Monthly Notices of the ROYAL ASTRONOMICAL SOCIETY

MNRAS **505,** 4345–4361 (2021)
Advance Access publication 2021 May 28

https://doi.org/10.1093/mnras/stab1545

Light-curve classification with recurrent neural networks for GOTO: dealing with imbalanced data

U. F. Burhanudin,[1★] J. R. Maund,[1] T. Killestein,[2] K. Ackley,[3,4] M. J. Dyer,[1] J. Lyman,[2] K. Ulaczyk,[2] R. Cutter,[2] Y.-L. Mong,[3,4] D. Steeghs,[2,4] D. K. Galloway,[3,4] V. Dhillon,[1,5] P. O'Brien,[6] G. Ramsay,[7] K. Noysena,[8] R. Kotak,[9] R. P. Breton,[10] L. Nuttall,[11] E. Pallé,[5] D. Pollacco,[2] E. Thrane,[3] S. Awiphan,[8] P. Chote,[2] A. Chrimes,[2] E. Daw,[1] C. Duffy,[7] R. Eyles-Ferris,[6] B. Gompertz,[2] T. Heikkilä,[9] P. Irawati,[8] M. R. Kennedy,[10] A. Levan,[2] S. Littlefair,[1] L. Makrygianni,[1] D. Mata-Sánchez,[10] S. Mattila,[9] J. McCormac,[2] D. Mkrtichian,[8] J. Mullaney,[1] U. Sawangwit,[8] E. Stanway,[2] R. Starling,[6] P. Strøm,[2] S. Tooke[6] and K. Wiersema[2]

[1]*Department of Physics and Astronomy, University of Sheffield, Sheffield S3 7RH, UK*
[2]*Department of Physics, University of Warwick, Gibbet Hill Road, Coventry CV4 7AL, UK*
[3]*School of Physics & Astronomy, Monash University, Clayton, VIC 3800, Australia*
[4]*OzGRav-Monash, School of Physics and Astronomy, Monash University, Clayton, VIC 3800, Australia*
[5]*Instituto de Astrof'isica de Canarias, E-38205 La Laguna, Tenerife, Spain*
[6]*School of Physics & Astronomy, University of Leicester, University Road, Leicester LE1 7RH, UK*
[7]*Armagh Observatory and Planetarium, College Hill, Armagh BT61 9DG, UK*
[8]*National Astronomical Research Institute of Thailand, 260 Moo 4, T. Donkaew, A. Maerim, Chiangmai 50180, Thailand*
[9]*Department of Physics & Astronomy, University of Turku, Vesilinnantie 5, Turku FI-20014, Finland*
[10]*Jodrell Bank Centre for Astrophysics, Department of Physics and Astronomy, The University of Manchester, Manchester M13 9PL, UK*
[11]*Institute of Cosmology and Gravitation, University of Portsmouth, Portsmouth PO1 3FX, UK*

Accepted 2021 May 20. Received 2021 April 30; in original form 2021 March 5

ABSTRACT

The advent of wide-field sky surveys has led to the growth of transient and variable source discoveries. The data deluge produced by these surveys has necessitated the use of machine learning (ML) and deep learning (DL) algorithms to sift through the vast incoming data stream. A problem that arises in real-world applications of learning algorithms for classification is imbalanced data, where a class of objects within the data is underrepresented, leading to a bias for overrepresented classes in the ML and DL classifiers. We present a recurrent neural network (RNN) classifier that takes in photometric time-series data and additional contextual information (such as distance to nearby galaxies and on-sky position) to produce real-time classification of objects observed by the Gravitational-wave Optical Transient Observer, and use an algorithm-level approach for handling imbalance with a focal loss function. The classifier is able to achieve an Area Under the Curve (AUC) score of 0.972 when using all available photometric observations to classify variable stars, supernovae, and active galactic nuclei. The RNN architecture allows us to classify incomplete light curves, and measure how performance improves as more observations are included. We also investigate the role that contextual information plays in producing reliable object classification.

Key words: methods: data analysis – techniques: photometric – survey.

1 INTRODUCTION

Time domain astronomy concerns the study of astronomical objects that exhibit variability in brightness over short time-scales, ranging from seconds to months, compared to typical cosmic time-scales that span millions to billions of years. The study of transient and variable sources allows us to gain a better understanding of the Universe, by way of measuring cosmic distances and cosmic expansion. The discovery of the period–luminosity relation of Cepheid variables in the Small Magellanic Cloud enabled the use of Cepheid variables as a standard candle to measure extragalactic distances (Leavitt & Pickering 1912). Type Ia supernovae (SNe) have been used to trace the expansion history of the Universe, and two teams independently measured the luminosity–distance relationship of SNe Ia up to redshift $z \sim 1$ to determine that the Universe is undergoing accelerated expansion (Riess et al. 1998; Perlmutter et al. 1999).

Historically, the discovery of transient and variable sources has been serendipitous. The advent of repeated sky surveys such as the Catalina Real-Time Survey (CRTS; Drake et al. 2009), the Panoramic Survey Telescope and Rapid Response System (Pan-STARRS; Kaiser et al. 2010), the Palomar Transient Factory (PTF; Rau et al. 2009), the All Sky Automated Survey for SuperNovae (ASAS-SN; Shappee et al. 2014), and the Subaru Hyper Suprime-Cam transient

★ E-mail: ufburhanudin1@sheffield.ac.uk

© 2021 The Author(s)
Published by Oxford University Press on behalf of Royal Astronomical Society

Downloaded from https://academic.oup.com/mnras/article/505/3/4345/6287587 by University of Leicester user on 02 September 2021

survey (HSC; Yasuda et al. 2019) has greatly increased the rate at which new transient and variable sources are discovered. The large quantity of data produced by these surveys has led to the discovery of new types of transients such as superluminous SNe (Quimby et al. 2007), and calcium-rich transients (Kasliwal et al. 2012).

Recently, the growing number of surveys across the world has facilitated a new era of multimessenger astronomy. Observing a single event across multiple wavelengths and through different detectors allows for a deeper understanding of the physics behind transient phenomena. In 2017, the gravitational wave signals of a binary neutron star merger (designated GW170817) were detected by the Advanced LIGO and Advanced Virgo gravitational-wave detectors (Abbott et al. 2017a). Follow-up observations across the electromagnetic spectrum led to the discovery of the kilonova AT2017gfo, thought to be powered by the radioactive decay of *r*-process nuclei following a binary neutron star merger (Abbott et al. 2017b; Chornock et al. 2017; Coulter et al. 2017; Drout et al. 2017; Shappee et al. 2017; Smartt et al. 2017; Villar et al. 2017).

Current surveys such as the Zwicky Transient Factory (ZTF; Bellm et al. 2018) and the upcoming Vera C. Rubin Observatory Legacy Survey of Space and Time (LSST; Ivezić et al. 2008) will generate large amounts of observational data. On a typical night of observing, ZTF will produce up to ∼1 TB of raw image data, and up to 2 million source alerts extracted from difference imaging (Masci et al. 2018). The rate of data generation exceeds human capability for manual processing, and the role of identifying and classifying new sources from survey data falls to machine learning (ML) algorithms (Ball & Brunner 2010; Bloom & Richards 2012; Bloom et al. 2012).

The first step in the automated process of discovery from image data is determining whether a source is astrophysically real, or if it is a 'bogus' detection such as an image artefact (e.g. as a result of poor subtraction from difference imaging) or a cosmic ray. The real-bogus classification problem has been well studied and applied to surveys, and is becoming a standard part of automated discovery pipelines (Brink et al. 2013; Wright et al. 2015; Gieseke et al. 2017; Lin et al. 2018; Duev et al. 2019; Mong et al. 2020; Killestein et al. 2021).

Once a source has been identified as real, the next step is classifying the source. Various taxonomies exist in astronomy, with broad categorizations such as consistently varying sources versus transient events, and classifications based on spectroscopic features for SN subtypes (Filippenko 1997). Usually, confidently classifying new optical transients relies on additional observations with spectroscopic facilities. However, in the era of large-scale sky surveys, spectroscopic follow-up of all new discoveries is not guaranteed and is prohibitively time consuming. As a result, alternative methods of classification relying on photometric and image data have been developed.

Both traditional feature-based ML and deep learning (DL) methods have been used to classify astronomical objects into distinct classes. Features are metrics derived from the data that are able to encapsulate the properties that differentiate between different classes. Light curve-derived features have been used to classify a set of simulated SN light curves from the Supernova Photometric Classification Challenge (Kessler et al. 2010) into different spectroscopic types (Lochner et al. 2016; Ishida & de Souza 2013), and also for SN classification with real survey data (Dauphin et al. 2020; Hosseinzadeh et al. 2020).

In contrast to feature-based ML models, DL models are able to learn salient features from the data, and do not require a feature extraction step prior to training. Deep neural network and recurrent neural network (RNN) architectures have been used to classify simulated light curves (Charnock & Moss 2017; Pasquet et al. 2019; Möller & de Boissière 2020) and real light curves (Takahashi et al. 2020) for SN classification, general transient classification (Muthukrishna et al. 2019), and variable star (VS) classification with real light curves (Mahabal et al. 2017; Tsang & Schultz 2019; Becker et al. 2020). Work has also been done on classifying objects using image stamps as input to convolutional neural networks (Carrasco-Davis et al. 2019; Gómez et al. 2020; Wardęga et al. 2020).

In search of the optical counterparts to gravitational wave signals is the Gravitational-wave Optical Transient Observer (GOTO; Steeghs et al., in preparation) survey. When GOTO receives an alert for gravitational wave or gamma-ray burst detection from other facilities, it will rapidly begin observing the localized region of sky to look for optical counterparts (Dyer et al. 2020).

While not in gravitational wave follow-up mode, GOTO conducts an all-sky survey to search for transient and variable sources. A real-bogus classifier (Killestein et al. 2021) first identifies astrophysically real detections and 'bogus' subtraction artefacts from difference images. Detections that are identified as real by the real-bogus classifier are then passed on to the GOTO Marshall (Lyman et al., in preparation), a web-based interface for GOTO observers to vet, search, and trigger follow-up observations of new discoveries. The next step in object classification is classifying real discoveries into distinct astrophysical types. Providing object classifications for real objects will be useful for the GOTO collaboration in helping to identify interesting targets for follow-up and further science goals.

Effective classification by ML and DL models relies on good representation of the labelled classes in the data set to learn class separability across the labelled objects. In real-world applications, the data will typically contain one or more classes that have more examples than other classes. This type of data is referred to as 'imbalanced data', and it poses a difficulty for classification as models will be biased towards the class where there are many more examples to learn from. Here, we present an RNN-based classifier for the classification of objects discovered by GOTO.

In Section 2, we provide an overview of the GOTO survey and the data used to train and test the RNN classifier. In Section 3, we introduce the RNN architecture, the class imbalance problem, and the approach taken to deal with an imbalance data set. In Section 4, we outline the training process, and in Section 5, we discuss the performance of the classifier, and how contextual information plays a role in how models learn to classify. We discuss how the work presented in this paper can be improved upon, and conclude with Sections 6 and 7.

2 THE GOTO SURVEY AND DATA

2.1 The GOTO

The GOTO is a ground-based observatory, with a modular design situated at the Roque de los Muchachos Observatory on La Palma, Canary Islands (Steeghs et al., in preparation). GOTO consists of multiple nodes, with each node hosting an array of up to eight 40 cm diameter unit telescopes (UTs) providing a combined 40 square degree field of view in a single pointing. The current configuration consists of a single node in La Palma, with a plan to add another node in La Palma and another two nodes at the Siding Spring Observatory in Australia. When fully complete, GOTO will have two nodes in La Palma (GOTO North) and two nodes in Australia (GOTO South), for a total of 4 nodes and 32 UTs and the ability to have constant coverage of the night sky. The primary science aim of GOTO is to search for optical signatures following a gravitational wave detection from detectors such as the Advanced LIGO (Abbott et al. 2009) and

Downloaded from https://academic.oup.com/mnras/article/505/3/4345/6287587 by University of Leicester user on 02 September 2021

Advanced Virgo (Acernese et al. 2015) facilities. When GOTO is not searching for gravitational wave counterparts, it conducts an all-sky survey enabling the discovery of new variables and, in particular, new transients.

Each GOTO UT is equipped with four Baader filters: a wide band L filter (covering 400–700 nm), and three R, G, and B filters. GOTO currently conducts all-sky surveys with the L filter, and is able to achieve a depth of up to 20.5 mag in a 60 s exposure under dark conditions. New transient and variable detections are obtained through difference imaging, where the difference between an image of a new source that is undergoing a change in brightness and a reference image at the same location is calculated to produce an image of the new source. The difference imaging process is done through an image processing pipeline (Steeghs et al., in preparation).

