification. *MNRAS*, 493(2):2981–2995, Apr 2020.

[2] Siddharth Chaini and Soumya Sanjay Kumar. Astronomical Classification of Light Curves with an Ensemble of Gated Recurrent Units. *arXiv e-prints*, page arXiv:2006.12333, June 2020.

[3] Tom Charnock and Adam Moss. Deep Recurrent Neural Networks for Supernovae Classification. *ApJ Lett.*, 837(2):L28, March 2017.

[4] Eric W. Flesch. The million quasars (milliquas) catalogue, v6.4, 2019.

[5] Felix Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12:2451–71, 10 2000.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[7] M. J. Graham, K. E. S. Ford, B. McKernan, N. P. Ross, D. Stern, K. Burdge, M. Coughlin, S. G. Djorgovski, A. J. Drake, D. Duev, M. Kasliwal, A. A. Mahabal, S. van Velzen, J. Belecki, E. C. Bellm, R. Burruss, S. B. Cenko, V. Cunningham, G. Helou, S. R. Kulkarni, F. J. Masci, T. Prince, D. Reiley, H. Rodriguez, B. Rusholme, R. M. Smith, and M. T. Soumagnac. Candidate electromagnetic counterpart to the binary black hole merger gravitational-wave event s190521g. *Phys. Rev. Lett.*, 124:251102, Jun 2020.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[9] Željko Ivezić and Chelsea MacLeod. Optical variability of quasars: a damped random walk. In Areg M. Mickaelian and David B. Sanders, editors, *Multiwavelength AGN Surveys and Studies*, volume 304 of *IAU Symposium*, pages 395–398, July 2014.

[10] Željko Ivezić, J. Allyn Smith, Gajus Miknaitis, Huan Lin, Douglas Tucker, Robert H. Lupton, James E. Gunn, Gillian R. Knapp, Michael A. Strauss, Branimir Sesar, Mamoru Doi, Masayuki Tanaka, Masataka Fukugita, Jon Holtzman, Steve Kent, Brian Yanny, David Schlegel, Douglas Finkbeiner, Nikhil Padmanabhan, Constance M. Rockosi, Mario Jurić, Nicholas Bond, Brian Lee, Chris Stoughton, Sebastian Jester, Hugh Harris, Paul Harding, Heather Morrison, Jon Brinkmann, Donald P. Schneider, and Donald York. Sloan Digital Sky Survey Standard Star Catalog for Stripe 82: The Dawn of Industrial 1% Optical Photometry. *AJ*, 134(3):973–998, Sept 2007.

[11] Sara Jamal and Joshua S. Bloom. On Neural Architectures for Astronomical Time-series Classification with Application to Variable Stars. *arXiv e-prints*, page arXiv:2003.08618, March 2020.

[12] R. Kessler, G. Narayan, A. Avelino, E. Bachelet, R. Biswas, P. J. Brown, D. F. Chernoff, A. J. Connolly, M. Dai, S. Daniel, R. Di Stefano, M. R. Drout, L. Galbany, S. González-Gaitán, M. L. Graham, R. Hložek, E. E. O. Ishida, J. Guillochon, S. W. Jha, D. O. Jones, K. S. Mand el, D. Muthukrishna, A. O'Grady, C. M. Peters, J. R. Pierel, K. A. Ponder, A. Prša, S. Rodney, V. A. Villar, LSST Dark Energy Science Collaboration, and Transient and Variable Stars Science Collaboration. Models and Simulations for the Photometric LSST Astronomical Time Series Classification Challenge (PLAsTiCC). *PASP*, 131(1003):094501, Sept 2019.

[13] Richard Kessler, Alex Conley, Saurabh Jha, and Stephen Kuhlmann. Supernova Photometric Classification Challenge, Jan 2010.

[14] O. G. King, T. Hovatta, W. Max-Moerbeck, D. L. Meier, T. J. Pearson, A. C. S. Readhead, R. Reeves, J. L. Richards, and M. C. Shepherd. A quasi-periodic oscillation in the blazar J1359+4011. *MNRAS*, 436:L114–L117, Nov 2013.

[15] Daniel Muthukrishna, Gautham Narayan, Kaisey S. Mandel, Rahul Biswas, and Renée Hložek. RAPID: Early Classification of Explosive Transients Using Deep Learning. *PASP*, 131(1005):118002, Nov 2019.

[16] Brett Naul, Joshua S. Bloom, Fernando Pérez, and Stéfan van der Walt. A recurrent neural network for classification of unevenly sampled variable stars. *Nature Astronomy*, 2:151–155, Nov 2018.

[17] Isabelle Pâris, Patrick Petitjean, Éric Aubourg, Adam D. Myers, Alina Streblyanska, Brad W. Lyke, Scott F. Anderson, Éric Armengaud, Julian Bautista, Michael R. Blanton, and et al. The sloan digital sky survey quasar catalog: Fourteenth data release. *Astronomy & Astrophysics*, 613:A51, May 2018.

[18] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, Oct 1986.

[19] M. Schmidt. 3c 273 : A star-like object with large red-shift. *Nature*, 197(4872):1040–1040, Mar 1963.

[20] H. E. Smith, E. O. Smith, and H. Spinrad. The revised 3c catalog of radio sources - a review of optical identifications and spectroscopy. *Publications of the Astronomical Society of the Pacific*, 88:621, Oct 1976.

[21] Yutaro Tachibana, Matthew J. Graham, Nobuyuki Kawai, S. G. Djorgovski, Andrew J. Drake, Ashish A. Mahabal, and Daniel Stern. Deep modeling of quasar variability. *arXiv e-prints*, page arXiv:2003.01241, March 2020.

[22] Yongheng Li Yanxia Zhang, Chenzhou Cui. Big data and paradigm shift for astronomy in the 21st century. *Big Data*, 2(06):65–74, 2016.