2.2 Data

The data set used to train the classifier is obtained from GOTO observations spanning the period of 2019 March 20 to 2020 November 4, during the GOTO prototype phase. Light curves of objects observed during this period are created using photometric measurements derived from difference imaging in the L filter. The catalogue of GOTO objects is then cross-matched to a number of external catalogue to determine objects that have also been observed by other telescopes and surveys, and to obtain classification labels. The list of external catalogues includes the following:

(1) The American Association of Variable Star Observers (AAVSO) International Variable Star Index (Watson, Henden & Price 2006).

(2) The Veron Catalog of Quasars and AGN, 13th edition (Véron-Cetty & Véron 2010).

(3) The Transient Name Server.[1]

Additionally, the GOTO objects are also cross-matched against the Galaxy List for the Advanced Detector Era (GLADE; Dálya et al. 2018) galaxy catalogue to identify if there is a nearby galaxy associated with the object. In total, the data set comprises 99 201 labelled objects, and is split into three broad classes: variable stars (VSs), active galactic nuclei (AGNs), and SNe. The data set is heavily imbalanced with 99 per cent (98 457) of labelled objects belonging to the VS class, and only 543 and 201 belonging to the AGN and SN classes, respectively. The largest class (VS) contains almost 500 times more examples than the smallest class (SN).

Within the astronomical taxonomy for transient and variable sources, there is a wide range of classification schemes: classifying VSs by the physical mechanism that causes variability (eclipsing binaries, RR Lyrae stars, and Cepheids) and classifying SNe by spectroscopic features (type Ia, type Ib/c, and type II). The use of 'superclasses' that group together distinct types of objects simplifies the classification task while still providing clear classifications. For rapidly evolving objects that would benefit from early-time follow-up observations such as SNe, it is beneficial to separate it from objects that show photometric variation over longer time-scales with just a few observations. Providing a more general classification in real time acts as a 'first-pass' classification, and further classification into more specific subtypes can be done when additional observations become available.

Fig. 1 shows the distribution of mean magnitudes and standard deviation in magnitudes, where each statistic is calculated from all measurements in a light curve. VSs have a broad range of magnitudes, while SN and AGN tend to be fainter compared to VS. Fig. 2 summarizes the light-curve properties within the data set. SNe are typically observed no longer than 100 d, while VS and AGN are observed over longer time-scales (>300 d). The median number of points in a light curve for the data set is 6, and the median time between successive observations is ∼11 d. The data set contains light curves ranging in duration from a single day to a few hundred days. The distribution of number of observations in a light curve and the time between successive observations within a light curve over all classes are fairly similar.

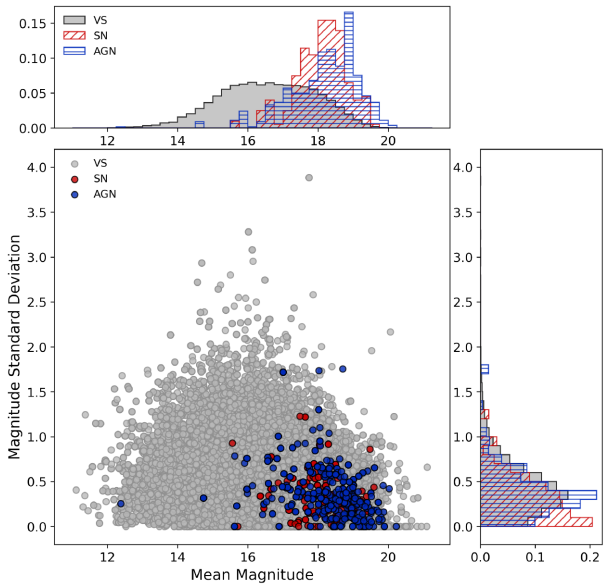


Figure 1. A scatter plot of the mean magnitude of light curves against the standard deviation in magnitudes of light curves for VSs in grey, SNe in red, and AGNs in blue. In the top and left-hand panels are histograms showing the distribution of mean magnitudes (top) and standard deviation in magnitudes (left). The histograms are plotted as normalized counts, for each class.

2.3 Data pre-processing

Before being used as input into the classifier, the data need to be pre-processed. For each light curve, the time-series input matrix consists of the times of observation $\mathbf{t}$, the magnitudes $\mathbf{m}$, and the errors in magnitude $\boldsymbol{\sigma}_m$. The time is scaled to the time of the first observation, so that it starts at zero, and subsequent time-steps are times since the first observation. The time-series input matrix $\mathbf{X_T}$ for a light curve with n observations is

$$\mathbf{X_T} = \begin{bmatrix} \mathbf{t} \\ \mathbf{m} \\ \boldsymbol{\sigma}_m \end{bmatrix} = \begin{bmatrix} t_0 & t_1 & \dots & t_n \\ m_0 & m_1 & \dots & m_n \\ \sigma_{m_0} & \sigma_{m_1} & \dots & \sigma_{m_n} \end{bmatrix}. \quad (1)$$

Light curves with more than 30 observations are truncated, and light curves with fewer than 30 observations are padded, so that all time-series input matrices $\mathbf{X_T}$ are matrices with 3 rows and 30 columns, where each column represents a time-step. Padding is required since the input matrices to the classifier need to have a fixed input size, but a masking layer can be used to tell the classifier to ignore padded time-steps (Section 3.2). We choose 30 observations since we are interested in being able to classify objects early in their light-curve evolution.

[1] https://wis-tns.weizmann.ac.il/

Downloaded from https://academic.oup.com/mnras/article/505/3/4345/6287587 by University of Leicester user on 02 September 2021

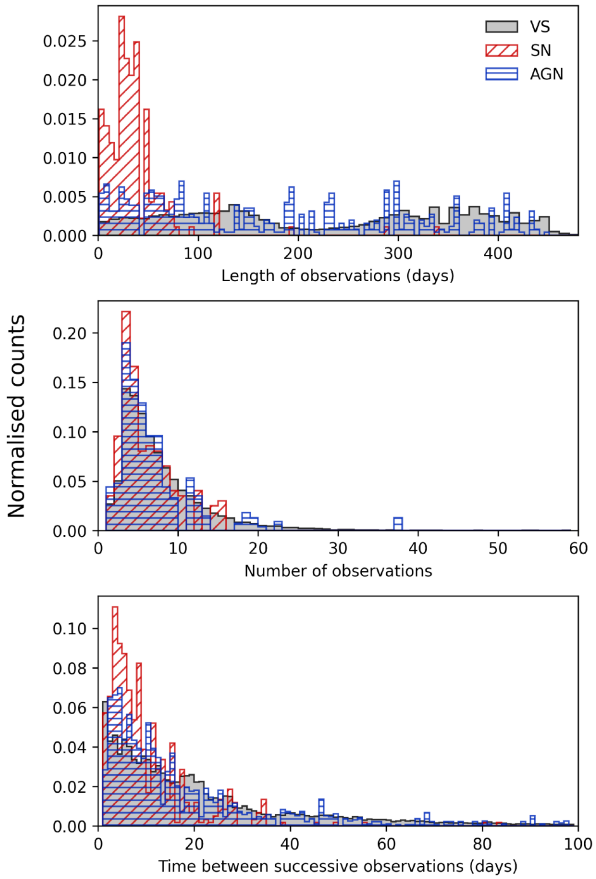


Figure 2. Normalized histograms showing different properties of the light curves in the data set, separated by class. From top to bottom: the time between the first and last observations of the light curves (length of observation), the number of observations in the light curves, and the time between successive observations over all light curves.

For each object, the contextual information used is the galactic longitude l and latitude b in degrees, and distance in arcseconds to the nearest galaxy in the GLADE catalogue $d_{\rm G}$. Objects that have $d_{\rm G} > 60$ arcsec have their $d_{\rm G}$ set to a dummy value. The contextual information input vector $\mathbf{X_C} = (l, b, d_{\rm G})$ is used alongside the time-series input matrix $\mathbf{X_T}$ as inputs for the classifier. Fig. 3 shows the distribution of $d_{\rm G}$ and b for all objects. AGN and SN are more commonly found to have a nearby galaxy in the GLADE catalogue than VS. VSs are mostly located close to the Galactic plane, while AGN and SN are usually found $\gtrsim 10^{\circ}$ away from the Galactic plane.

3 MODEL

3.1 RNNs

RNNs are a class of neural networks that operate on sequential data. The sequential data can take the form $[\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\tau]$, where $\boldsymbol{x}_t$ is a vector with time-step index t running from $t = 0$ to τ for a sequence with τ time-steps. The time-step index t does not necessarily have to represent the passage of time, but can also denote the position of a vector in the sequence. RNNs make use of 'hidden states' (similar to the hidden layers of a deep neural network) that incorporate information from the hidden state at the previous time-step.

The traditional RNN learns by minimizing a loss function (loss functions are explained in detail in Section 3.3) through gradient optimization, in a process known as backpropagation through time (Werbos 1990) – this is the analogue for backpropagation in fixed input size neural networks but applied to sequential data. Traditional RNNs struggle to encapsulate long-term time dependences because gradients propagated over many time-steps tend to explode or vanish – this is known as the *vanishing and exploding gradient problem*. Vanishing gradients make it difficult to improve the cost function because the incremental steps needed to find a local minimum become infinitesimal, and exploding gradients can make training unstable since a local minimum may never be found. Practical applications in processing sequential data make use of gated RNNs, which can control the flow of information through time. For an in-depth discussion on RNNs, refer to Goodfellow, Bengio & Courville (2016).

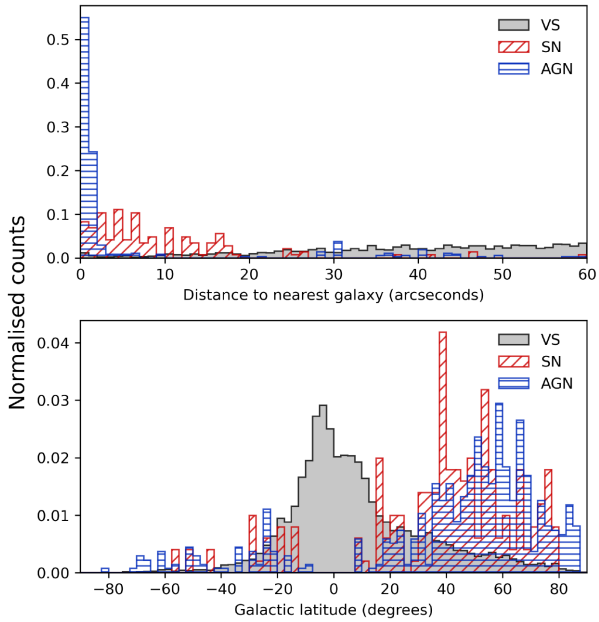


Figure 3. Normalized histograms showing how the distance to the nearest galaxy in the GLADE catalogue and galactic latitude for all objects in the data set varies by class.

There are two widely used gated RNN models: the long short-term memory (LSTM) and gated recurrent unit (GRU) networks. LSTM networks (Hochreiter & Schmidhuber 1997) use 'LSTM' cells that behave similar to hidden states in vanilla RNNs, but have additional parameters and a series of gates that control the flow of inputs and outputs within the cell, and the amount of information from previous time-steps to retain. The cell parameters are trainable, so the LSTM network also learns how to control the flow of information through time. An alternative gated RNN is the GRU (Cho et al. 2014), which uses a single gate rather than a set of multiple gates to control how information flows within the unit.

In this work, both LSTM and GRU networks are used to process light curve data and produce a set of class probabilities. It should be noted that the values returned by the classifier are not automatically true probabilities, rather they are scores given to an object by the classifier that indicate the level of 'belongingness' to a certain class. In this work, the scores returned by the RNN classifiers are referred to as the class or prediction probabilities. The outputs are produced after reading in the entire light curve, along with additional contextual information.

Downloaded from https://academic.oup.com/mnras/article/505/3/4345/6287587 by University of Leicester user on 02 September 2021

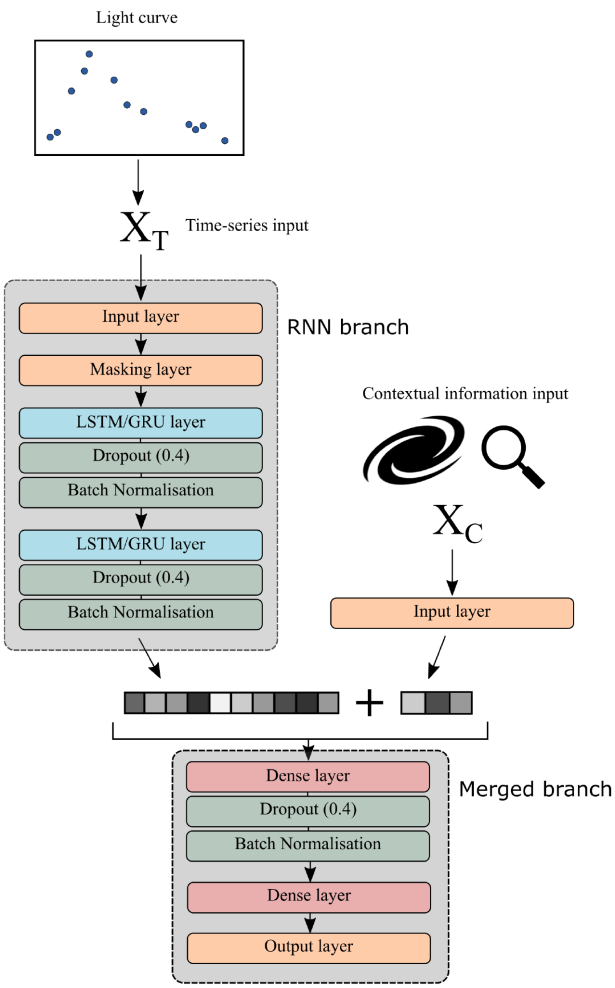


Figure 4. Diagram of the mixed input network. The time-series input $\mathbf{X_T}$ is passed to the RNN branch, and the contextual information input $\mathbf{X_C}$ is appended to the output of the final RNN layer, before being passed on to merged branch of the network.

3.2 Mixed input network

In order to utilize both the time-series data from the light curve and contextual information, the neural network model is formed of two branches: The first branch reads in the time-series input matrix $\mathbf{X_T}$ and the second branch reads in the contextual information input vector $\mathbf{X_C}$. Since RNNs are optimized for processing sequential data, the contextual information is fed to the model separately as opposed to together with the time-series data (e.g. as a vector of constant values), so that the RNN can extract high-level time-series features from the light curve. We use a similar approach to Muthukrishna et al. (2019) to construct the RNN branch of the network.

The first branch acts as a standard RNN, consisting of two RNN layers, and the second branch is just an input layer for $\mathbf{X_C}$. The output of the final layer in RNN branch is concatenated with $\mathbf{X_C}$, and forms the input for two dense layers, the latter of which is connected to the final output layer. The output layer produces a list of numbers that sum to 1, which are the class probabilities of an object belonging to each of the defined classes. Fig. 4 illustrates the model architecture. To avoid overfitting (where a model performs well on the training set but underperforms on the test set), dropout and regularization are used within the model. A summary of the different components of the model is presented below.

(i) *Masking layer*: The masking layer is placed between the input layer of the time-series branch of the model and the first RNN layer. It applies a mask to a sequence of time-steps, where each time-step refers to a column in the $\mathbf{X_T}$, and uses a mask value to skip time-steps. Subsequent layers will ignore masked time-steps. A masking layer allows the model to process sequences with a different number of time-steps.

(ii) *LSTM layer*: The LSTM layer takes in the masked $\mathbf{X_T}$ as input, and applies the LSTM operation. The dimension of the output $n_{\rm out}$ from this layer is an adjustable hyperparameter. Two LSTM layers are stacked in the RNN branch of the model: The first LSTM layer returns a sequence of outputs each with dimension $n_{\rm out}$, which is passed on to the second LSTM layer that returns a single output with dimension $n_{\rm out}$.

(iii) *GRU layer*: The GRU layer functions in the same way as the LSTM layer and has the same adjustable hyperparameter $n_{\rm out}$, but it applies the GRU operation to the data instead. Like the LSTM layers, two GRU layers are stacked and the first GRU layer passes a sequence to the second GRU layer, which returns a single output with dimension $n_{\rm out}$.

(iv) *Dropout*: A dropout layer randomly drops input neurons and their corresponding connections during training, as a method to reduce overfitting (Srivastava et al. 2014). This forces the neurons to derive more meaningful features from the data without heavily relying on other neurons in the network. During testing, the data are passed through the network without dropout. The dropout fraction sets the fraction of input neurons that are dropped.

(v) *Batch normalization*: During training, the parameters for each layer change, affecting the distribution of the inputs in the proceeding layers in the network. The change in the distribution of inputs in the layers requires the layers to adapt to the new distribution, a phenomenon known as internal covariate shift. Batch normalization scales the inputs to a layer for each batch so that mean value is close to 0 and the standard deviation is close to 1, reducing the impact of internal covariate shift (Ioffe & Szegedy 2015).

(vi) *Dense layer*: A dense layer is the simplest layer in a neural network: It consists of a fully connected layer of neurons and takes in a fixed size input. The number of neurons in the dense layer is an adjustable hyperparameter. All dense layers in the merged branch have the same number of neurons, which is set by the hyperparameter. The output layer is just a dense layer with three neurons with a softmax activation function that ensures the values returned by the output layer all sum to 1.

(vii) *Regularization*: Regularization introduces a penalty term to the loss function as a method to reduce overfitting. The L2 regularization is used, which adds a penalty term equal to the sum of all the model weights squared, multiplied by a regularization factor λ. The regularization factor λ sets the strength of 'weight decay' in the loss function. A larger value of λ forces the weights to have smaller values and helps to reduce overfitting (Goodfellow et al. 2016).

The LSTM and GRU layers serve the purpose of extracting meaningful features from the time-series data. The dense layers then combine the time-series representations and contextual information and further extract features from the combined features to produce a prediction. After each LSTM/GRU layer and the first dense layer, dropout is applied in an attempt to reduce overfitting, and batch normalization is applied to reduce internal covariate shift throughout the network. We use a combination of three methods as an approach to deal with potential overfitting: dropout, batch normalization, and regularization in the loss function. It is possible to further investigate how these methods work with or without each other and the impact they have on reducing overfitting, but it is beyond the scope of this work.

Downloaded from https://academic.oup.com/mnras/article/505/3/4345/6287587 by University of Leicester user on 02 September 2021

3.3 Class imbalance

In classification problems, class imbalance occurs when one class contains significantly fewer examples compared to other classes. The class with fewer examples is often referred to as the minority class, and the classes with many examples are referred to as the majority class. Extreme imbalance can occur when the minority class contains significantly fewer examples than the majority class. Learning from imbalanced data can be difficult, since conventional ML and DL algorithms assume an even distribution of classes within the data set. Classifiers will tend to misclassify examples from the minority class, and will be optimized to perform well on classifying examples from the majority class.

There are three main approaches for dealing with class imbalance (Krawczyk 2016):

(1) *Data-level methods*: reduce imbalance by modifying the data with resampling methods.
(2) *Algorithm-level methods*: modify the algorithm to reduce bias towards examples from the majority class.
(3) *Hybrid methods*: combine both data resampling and algorithm-level methods.

The problem of class imbalance within astronomy has been addressed in a number of works dealing with light-curve classification. Synthetic Minority Oversampling Technique (Chawla et al. 2002) and a Gaussian resampling variant have been used to augment a training set of spectroscopically classified SNe from the Pan-STARRS1 Medium-Deep Survey (Villar et al. 2019). Gaussian processes have been used to augment a set of simulated LSST light curves from the PLAsTiCC data set (The PLAsTiCC Team et al. 2018) to generate a more representative training set (Boone 2019). Hosenie et al. (2020) combine data resampling methods and a hierarchical classification system to deal with class imbalance for VS classification.

All the above methods use feature-based ML algorithms to classify light curves. Despite the popularity of DL-based classifiers, there is a lack of research in dealing with class imbalance when using DL architectures (Johnson & Khoshgoftaar 2019). Data augmentation for light curves of different astrophysical objects can be a laborious process, as models are needed to generate simulated examples of real observations, and multiple models may be required to simulate objects from multiple classes. This work attempts an algorithm-level approach for dealing with class imbalance by using a focal loss function to optimize the RNN classifier, as an alternative to a data-level approach such as data augmentation.

In supervised learning, a model is trained for a prediction task by optimizing an objective function, commonly referred to as the loss function. A loss function measures the error between the model output and the target output. The model adjusts its internal weights to reduce the error by calculating the gradient in weight space that will minimize the loss function value (Lecun, Bengio & Hinton 2015). This process is referred to as gradient descent.

For multiclass classification, the cross-entropy loss is typically used. Given a multiclass problem with N classes, the cross-entropy loss (CE) for an example i is

$$\mathrm{CE} = -\sum_{j=1}^{N} \delta_{ij} \log(p_{ij}), \tag{2}$$

where p_{ij} is the probability of example i belonging to class j, and δ_{ij} is the Kronecker delta function. The loss for the entire data set is given by summing the loss of all examples.

The focal loss (Lin et al. 2017) addresses class imbalance by down-weighting examples from the majority class. For the same multiclass problem as above, the focal loss (FL) for an example i is

$$\mathrm{FL} = -\sum_{j=1}^{N} \delta_{ij}(1 - p_{ij})^{\gamma} \log(p_{ij}), \tag{3}$$

where $(1 - p_{ij})$ is the modulating factor, and γ is the parameter that adjusts the rate at which majority class examples are down-weighted. Increasing γ reduces the contribution from well-classified examples to the loss, and increases the importance of improving misclassified examples (Lin et al. 2017). For a misclassified example, p_{ij} is small, so the modulating factor is close to 1 and the loss is unaffected. Examples that are well classified will have p_{ij} close to 1, so the modulating factor is small, and the loss from well-classified examples will be down-weighted. The focal loss is equivalent to the cross-entropy loss when $\gamma = 0$. In practice, a weighted version of the focal loss can be used, which Lin et al. (2017) find to perform better than the unweighted focal loss (equation 3) for imbalanced classification tasks:

$$\mathrm{FL} = -\sum_{j=1}^{N} \delta_{ij}\alpha_j(1 - p_{ij})^{\gamma} \log(p_{ij}), \tag{4}$$

where α_j is a weighting factor for class j. In this work, the weighting factor is given by

$$\alpha_j = \frac{1}{n} \times \frac{N}{N_j}, \tag{5}$$

where N is the total number of examples in the training set, N_j is the number of examples in class j in the training set, and n is the total number of classes, which in this case is $n = 3$. For all the above loss functions, the best case is a loss value of zero.

4 METHOD

The GOTO data set is split into 70 per cent for training, and 30 per cent for testing. Of the training set, 30 per cent is set aside for validation. The data are split so that the training, validation, and test sets all have the same proportions of VS, AGN, and SN. A validation set is useful as it provides a measure of how well a model is able to generalize during training. The RNN is implemented and trained using the TENSORFLOW 2.0 package for PYTHON (Abadi et al. 2016)[2] with KERAS (Chollet et al. 2021) for implementation of network layers.

4.1 Classification metrics

For a binary classification task where one class is positive and the other class is negative, there are four possible outcomes when evaluating the performance of a classifier. If an example from the positive class is classified as positive, then the result is a *true positive*; if it is classified as negative then the result is a *false negative*. If an example from the negative class is classified as negative, the result is a *true negative*; if it is classified as positive then the result is a *false positive*.

Fig. 5 shows a confusion matrix that represents the possible outcomes from a classifier in a binary classification task. There

[2]https://www.tensorflow.org/

Downloaded from https://academic.oup.com/mnras/article/505/3/4345/6287587 by University of Leicester user on 02 September 2021

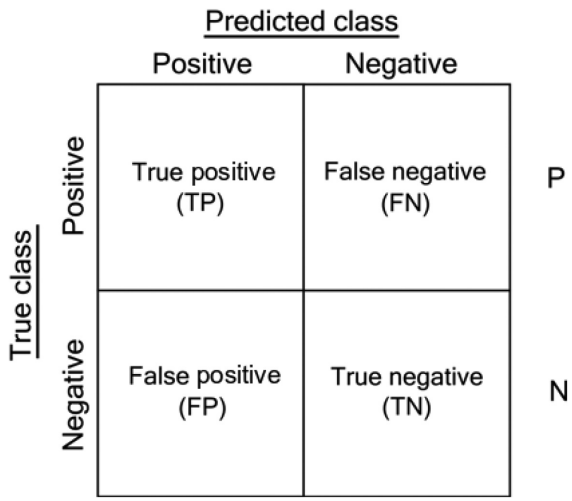


Figure 5. A confusion matrix representing the possible outcomes in binary classification for a positive and negative class. Class labels along the horizontal axis are the labels predicted by the classifier, and class labels along the vertical axis are the true labels. The total number of all true positives is denoted by P and the total number of all true negatives is denoted by N.

are a number of commonly used classification metrics that can be calculated from the confusion matrix. The accuracy of a classifier is

$$\text{Accuracy} = \frac{TP + TN}{P + N}. \tag{6}$$

The F_1 score of a classifier is

$$F_1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)}. \tag{7}$$

The true positive rate (TPR), or the recall, is

$$\text{TPR} = \frac{\text{Correctly classified positives}}{\text{Total positives}} = \frac{TP}{TP + FN} = \frac{TP}{P}. \tag{8}$$

The false positive rate (FPR) or the false alarm rate is

$$\text{FPR} = \frac{\text{Incorrectly classified negatives}}{\text{Total negatives}} = \frac{FP}{FP + TN} = \frac{FP}{N}. \tag{9}$$

When dealing with imbalanced data, metrics such as accuracy and F_1 score are sensitive to the class distribution in the data set. For example, a classifier predicting on a test set that contains 99 per cent positive examples and 1 per cent negative examples can achieve an accuracy of 99 per cent by predicting all examples as positive. We consider the confusion matrix in Fig. 5. The proportion of positive to negative examples is a relationship between the top row (positive) and the bottom row (negative). Metrics that are calculated using values from both rows (i.e. accuracy and F_1 score) will be sensitive to class imbalance. Alternative metrics for classification are receiver operating characteristics (ROC) graphs and the area under the ROC graph (Area Under the Curve, AUC). The ROC is based on the TPR and FPR, where each metric is a ratio calculated from values along a row of the confusion matrix, and hence is insensitive to class imbalance. ROC graphs plot the TPR on the y-axis and the FPR on the x-axis.

An ROC graph shows the trade-off between true positives and false positives – an ideal classifier will return all true positives and no false positives. A classifier that returns discrete classifications (i.e. one that predicts the class labels) will produce a single pair of values for TPR and FPR, and a corresponding point in ROC space. A classifier that returns probabilities can produce a range of TPR and FPR values by varying a threshold: If the probability is above the threshold, then the classifier returns a positive classification, and if it is below the threshold then the classifier returns a negative classification. Since different threshold values produce different TPR and FPR values, it is possible to plot a curve in ROC space by varying the threshold.

In order to compare different classifiers, it is convenient to have a single score that represents classifier performance. Since the ROC curve is in two-dimensional space, the AUC is a fraction of the total area of ROC space, and will always have a value between 0 and 1. For an in-depth discussion of ROC analysis, see Fawcett (2006).

These metrics can be extended to a multiclass classification problem: The confusion matrix for n classes becomes an $n \times n$ matrix with the diagonal entries representing correct classifications and the off-diagonal entries representing incorrect classifications. A method for plotting ROC graphs for multiple classes is to plot ROC graphs for each class, treating one class as the positive, and all others as negative. A formulation of the AUC for multiclass classification that is insensitive to imbalanced data was derived by Hand & Till (2001), which calculates the unweighted mean of the pairwise AUC over all classes. Although this formulation of the AUC is insensitive to class imbalance, it is not straightforward to visualize the ROC space with this method. Nevertheless, the pairwise AUC metric is useful to evaluate the performance of multiple classification models during hyperparameter optimization.

Table 1. Adjustable hyperparameters in the model. Hyperparameters with values in square brackets indicate the range of values used during training.

Hyperparameter	Value
Batch size	128
Learning rate	1×10^{-4}
LSTM/GRU output dimension	[100, 150]
Dropout	0.4
Number of neurons in dense layers	[100, 200]
Focal loss γ	[1.0, 2.0]
Regularization factor λ	0.01

4.2 Hyperparameters

The model has a number of adjustable hyperparameters, parameters that are not derived through training, but are pre-defined before the training process. Table 1 lists the adjustable hyperparameters of the model. Two hyperparameters are varied during training for all models: the dimension of the LSTM/GRU output and the number of neurons in the dense layers. We select the values of the LSTM/GRU output dimension to be similar to those used in Muthukrishna et al. (2019). For models trained with the focal loss, there is another hyperparameter that is varied: the focal loss γ parameter. We limit the number of adjustable hyperparameters since we want to limit the total number of models to be trained – there are a total of 32 models with all possible hyperparameter combinations. We find that varying the hyperparameters during hyperparameter optimization does not hugely impact performance on the validation set, and the hyperparameters listed in Table 1 produce good classification results.

Hyperparameters that remain fixed are the batch size, the learning rate, the dropout fraction, and the regularization factor. Batch size sets the number of samples that is passed through the model at each epoch of training before the weights are updated. Given that the data are

Downloaded from https://academic.oup.com/mnras/article/505/3/4345/6287587 by University of Leicester user on 02 September 2021

Table 2. Results for the best performing models and their best hyperparameters with AUC and F_1 scores, evaluated on the validation and test sets. On the bottom row, GRU NC denotes the GRU model with weighted focal loss trained only on time-series data without contextual information.

RNN type	Loss	Dense layer neurons	RNN output dimension	γ	Validation		Test	
					AUC	F_1	AUC	F_1
LSTM	Weighted focal loss	200	150	1	0.958	0.469	0.966	0.486
GRU	Weighted focal loss	200	150	2	0.947	0.425	0.972	0.468
LSTM	Weighted cross-entropy	200	100	–	0.939	0.404	0.967	0.464
GRU	Weighted cross-entropy	200	150	–	0.932	0.378	0.968	0.442
LSTM	Unweighted cross-entropy	200	150	–	0.899	0.727	0.948	0.794
GRU	Unweighted cross-entropy	200	150	–	0.909	0.758	0.937	0.806
GRU NC	Weighted focal loss	100	100	1	0.922	0.322	0.909	0.324

imbalanced, the batch size is set to 128 to ensure that examples from the minority class are passed through the model while the weights are being updated. We note that this is a rudimentary method, and a more thorough approach would be to employ a 'stratified' batching, where each batch has the same proportion of minority and majority examples. Nevertheless, we find that simply setting the batch size to 128 is sufficient to achieve good results (Section 5). The learning rate defines the step size taken during gradient descent to determine the optimal set of weights; in other words, it controls how much the weights are changed during training. A learning rate that is too small will fail to find a local minimum in weight space, and a learning rate too large will result in unstable training and fail to find a local minimum. During training, the learning rate is left to the default TENSORFLOW value of 1×10^{-4}.

Srivastava et al. (2014) find that setting the probability of retaining a unit in the network between 0.4 and 0.8 optimally reduces test error on a classification task with a deep neural network. In this work, the dropout fraction used in the TENSORFLOW implementation is defined as the fraction of units that are dropped, so the optimal range found by Srivastava et al. (2014) translates to 0.2–0.6 when expressed as the fraction of units to be dropped. We opt for a dropout fraction of 0.4 in this work. The regularization factor λ is set to the default TENSORFLOW value 0.01.

The dropout fraction and regularization factor λ could have been used as additional hyperparameters to see if varying their values would impact classification performance, but we choose to keep these values constant as to minimize the total number possible model configurations that need to be trained. The main motivation for this work is to see how using weighted loss functions affects performance on imbalanced data.

4.3 Training process

There are two different RNN architectures used, one with LSTM and the other with GRU, and three different loss functions: an unweighted cross-entropy loss (equation 2), a weighted cross-entropy loss, and a weighted focal loss (equation 4). In total, there are six classes of models, each trained with a range of hyperparameters (Table 1). For models trained with the cross-entropy loss functions, the focal loss parameter γ is not an adjustable parameter.

All models are trained for 200 epochs with no early stopping using the Adam optimizer (Kingma & Ba 2014), and then evaluated on the validation set. The best performing model is selected by choosing the model that has the best AUC score calculated on the validation set. The best models from the six different configurations are then evaluated on the test set. Training was executed on an eight-core CPU. The average time taken to train a single model was approximately 1 h at an average of 20 s per epoch, and the total time taken to train all models presented in this paper was ∼44 h. The trained models are able to return predictions on the entire test set within ∼5 s.

5 RESULTS

5.1 Hyperparameter optimization

The models are trained with all possible hyperparameter combinations presented in Table 1. After training, all models are evaluated on the validation set. The best set of hyperparameters for the six different model configurations is summarized in Table 2, with both AUC and the F_1 score shown for each model. All models converge within 200 epochs; Fig. A1 illustrates how the loss on the training and validation sets evolves during training.

The weighted cross-entropy models show a small increase of ∼0.03 in AUC over the unweighted cross-entropy models, and the weighted focal loss shows an additional increase of ∼0.05 in AUC. In this work, we used the unweighted F_1 score, which calculates the F_1 score for each class and takes the unweighted mean. Although the unweighted cross-entropy loss models both have higher F_1 scores, the models with weighted loss functions perform better overall across all classes – the F_1 score is skewed towards the class with more examples and is not a metric well suited for imbalanced data. For the weighted focal loss models, the LSTM appears to perform slightly better than the GRU on the validation set.

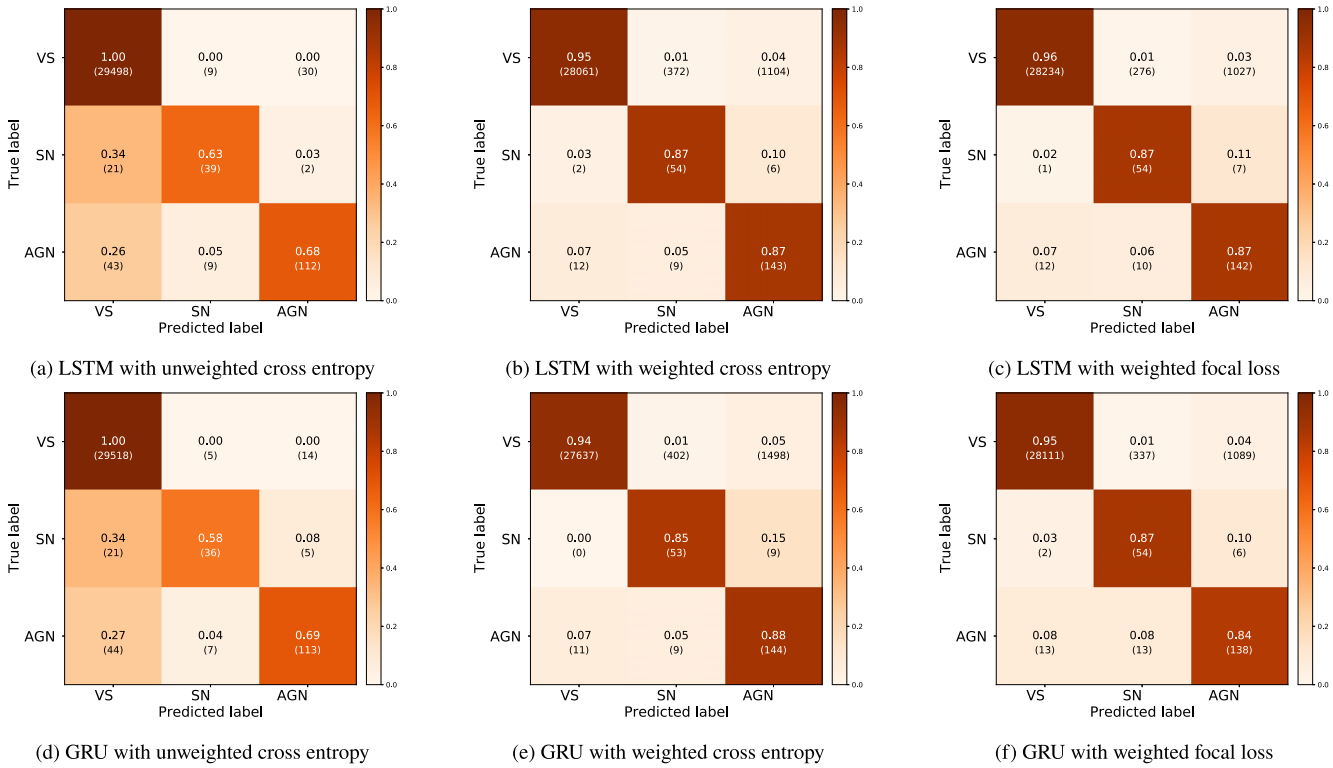
5.2 Test set performance

After the best hyperparameters are determined for all the models, the models are then evaluated on the test set. The test set consists of data that the classifier has not seen during the training phase. Columns 8 and 9 of Table 2 summarize the performance of the six different model configurations on the test set, using all available observations.

As in the results for the validation set, the performance of the models on the test set indicates that simply adding weights to the loss function to account for class imbalance improves performance. Both the weighted cross-entropy and weighted focal loss models show an increase of ∼0.02 in AUC over the unweighted cross-entropy. The F_1 scores are also shown for illustrative purposes; although the models with unweighted cross-entropy loss have the highest F_1 scores, they have the lowest AUC scores. To better clarify the improvement of weighted loss models over the unweighted cross-entropy models, it is prudent to also look at the confusion matrices for each of these models.

Fig. 6 shows the confusion matrices for all the models evaluated on the test set. Confusion matrices are useful evaluation tools for multiclass classification problems; they visualize how often a

Downloaded from https://academic.oup.com/mnras/article/505/3/4345/6287587 by University of Leicester user on 02 September 2021



(a) LSTM with unweighted cross entropy (b) LSTM with weighted cross entropy (c) LSTM with weighted focal loss

(d) GRU with unweighted cross entropy (e) GRU with weighted cross entropy (f) GRU with weighted focal loss

Figure 6. Confusion matrices for the best performing models on test data. The labels on the *x*-axis are the labels predicted by the classifier, and the labels on the *y*-axis are the true labels. Correct predictions are represented by values along the diagonal, and incorrect predictions are represented by values in the off-diagonal. The rows of the matrix show the fraction of correct and incorrect predictions for each class, and where incorrect predictions between classes occur. Below the fractions are the number of objects that have been predicted, in parentheses.

classifier makes correct predictions, and where misclassifications between classes occur (Fig. 5). The models with unweighted cross-entropy loss perform the worst, and the confusion matrices for these models show the impact of class imbalance on the model.

Figs 6(a) and (d) show that the unweighted cross-entropy loss models have complete recovery for VS, but also misclassify roughly a third of SN and AGN as VSs. The LSTM model correctly classifies 63 per cent of SN and 68 per cent of AGN, and the GRU model correctly classifies 58 per cent of SN and 69 per cent of AGN. There is relatively little confusion between SN and AGN for the LSTM and GRU models.

Using a weighted cross-entropy loss function improves performance on AGN and SNe, but slightly decreases performance for VS. Figs 6(b) and (e) show increased recovery for SN and AGN: up to 87 per cent and 85 per cent accuracy for SN with the LSTM and GRU models, respectively, and up to 87 per cent and 88 per cent accuracy for AGN with the LSTM and GRU models, respectively. The accuracy for VS dropped to 95 per cent for the LSTM model and 94 per cent for the GRU model, with misclassifications occurring in both SN and AGN. There is a small amount of confusion ($<$15 per cent) between SN and AGN for the LSTM and GRU models with weighted cross-entropy loss functions.

The models using weighted focal loss show similar performance to the weighted cross-entropy models, with minor improvements. Figs 6(c) and (f) show that both LSTM and GRU models with weighted focal loss are able to achieve 88 per cent accuracy for SN, up to 96 per cent accuracy for VS (with the GRU model achieving 95 per cent accuracy for VS), and up to 87 per cent accuracy for AGN (with the GRU model achieving 84 per cent accuracy for AGN). There is some degree of confusion between SN and AGN, but no more than 11 per cent of examples from these classes are misclassified as the other. Only up to 3 per cent of examples from SN are classified as VS, and up to 8 per cent of AGN are classified as VS. This is a significant improvement over the models with unweighted cross-entropy, where up to a third of objects from the minority classes (SN and AGN) are misclassified as the majority class (VS).

Looking at the number of predictions made for each entry in the confusion matrix for the GRU model trained with focal loss (Fig. 6f), it can be seen that the number of VSs predicted as SNe and AGNs is greater than the number of actual SNe and AGNs in the test set. The classifier is designed to produce prediction probabilities, and the predicted class is simply selected by choosing the class prediction that has the highest probability. We can examine how varying the threshold value for the class probability can refine the classification results. By breaking down the three-class problem into three binary classification problems, where for each case the positive class is one of VS, SN, or AGN, and the negative class is the other two, we can calculate the TPR and FPR for each case. The probability of a negative prediction is simply given as the sum of the probability of the two non-positive classes. We use the predictions given by the GRU model trained with focal loss since it has the highest AUC score on the test set, and compute the TPR and FPR., and produce an ROC curve for each class (Fig. 7). The ROC curves for all the classes reflect the high AUC score of the GRU model trained with weighted focal loss, covering most of the TPR–FPR space and reaching the top left-hand corner (high TPR at low FPR).

We can examine how varying the threshold can reduce contamination, that is, to reduce the number of false positives in each class. By selecting a 'cut-off' threshold, any objects that have a prediction probability below the threshold can be regarded as negative, and those

Downloaded from https://academic.oup.com/mnras/article/505/3/4345/6287587 by University of Leicester user on 02 September 2021

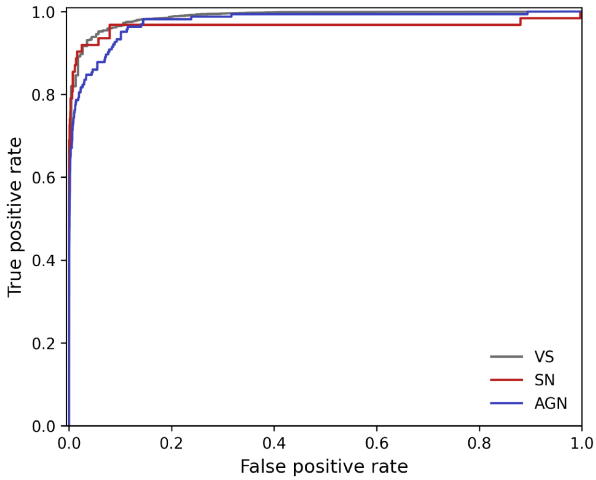


Figure 7. Receiver operating characteristic (ROC) curves for the VS, SN, and AGN classes (grey, red, and blue, respectively). ROC curves plot the TPR against the FPR for a range of threshold values that dictate whether an object is classified as positive or negative. The curve is obtained by considering a separate binary classification case for each class, treating one class as positive and the rest as negative.

Table 3. The TPR and FPR for each class, evaluated on the test set at a threshold for the GRU model trained with focal loss. The positive and negative predictions are obtained by treating each class as positive, and the other two as negative, creating a binary classification problem for each of the three classes. The numbers of true positive predictions and false positive predictions are shown in parentheses with the TPR and FPR values.

Class	Threshold	TPR	FPR
VS	0.7	86.2 per cent (25,465)	1.8 per cent (4)
SN	0.7	74.2 per cent (46)	0.2 per cent (57)
AGN	0.7	72.6 per cent (119)	0.7 per cent (213)

with a prediction probability above the threshold can be regarded as positive. Table 3 shows the TPR and FPR along with the number of true positive and false positive predictions for each class by using a threshold value of 0.7. By selecting a cut-off threshold, the number of false positive predictions for all classes is reduced. At a threshold value of 0.7, the TPR of SN and AGN is >70 per cent at an FPR of <1 per cent. The number of false positives for SN and AGN is still significant, and we note that reducing contamination on minority classes in an imbalanced data setting remains a challenge.

5.3 Time-dependent performance

With an RNN architecture, it is possible to take in sequential inputs of different lengths. Hence, it is possible to evaluate the classifiers performance by varying the number of light-curve observations used. To do this, the time-series input matrix can be formatted so that it contains only the first n observations, and the remaining values are padded. The models are then evaluated on the test data, using an increasing number of observations from $n = 1$ to a maximum of $n = 30$.

Fig. 8 shows how the AUC of all models varies as the number of observations included in the light curves is increased. In this case, the number of observations refers to the maximum number of observations that are included. If the maximum number of observations is m, then a light curve with fewer than m observations will have all its observations included. If a light curve has more than m observations, then only the first m observations are used.

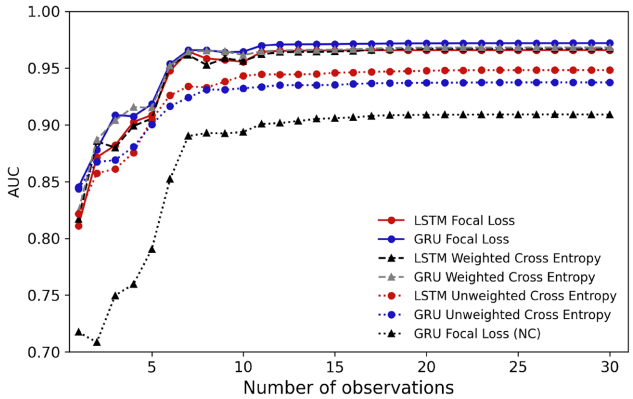


Figure 8. AUC scores evaluated for all models on the test data, plotted as a function of increasing number of light-curve observations included in the light curve. The black dotted line with triangular markers shows the AUC scores for the GRU model trained with weighted focal loss without contextual information [labelled GRU Focal Loss (NC)].

With just one light-curve observation, the unweighted cross-entropy models, weighted cross-entropy models, and the LSTM model with weighted focal loss achieve AUC scores of ∼0.82. The GRU model with weighted focal loss achieves the highest AUC score of 0.84 with a single light-curve observation. As more observations are included, all models show an increase in AUC until around 10 observations, after which the AUC scores maintain a constant value. Since a majority of the light curves in the data set only have up to 10 observations (see Fig. 2), it is not surprising that model performance remains constant after a maximum of 10 light-curve observations are included. The final values of the AUC for all models are shown in column 8 of Table 2.

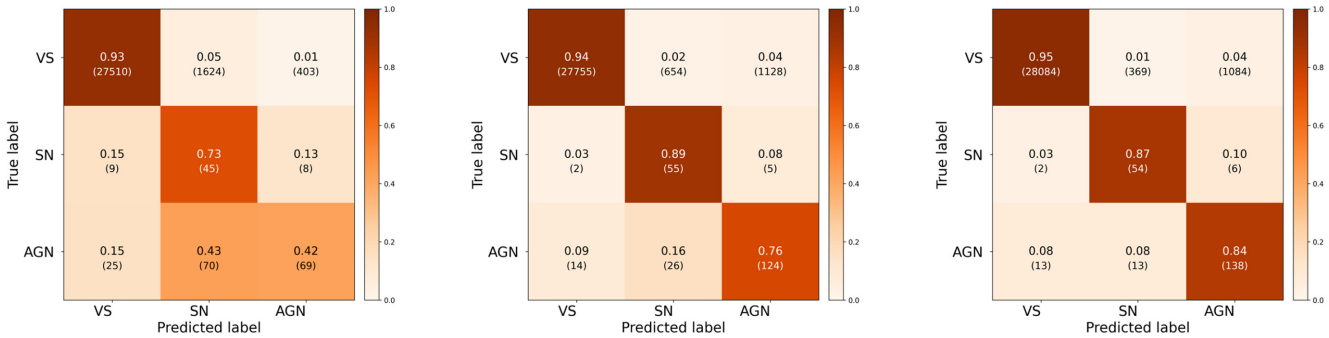
We perform some additional analysis on how the GRU model with weighted focal loss performs over time, choosing the aforementioned model since it has the highest AUC score on the test set. Fig. 9 shows the confusion matrices for the GRU model with weighted focal loss evaluated with different numbers of light-curve observations. With one observation, the model is already able to separate out VS from other objects to a good degree of accuracy, achieving 93 per cent accuracy for VS. At one epoch of observation, the model achieves 73 per cent accuracy for SN, and 42 per cent accuracy for AGN, with 43 per cent of AGNs being misclassified as SNe. Other than AGNs being misclassified as SNe, there is some degree of confusion between all classes: 13 per cent of SNe are misclassified as AGNs, and 15 per cent of SNe and AGNs are misclassified as VSs.

With up to six light-curve observations, the accuracy for SN and AGN improves. 89 per cent of SNe and 76 per cent of AGN are correctly classified, with some misclassifications between the two (<16 per cent), and few SNe and AGNs being misclassified as VSs. At a maximum of 20 observations, the model reaches the maximum performance. The confusion matrix for the GRU model with weighted focal loss at 20 epochs in Fig. 9(c) appears similar to the confusion matrix evaluated with all epochs of observations as in Fig. 6(f). The SN accuracy drops from 89 per cent at six observations to 87 per cent at 20 observations, suggesting some confusion as more observations are included and light curves appearing similar to each other at longer time-scales.

5.4 Importance of contextual information with t-SNE

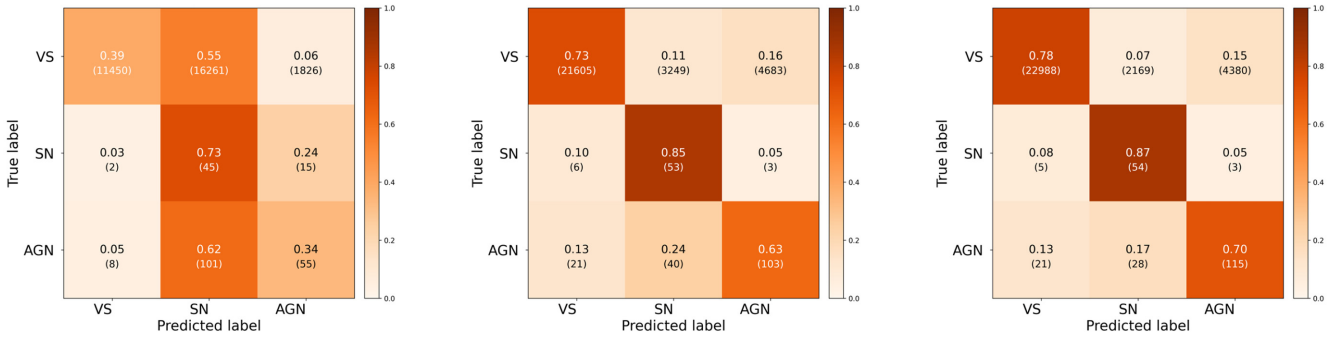
Imaging surveys such as GOTO will be able to provide contextual information for newly discovered objects (for example, cross-

Downloaded from https://academic.oup.com/mnras/article/505/3/4345/6287587 by University of Leicester user on 02 September 2021



(a) Maximum observations = 1 (b) Maximum observations = 6 (c) Maximum observations = 20

Figure 9. Confusion matrices for the GRU model with weighted focal loss, evaluated with an increasing number of light-curve observations.



(a) Maximum observations = 1 (b) Maximum observations = 6 (c) Maximum observations = 30

Figure 10. Confusion matrices for the GRU model with weighted focal loss trained only on time-series data, evaluated with an increasing number of light-curve observations.

matching to galaxy catalogues) in addition to photometric data. We now discuss the importance of contextual information for the model in learning to differentiate between objects from different classes. To investigate the impact of contextual information, we train a grid of GRU models with weighted focal loss using the hyperparameters in Table 1 on the same data as the other models, but without additional contextual information. The inputs to these models are just the time-series data from the light curves $\mathbf{X_T}$. We identify the best performing model by selecting the model that achieves the highest AUC score on validation set.

5.4.1 Performance without contextual information

The hyperparameters, AUC and F_1 scores of the best performing GRU model with weighted focal loss trained only on time-series data are shown in the bottom row of Table 2. The model achieves an AUC score of 0.922 on the validation set, which is higher than the validation AUC scores of the unweighted cross-entropy models.

We follow the same process as in Section 5.2 and evaluate this model on the test set. The model achieves an AUC score of 0.902 on the test set, which is the lowest AUC score out of all models. In Fig. 8, we plot how the AUC score evolves as more light-curve observations are included. At one observation, the model achieves an AUC score of 0.717 and then increases up until the six observations after which is starts to maintain a constant value.

Fig. 10 shows the confusion matrices evaluated on the test set using an increasing number of maximum observations. With just one light-curve observation, the model performs worse than the GRU model with weighted focal loss trained with contextual information. The model achieves an accuracy of 39 per cent for VS and incorrectly classifies 55 per cent of VSs as SNe, and 34 per cent accuracy for AGN and incorrectly classifies 62 per cent of AGNs as SNe. SN accuracy for the model is similar to the models trained with weighted focal loss, with an accuracy of 73 per cent. When using all available light-curve observations, the model trained only on time-series data achieves a similar accuracy for SN as the weighted focal loss models at 87 per cent, but lower accuracy for VS at 78 per cent and AGN at 70 per cent. There is a slightly higher degree of misclassification between classes compared to the models with weighted loss functions trained with contextual information.

Fig. 11 shows the ROC curve for the model trained only on time-series data (using all available light-curve observations), by separating the three-class problem into three separate binary classification problems. Compared to Fig. 7, the ROC curves show that the model does not perform as well when contextual information is excluded.

Overall, the model trained only with time-series data performs worse than its counterpart trained with contextual information, but is able to achieve comparable accuracy for SN. This suggests that the model is able to extract information from the light curves that allows for good separation of SN from the other classes. We expand on this in the following analysis.

5.4.2 t-distributed stochastic neighbour embedding representation

We use t-distributed stochastic neighbour embedding (t-SNE) to represent how the model transforms the input data at different layers of the network. t-SNE is a data visualization technique used to map a high-dimensional data set into a low-dimensional data set that

Downloaded from https://academic.oup.com/mnras/article/505/3/4345/6287587 by University of Leicester user on 02 September 2021

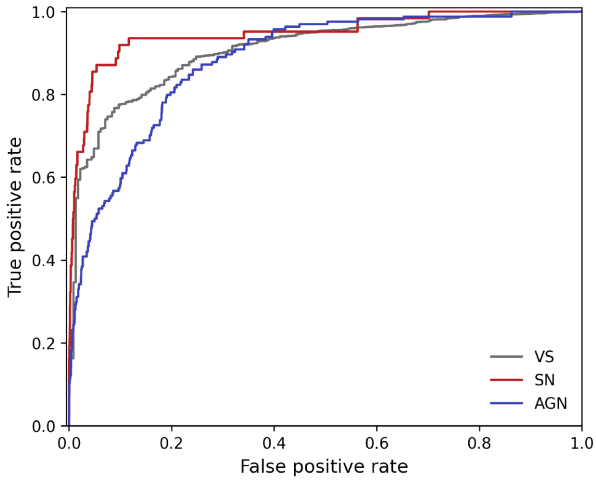


Figure 11. ROC curves for the VS, SN, and AGN classes (grey, red, and blue, respectively) for the GRU model with weighted focal loss, trained only on time-series data.

can be visualized in a two- or three-dimensional scatter plot (van der Maaten & Hinton 2008). We provide a summary of how t-SNE makes a low-dimensional visualization of a high-dimensional data set in the appendix.

t-SNE has been used to visualize class separability in SN classification with ML algorithms (Lochner et al. 2016). In this analysis, we use t-SNE to visualize how class separability changes at different layers in the network. We consider two models: the GRU model with weighted focal loss trained with contextual information and the same model trained with only time-series data.

We take the output of the model at two points: the output after the final GRU layer, where the model only considers time-series information from the light curve, and the output after the final dense layer before the output layer. In the model trained with contextual information, the output of the final dense layer will encode the contextual information introduced after the final GRU layer. The model is fed the training data as input, and t-SNE is used to produce a low-dimension visualization of the high-dimensional intermediate outputs. Fig. 12 shows the two-dimensional mapping of the model outputs at the final GRU layer and the final dense layer, for both models. The SCIKIT-LEARN PYTHON package (Pedregosa et al. 2011) implementation of t-SNE is used in this analysis.

Looking at the t-SNE representation of the outputs for the model trained without contextual information, there appears to be no clear clustering in the representation for the output after the GRU layer in Fig. 12(a). The majority of AGNs occupy the right-hand side of the plot, the majority of SNe are sparsely clustered in the bottom-right region, and few AGNs and SNe occupy the left-hand side. In Fig. 12(b), the t-SNE representation of the output after the final dense layer shows some coherent clustering of SN and AGN. AGNs cluster around the top-right and bottom-left region, with some spread out in the middle. There is a compact structuring of SN on the left-hand side, and a cluster of SNe in the bottom-left near the AGN. There are a few AGNs and SNe in the top-left region of the plot.

From Fig. 12(c) for the output after the final GRU layer for the model trained with contextual information, there appears to be no clear cluster of objects from the same class. AGNs are spread out across the plot, and there is some clustering of SNe in the top-right region. Looking at Fig. 12(d) after including contextual information, the clustering of objects from the same class becomes more apparent. There appears to be a compact and distinct cluster of AGNs near the bottom of the plot, and a tight clustering of SNe along with AGN in the bottom-right region. There is some overlap between SN and AGN, indicating where some of the misclassifications are occurring.

We note that the VS class is a very broad class, containing a multitude of different variable objects with distinct subclassifications. In Fig. 12(d), there seem to be a few coherent structures that may indicate where variable objects from the same or similar classes are located.

From Fig. 12, it is clear that the incorporation of contextual information into the classifier provides useful information that allows the model to learn better class separability. Using t-SNE in this way is a method to provide an approximate measure of 'feature importance' in DL models, which can be challenging compared to deriving feature importance in feature-based ML algorithms such as random forests that utilize hand-made features.

It should be noted that t-SNE is primarily a *data visualization* tool, and here it is used to visualize class separability learnt by the model at different stages. It is possible to explore the hyperparameter space when generating t-SNE plots, and use a number of diagnostics to assess the 'quality' of the dimensionality reduction (such as comparing distances between points in high- and low-dimensional space) and identify any correlating features within the high-dimensional data set (Chatzimparmpas, Martins & Kerren 2020). However, further interpretation of the t-SNE plots is beyond the scope of this work.

6 DISCUSSION

Here, we provide a discussion on the task of object classification in the context of the GOTO survey, how we handle class imbalance in this work, and how the use of contextual information in addition to time-series data from light curves can help classifiers produce more robust classifications.

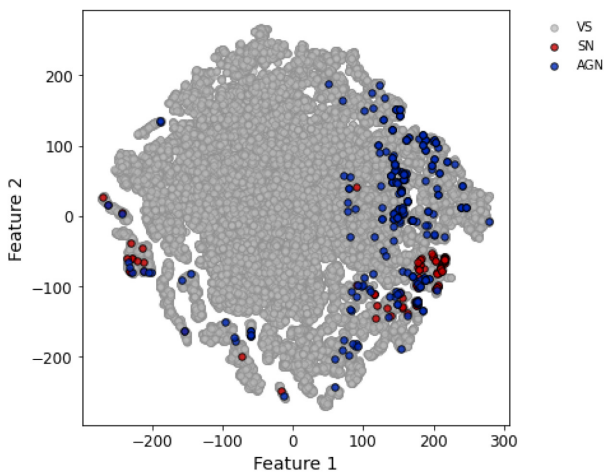
6.1 Classification strategy for GOTO

In this work, the data set of 99 201 labelled objects is split into three broad classes: VSs, SNe, and AGNs. There are over 98 000 VS light curves in the data set, and more than 350 unique class labels provided in the AAVSO catalogue (Watson et al. 2006). A number of these labels include a mixture of classes, where there is no certainty to which class an object belongs to. Grouping all variable-type objects into a single superclass makes the classification task simpler for the model, and allows for a reliable early-time classifications. We also assume that the classifications provided in these catalogues are the ground truth. To create an extended and more robust labelled data set, additional catalogues can be used to provide more labelled examples, and also verify the labels provided for already known objects.

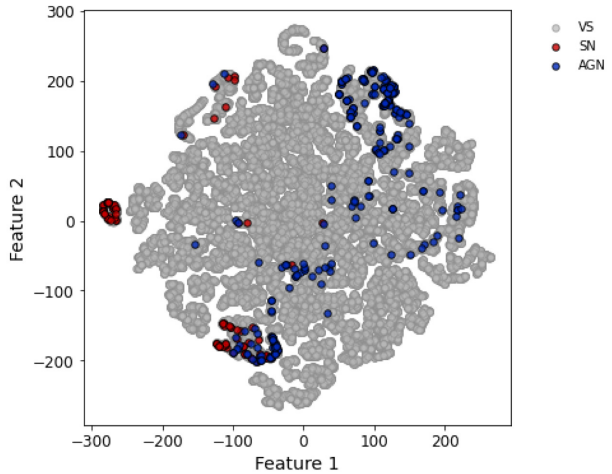
The data set used to train the classifier was obtained from observations conducted in the GOTO all-sky survey mode. Given that the light curves in the data set can be quite sparse, and are only in a single filter, this may have presented a challenge for classification into multiple object subtypes. A more refined classifier can be trained to classify between subtypes of objects. An approach for a more refined classification could be to take a hierarchical classification scheme as in Hosenie et al. (2020), where once a subset of objects has been classified into broad types, another classifier is used to further classify the subset into more specific classes.

Targeted subsky surveys that focus on observing a smaller area of sky and local galaxy clusters in search of transients present a chance to conduct observations at higher cadences and in multiple filters. Additional colour information and more uniform light curve sampling can provide data that can be used to train models to produce better and more refined classifications into object subtypes. Models

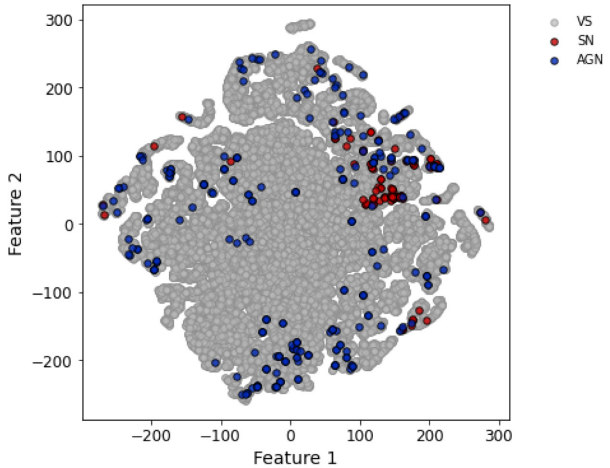
Downloaded from https://academic.oup.com/mnras/article/505/3/4345/6287587 by University of Leicester user on 02 September 2021



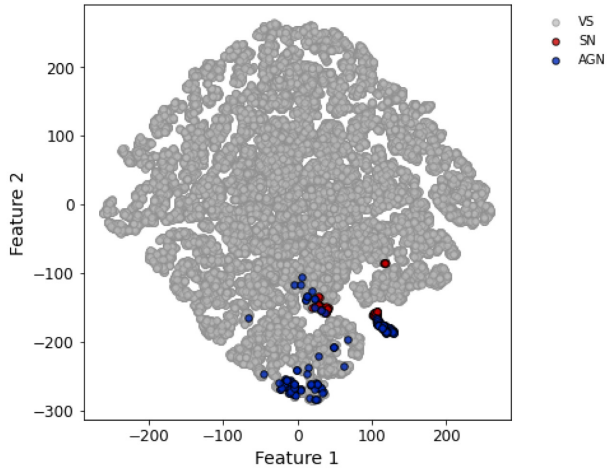
(a) t-SNE representation of model output after the final GRU layer, with no contextual information.



(b) t-SNE representation of model output after the dense layer before the output layer, with no contextual information.



(c) t-SNE representation of model output after the final GRU layer, with contextual information.



(d) t-SNE representation of model output after the dense layer before the output layer, with contextual information.

Figure 12. t-SNE representation for the network outputs at different stages. Each data point corresponds to a single object in the training set; grey points are VS, red points are SN, and blue points are AGN.

trained on light curves in multiple filters have been shown to be able to differentiate between different types of transients and SN subtypes (Muthukrishna et al. 2019; Villar et al. 2019).

One of the objectives of this work was to highlight the usefulness of RNNs for real-time object classification in surveys. Work is currently ongoing to implement this classifier in the GOTO discovery pipeline, where new objects observed by GOTO are classified and the classification is returned to the GOTO Marshall (Lyman et al., in preparation), where it can be displayed in a web interface for users to see. The classifier returns class probabilities that give a measure of how confident the classifier thinks an object belongs to a certain class. This information can be used to decide to trigger additional follow-up observations. With the RNN architecture, it is possible to update the classification probability as new observations of an object are made.

6.2 Handling class imbalance in deep neural network architectures

The data set used to train the RNN classifier for GOTO light curves presented a class imbalance problem. Other works dealing with class imbalance have utilized data augmentation methods for classification with feature-based ML algorithms (Revsbech, Trotta & van Dyk 2018; Boone 2019; Villar et al. 2019; Hosenie et al. 2020) and achieved good performance. Here, we provide an alternative to data augmentation with an RNN classifier, and use an algorithm-level approach to dealing with imbalance, where a weighted cross-entropy loss function and a weighted focal loss function are used to account for imbalanced class distribution in the training set. We train two types of RNN architectures, the LSTM and GRU, with three different loss functions: an unweighted cross-entropy, a weighted cross-entropy, and a weighted focal loss.

Weighting the cross-entropy loss function shows an improvement over the unweighted cross-entropy, going from an AUC of 0.948 to 0.968 for the best unweighted cross-entropy and weighted cross-entropy loss models when evaluated on the test set, respectively. The confusion matrices show that the degree of confusion between the majority and minority classes is reduced by simply weighting the cross-entropy loss function.

The focal loss models perform similar to the weighted cross-entropy loss models. The AUC for the best focal loss model was 0.972, evaluated on the test set. For the focal loss models, the GRU model achieves a higher AUC score. We have shown that by using an

Downloaded from https://academic.oup.com/mnras/article/505/3/4345/6287587 by University of Leicester user on 02 September 2021

appropriate loss function to account for imbalanced data, it is possible to achieve good real-time classification of transient and variable sources, without having to artificially augment the training data.

Krawczyk (2016) note that even if a data set is heavily imbalanced, if the classes are well represented and come from non-overlapping class distributions, it is still possible to achieve good classifications. Future surveys may benefit from obtaining spectroscopy of a wider diversity of targets, and not just based on good signal-to-noise ratios. Having good spectroscopic coverage of sources over a range of magnitudes can help data augmentation efforts to create representative training sets for SN classification (Carrick et al. 2021). Active learning, a class of ML algorithms used to optimize a labelling strategy for unlabelled data, has been used to select spectroscopic follow-up for objects that would give the best improvement to a learning model in SN classification (Ishida et al. 2018). Data augmentation methods could be combined with the approach used in this work for an improvement in classification performance in a class imbalanced setting.

6.3 Contextual information

We trained a GRU model with weighted focal loss to classify light curves using only time-series data without additional contextual information, to investigate the impact of contextual information on classification. The model trained without contextual information achieved an AUC score of 0.902 on the test set, which is the lowest AUC score on the test set out of all models. Without contextual information, the model has an overall worse performance compared to the models with weighted focal loss functions that incorporate contextual information.

Using t-SNE to visualize the intermediate outputs of the GRU models trained with and without contextual information, it can be seen that using contextual information such as location in the sky and distance to the nearest galaxy allows the model to learn better class separability compared to just using information from the light curve. In principle, the model architecture used in this work where contextual information is ingested into the model separately to the time-series information from the light curve can be applied to any survey.

Only three values were used as the contextual information input into all the models: the galactic latitude and longitude, and the distance to the nearest galaxy from the object. It is possible to include additional information, such as additional information about nearby galaxies (colour and metallicity), redshift, and galactic extinction.

In the era of large sky surveys such as ZTF and the Vera Rubin LSST, and the availability of multiple alert brokers (Narayan et al. 2018; Smith et al. 2019; Förster et al. 2020; Möller et al. 2020), additional information on newly discovered objects other than photometric data should be leveraged to provide accurate real-time classification of objects. Being able to classify explosive transients early in their light-curve evolution allows for follow-up in the early stages of evolution that can provide constraints on explosion mechanisms and progenitor environments (Khazov et al. 2016; Zhang et al. 2018).

7 CONCLUSION

In this paper, we present an RNN classifier to classify objects observed with the GOTO survey using their light curves, and additional contextual information such as on-sky position and distance to the nearest galaxy obtained by cross-matching with a catalogue. We create a labelled data set from the GOTO data, and split the data set into three classes: VSs, SNe, and AGNs. The data set is imbalanced, with 99 per cent of labelled objects belonging to the VS class. We adopt weighted cross-entropy and focal loss functions to account for the imbalance, and reduce the model bias towards the majority class. The weighted loss functions improve overall classification performance over the standard approach of an unweighted cross-entropy loss function with deep neural network classifiers. We also train a model without contextual information and only time-series data, and find that it performs worse than the model with the same configuration trained with contextual information, but is still able to provide meaningful classification. Looking at the low-dimensional representations of model outputs shows that contextual information allows the model to make better distinctions between objects.

The classification problem presented in this work is a supervised learning problem, where the expected output of the classifier is known. With future surveys expected to discover orders of magnitudes more objects, a point of interest is identifying previously undiscovered objects – the 'unknown unknowns'. Unsupervised learning is an approach where a model is trained to identify patterns within unlabelled data with minimal human intervention. This approach can be used to identify outliers within the data, and this process is usually referred to as novelty or anomaly detection.

Unsupervised learning algorithms have been used to identify anomalies within the Open Supernova Catalog (Pruzhinskaya et al. 2019), and in conjunction with deep neural network supervised learning for anomaly detection in VSs (Tsang & Schultz 2019). Active learning has also been used for anomaly detection, and to identify informative objects for labelling to improve a learning model as new observations become available (Ishida et al. 2019).

Within the GOTO discovery pipeline, an anomaly detection algorithm can be used once an object has been identified as real to detect potential new discoveries, or applied to subsets of classified objects to identify peculiarities within known classes. The high-dimensional output of the dense layer carries some feature representation of the data, and could be fed into an anomaly detection algorithm. Both supervised and unsupervised classifications are useful in astronomical surveys: Supervised classification provides utility and automation by classifying new objects into known classes, and unsupervised classification acts as a facilitator for specific science goals that utilize observations of rare and novel objects or even discovery of new transients.

ACKNOWLEDGEMENTS

We thank the anonymous referee for their comments that helped us to improve the quality of this paper. The research works of UFB and JRM are funded through a Royal Society PhD studentship (Royal Society Enhancement Award RGF\EA\180234) and University Research Fellowship (Royal Society URF UF150689 and Science and Technology Facilities Council (STFC) grant ST/R000964/1), respectively. VSD and MJD acknowledge the support of a Leverhulme Trust Research Project Grant. RPB, MRK, and DMS acknowledge support from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement number 715051; Spiders). RLCS acknowledges funding from Science and Technology Facilities Council (STFC). Parts of this research were conducted by the Australian Research Council Centre of Excellence for Gravitational Wave Discovery (OzGrav), through project number CE170100004.

JDL acknowledges support from a UK Research and Innovation Fellowship (MR/T020784/1).

The GOTO project acknowledges the support of the Monash–Warwick Alliance; Warwick University; Monash University;

Downloaded from https://academic.oup.com/mnras/article/505/3/4345/6287587 by University of Leicester user on 02 September 2021

Sheffield University; the University of Leicester; Armagh Observatory & Planetarium; the National Astronomical Research Institute of Thailand (NARIT); the University of Turku; the University of Manchester; the University of Portsmouth; the Instituto de Astrofísica de Canarias (IAC); and the Science and Technology Facilities Council (STFC).

UFB would like to thank Maurico A. Àlavarez and Fariba Yousefi for helpful discussions in producing the work presented in this paper.

DATA AVAILABILITY

Data products will be available as part of planned GOTO public data releases.

REFERENCES

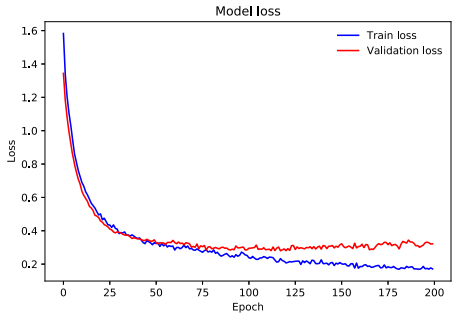
Abadi M. et al., 2016, 12th USENIX Symp. Oper. Syst. Des. Implementation (OSDI 16), TensorFlow: A System for Large-Scale Machine Learning. p. 265, Available at: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf
Abbott B. P. et al., 2009, Rep. Prog. Phys., 72, 076901
Abbott B. P. et al., 2017a, Phys. Rev. Lett., 119, 161101
Abbott B. P. et al., 2017b, ApJ, 848, L12
Acernese F. et al., 2015, Class. Quantum Gravity, 32, 024001
Ball N., Brunner R., 2010, Int. J. Mod. Phys. D, 19, 1049
Becker I., Pichara K., Catelan M., Protopapas P., Aguirre C., Nikzat F., 2020, MNRAS, 493, 2981
Bellm E. C. et al., 2018, PASP, 131, 018002
Bloom J. S., Richards J. W., 2012, in Way M. J., Scargle J. D., Ali K. M., Srivastava A. N., eds, Data Mining and Machine Learning in Time-Domain Discovery and Classification. Chapman Hall, London, p. 89
Bloom J. S. et al., 2012, PASP, 124, 1175
Boone K., 2019, AJ, 158, 257
Brink H., Richards J. W., Poznanski D., Bloom J. S., Rice J., Negahban S., Wainwright M., 2013, MNRAS, 435, 1047
Carrasco-Davis R. et al., 2019, PASP, 131, 108006
Carrick J. E., Hook I. M., Swann E., Boone K., Frohmaier C., Kim A. G., Sullivan M., 2021, preprint (arXiv:2012.12122)
Charnock T., Moss A., 2017, ApJ, 837, L28
Chatzimparmpas A., Martins R. M., Kerren A., 2020, IEEE Transactions on Visualization and Computer Graphics, 26, 2696
Chawla N. V., Bowyer K. W., Hall L. O., Kegelmeyer W. P., 2002, J. Artif. Int. Res., 16, 321
Cho K., van Merrienboer B., Bahdanau D., Bengio Y., 2014, Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, p. 103, preprint (arXiv:1409.1259)
Chollet F. et al., 2021, Keras. https://keras.io
Chornock R. et al., 2017, ApJ, 848, L19
Coulter D. A. et al., 2017, Science, 358, 1556
Dálya G. et al., 2018, MNRAS, 479, 2374
Dauphin F., Hosseinzadeh G., Villar V., Berger E., Gomez S., 2020, American Astronomical Society Meeting Abstracts, #276.18
Drake A. J. et al., 2009, ApJ, 696, 870
Drout M. R. et al., 2017, Science, 358, 1570
Duev D. A. et al., 2019, MNRAS, 489, 3582
Dyer M. J. et al., 2020, in Marshall H. K. Spyromilio J. Usuda T., eds, Ground-based and Airborne Telescopes VIII. SPIE
Fawcett T., 2006, Pattern Recognit. Lett., 27, 861
Filippenko A. V., 1997, ARA&A, 35, 309
Förster F. et al., 2020, AJ, 161, 242
Gieseke F. et al., 2017, MNRAS, 472, 3101
Gómez C., Neira M., Hernández Hoyos M., Arbeláez P., Forero-Romero J. E., 2020, MNRAS, 499, 3130
Goodfellow I., Bengio Y., Courville A., 2016, Deep Learning. MIT Press, Cambridge, MA
Hand D. J., Till R. J., 2001, Mach. Learn., 45, 171
Hochreiter S., Schmidhuber J., 1997, Neural Comput., 9, 1735
Hosenie Z., Lyon R., Stappers B., Mootoovaloo A., McBride V., 2020, MNRAS, 493, 6050
Hosseinzadeh G. et al., 2020, ApJ, 905, 93
Ioffe S., Szegedy C., 2015, in Bach F., Blei D., eds, Proceedings of the 32nd International Conference on Machine Learning, 37, Lille, France, p. 448
Ishida E. E. O., de Souza R. S., 2013, MNRAS, 430, 509
Ishida E. E. O. et al., 2018, MNRAS, 483, 2
Ishida E. E. O. et al., 2019, preprint (arXiv:1909.13260)
Ivezić Ž. et al., 2008, ApJ, 873, 111
Johnson J., Khoshgoftaar T., 2019, J. Big Data, 6, 27
Kaiser N. et al., 2010, in Stepp L. M., Gilmozzi R., Hall H., eds, Proc. SPIE Conf. Ser. Vol. 7733, Ground-Based and Airborne Telescopes III. SPIE, Bellingham, p. 77330E
Kasliwal M. M. et al., 2012, ApJ, 755, 161
Kessler R., Conley A., Jha S., Kuhlmann S., 2010, preprint (arXiv:1001.5210)
Khazov D. et al., 2016, ApJ, 818, 3
Killestein T. L. et al., 2021, MNRAS, 503, 4838
Kingma D. P., Ba J., 2014, ICLR 2015, preprint (arXiv:1412.6980)
Krawczyk B., 2016, Prog. Artif. Intell., 5, 221
Kullback S., Leibler R. A., 1951, Ann. Math. Stat., 22, 79
Leavitt H. S., Pickering E. C., 1912, Harv. Coll. Obs. Circ., 173, 1
Lecun Y., Bengio Y., Hinton G., 2015, Nat. Cell Biol., 521, 436
Lin T.-Y., Goyal P., Girshick R., He K., Dollár P., 2017, IEEE Transactions on Pattern Analysis & Machine Intelligence, 42, 1939
Lin H.-W., Chen Y.-T., Wang J.-H., Wang S.-Y., Yoshida F., Ip W.-H., Miyazaki S., Terai T., 2018, PASJ, 70, S39
Lochner M., McEwen J. D., Peiris H. V., Lahav O., Winter M. K., 2016, ApJS, 225, 31
Mahabal A., Sheth K., Gieseke F., Pai A., Djorgovski S. G., Drake A., Graham M., the CSS/CRTS/PTF Collaboration, 2017, 2017 IEEE Symposium Series on Computational Intelligence (SSCI), preprint (arXiv:1709.06257)
Masci F. J. et al., 2018, PASP, 131, 018003
Möller A., de Boissière T., 2020, MNRAS, 491, 4277
Möller A. et al., 2020, MNRAS, 501, 3272
Mong Y.-L. et al., 2020, MNRAS, 499, 6009
Muthukrishna D., Narayan G., Mandel K. S., Biswas R., Hložek R., 2019, PASP, 131, 118002
Narayan G. et al., 2018, ApJS, 236, 9
Pasquet J., Pasquet J., Chaumont M., Fouchez D., 2019, A&A, 627, A21
Pedregosa F. et al., 2011, J. Mach. Learn. Res., 12, 2825
Perlmutter S. et al., 1999, ApJ, 517, 565
Pruzhinskaya M. V., Malanchev K. L., Kornilov M. V., Ishida E. E. O., Mondon F., Volnova A. A., Korolev V. S., 2019, MNRAS, 489, 3591
Quimby R. M., Aldering G., Wheeler J. C., H"oflich P., Akerlof C. W., Rykoff E. S., 2007, ApJ, 668, L99
Rau A. et al., 2009, PASP, 121, 1334
Revsbech E. A., Trotta R., van Dyk D. A., 2018, MNRAS, 473, 3969
Riess A. G. et al., 1998, AJ, 116, 1009
Shappee B. J. et al., 2014, ApJ, 788, 48
Shappee B. J. et al., 2017, Science, 358, 1574
Smartt S. J. et al., 2017, Nature, 551, 75
Smith K. W. et al., 2019, Res. Notes AAS, 3, 26
Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R., 2014, J. Mach. Learn. Res., 15, 1929
Takahashi I., Suzuki N., Yasuda N., Kimura A., Ueda N., Tanaka M., Tominaga N., Yoshida N., 2020, PASJ, 72
The PLAsTiCC Team et al., 2018, preprint (arXiv:1810.00001)
Tsang B. T. H., Schultz W. C., 2019, ApJ, 877, L14
van der Maaten L., Hinton G., 2008, J. Mach. Learn. Res., 9, 2579
Véron-Cetty M. P., Véron P., 2010, A&A, 518, A10
Villar V. A. et al., 2017, ApJ, 851, L21
Villar V. A. et al., 2019, ApJ, 884, 83
Wardęga K., Zadrożny A., Beroiz M., Camuccio R., Díaz M. C., 2020, preprint (arXiv:2009.14614)

Downloaded from https://academic.oup.com/mnras/article/505/3/4345/6287587 by University of Leicester user on 02 September 2021

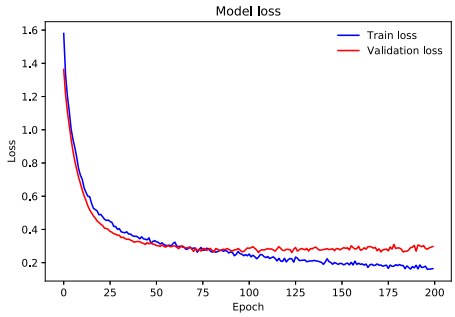
Watson C. L., Henden A. A., Price A., 2006, Soc. Astron. Sci. Annu. Symp., 25, 47
Werbos P. J., 1990, Proc. IEEE, 78, 1550
Wright D. E. et al., 2015, MNRAS, 449, 451
Yasuda N. et al., 2019, PASJ, 71, 74
Zhang J. et al., 2018, ApJ, 863, 109

APPENDIX A: TRAINING AND VALIDATION LOSS GRAPHS

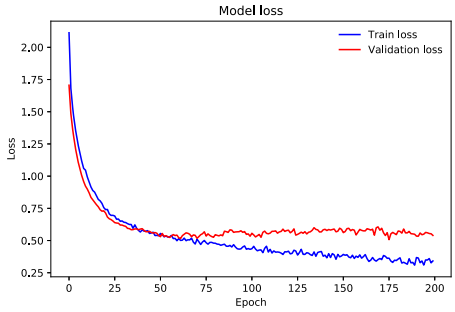
Fig. A1 shows how the models optimize the different loss functions during training. At each epoch, the model weights are updated through gradient descent such that the loss will be minimized. A



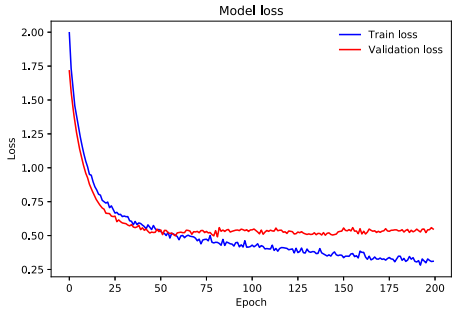
(a) GRU with weighted focal loss.



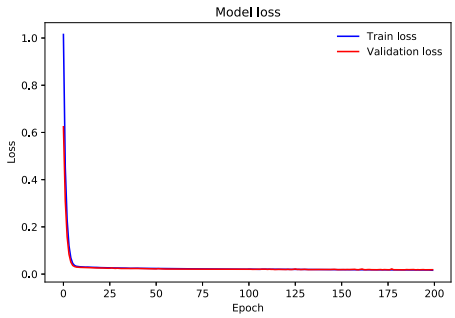
(b) LSTM with weighted focal loss.



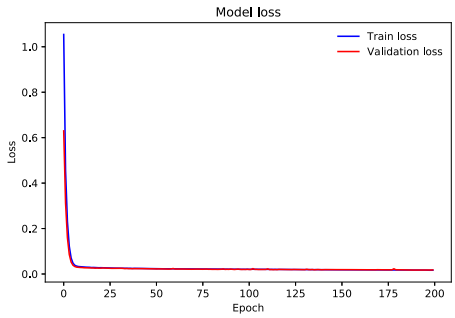
(c) GRU with weighted cross entropy loss.



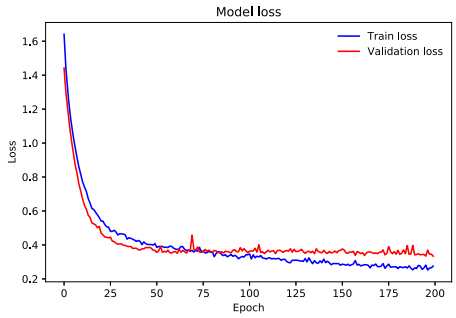
(d) LSTM with weighted cross entropy loss.



(e) GRU with unweighted cross entropy loss.



(f) LSTM with unweighted cross entropy loss.



(g) GRU with weighted focal loss, trained only time-series data

Figure A1. Evolution of training and validation loss for the best performing models during training. Models with the cross-entropy loss converge quickly, but the loss is dominated by contribution from easy-to-classify examples. Models with the weighted cross-entropy loss and focal loss eventually converge within 200 epochs of training, and are also able to account for examples from the minority classes.

Downloaded from https://academic.oup.com/mnras/article/505/3/4345/6287587 by University of Leicester user on 02 September 2021

model is said to converge once the loss stops decreasing, indicating that the model has reached a local minimum in weight space.

APPENDIX B: T-DISTRIBUTED STOCHASTIC NEIGHBOUR EMBEDDING

t-SNE is a dimensionality reduction technique, which takes a high-dimensional data set $\mathcal{X} = \{x_1, x_2, ..., x_n\}$ and converts it into a low-dimensional representation $\mathcal{Y} = \{y_1, y_2, ..., y_n\}$. The low-dimensional data points $\mathcal{Y}$ are mappings of the high-dimensional data points $\mathcal{X}$ in the low-dimensional space.

The high-dimensional Euclidean distances between points are converted into probabilities that represent similarities between points by centring a Gaussian distribution on to each point. The similarity between points x_i and x_j is encapsulated in the joint probability $p_{ij;}$ if x_i and x_j are near, then p_{ij} will be high and if they are far apart then p_{ij} will be low.

In the low-dimensional mapping, the probability q_{ij} is a measure of similarity between y_i and y_j, and q_{ij} will be high if the two points are near each other and low if they are far apart. Instead of centring a Gaussian distribution on to the points in low-dimensional space, a student t-distribution is used instead. Using a heavier tailed distribution allows moderate distances in the high-dimensional map to be modelled by larger distances in the lower dimensional map, preventing a 'crowding' of points that are not too dissimilar (van der Maaten & Hinton 2008).

The variance of the Gaussian distribution is set so that the probability distribution produced by the variance has a fixed perplexity, which is set by the user. The perplexity can be thought of as a measure of the effective number of neighbours in the region of the data point in question. Here, the perplexity parameter is set to 20.

Given the two joint probabilities p_{ij} and q_{ij}, t-SNE determines an optimal low-dimensional mapping $\mathcal{Y}$ of the high-dimensional data set $\mathcal{X}$ by minimizing the Kullback–Leibler divergence (Kullback & Leibler 1951) of q_{ij} and p_{ij} using a gradient descent method.

This paper has been typeset from a TEX/LATEX file prepared by the author.

Downloaded from https://academic.oup.com/mnras/article/505/3/4345/6287587 by University of Leicester user on 02 September 2021