MNRAS **000**, 1–10 (2020) Preprint 20 June 2020 Compiled using MNRAS LaTeX style file v3.0

Astronomical Classification of Light Curves with an Ensemble of Gated Recurrent Units

Siddharth Chaini,[1][⋆] Soumya S. Kumar,[1][†]
[1]*Department of Physics, Indian Institute of Science Education and Research, Bhopal*

20 June 2020

ABSTRACT
With an ever-increasing amount of astronomical data being collected, manual classification has become obsolete; and machine learning is the only way forward. Keeping this in mind, the Large Synoptic Survey Telescope (LSST) Team hosted the Photometric LSST Astronomical Time-Series Classification Challenge (PLAsTiCC) in 2018. The aim of this challenge was to develop ML models that accurately classify astronomical sources into different classes, scaling from a limited training set to a large test set. In this text, we report our results of experimenting with Bidirectional Gated Recurrent Unit (GRU) based models to deal with time series data of the PLAsTiCC data. We demonstrate that GRUs are indeed suitable to handle time series data. With minimum preprocessing and without augmentation, our stacked ensemble of GRU and Dense networks achieves an accuracy of 76.243%. Data from astronomical surveys such as LSST will help researchers answer questions pertaining to dark matter, dark energy and the origins of the universe; accurate classification of astronomical sources is the first step towards achieving this.

Our code is open-source and has been made available on GitHub: `https://github.com/AKnightWing/Astronomical-Classification-PLASTICC`

Key words: methods: data analysis – techniques: photometric

1 INTRODUCTION

Over the last decade, numerous large scale astronomical surveys have been conducted to systematically collect images of the night sky using various spectroscopic and photometric methods. These surveys have, in turn, led to the discovery of an unprecedented number of transients as well as variable astronomical objects. However, with the ever-increasing size of available data, these surveys have also brought to light the problem of astronomical classification for big data.

NASA's Kepler Space Telescope, designed to determine the occurrence rate of Earth-sized planets in temperate orbits around Sun-like stars, photometrically observed about 200,000 stars (Jenkins et al. (2010); Koch et al. (2010); Christiansen et al. (2015)) and discovered thousands of transiting exoplanets (Borucki et al. (2011a), Borucki et al. (2011b); Batalha et al. (2011); Burke et al. (2014); Rowe et al. (2014)). In the first stage of its operation, Sloan Digital Sky Survey (SDSS; Frieman et al. (2007)) measured the spectra of more than 700,000 celestial objects, and the SDSS Supernova Survey measured light curves for a few hundred supernovae, of which spectroscopic confirmations for 500 SN Ia and about 80 core-collapsed supernovae were obtained. Between 2013 and 2019, the Dark Energy Survey (DES; Collaboration: et al. (2016)) recorded information from about 300 million galaxies. The Zwicky Transient Facility (ZTF; Kulkarni (2016)), a collaboration project by Caltech and other notable institutions, had first light at Palomar Observatory in 2017. ZTF produced ~1 terabyte (TB) of raw image data and ~4 TB of real-time data products each night (on an average uninterrupted observing night spanning ~8 hr 40 min) and over the course of the nominal three-year survey, this would amount to ~50 TB of light curve data, ~60 TB of reference image products; the total volume of data amounting to ~3.2 petabytes (PB) (260 good weather observing nights) (Masci et al. 2018). ZFT will form the basis of even larger surveys such as the LSST which will build on ZFT's rapid scans of the sky. The enormous Vera C. Rubin Telescope and its ambitious Legacy Survey of Space and Time will usher in a new-age by generating ~20 TB of data per night, with the final dataset expected to be ~15 PB. During its ten-year survey duration, it is expected to observe 2×10^{10} galaxies, 1.7×10^{10} resolved stars and discover 10^7 supernovae (Collaboration et al. 2009).

These and other surveys have been generating data in various forms in very large quantities; and as technological progress is made in this field, the rate of data generation is only set to increase. Classification of celestial objects by astronomers depends heavily on obtaining their spectroscopic light curves, but extensive spectroscopic data is hard to come by due to the lack of availability of resources. This makes it even more crucial to identify objects

⋆ E-mail: siddharthc17@iiserb.ac.in
† E-mail: soumya17@iiserb.ac.in

of interest for spectroscopic follow-up. Traditionally, astronomers used to classify objects based on visual inspection, but this quickly became a bottleneck as the rate of data generation increased drastically. Numerous citizen-science projects were started to speed up this process by roping in amateur astronomers to help with classification. But the speed of human classification was soon subjugated by the sheer amount of raw data available. The situation called for an intervention utilising computational techniques to automate the process in its initial stages to speed up the rate of object classification as well as to ensure optimal allocation of spectroscopic resources to objects of interest.

The nature of the problem proved to be most suitable for the application of machine learning techniques. One of the first examples of machine learning techniques in astronomy dates back to Bailey et al. (2007). They reported on object classification for supernovae using the Supernovae Factory data with synthetic supernovae as training data. Since photometric data was readily available, the focus centred on using photometric data for classification. The Supernova Photometric Classification Challenge by DES (SNPhotCC; Kessler et al. (2010)) was the first of its kind with a spectroscopically confirmed training set of 1,103 objects and a test set of 20,216 objects without spectroscopic confirmation. Participants were challenged to develop classifiers that could use the known labels of the training set to infer the types of objects in the test set. SNPhotCC paved the way for people from a non-astronomy background to approach the problem of astronomical classification, allowing people to come up with fresh, innovative approaches to this problem.

In astronomy, Cepheid variables and Type Ia Supernovae (SNIa) are used as *standard candles*. Standard candles are sources that have a known luminosity and are used to measure distances. It was measurements with SNIa that led to the discovery of the accelerating expansion of the universe, and thus the discovery of dark energy. Photometry of type Ia supernovae as a function of redshift serves as a powerful probe to aid studies and arrive at important cosmological constraints. Pure samples of type Ia supernovae will prove to be crucial to arrive at statistically relevant conclusions regarding SN Ia properties; which may further lead to a better understanding of dark energy and its properties.

In this context, the LSST team had hosted a competition on Kaggle[1], called Photometric LSST Astronomical Time-Series Classification Challenge (PLAsTiCC; Allam Jr et al. (2018)). The challenge was to correctly classify the test dataset into 15 astronomical classes whilst using a small training set.

Early deep learning approaches (Hinners et al. (2018); Charnock & Moss (2017); Revsbech et al. (2017)) to photometric classification worked well on a test set representative of the training data but had poor performance when the test set was unrepresentative. However, later works (Muthukrishna et al. (2019); Pasquet, Johanna et al. (2019); Möller & de Boissière (2019); Boone (2019);) were successful in this regard. In this text, we describe our approach to the PLAsTiCC problem using deep learning.

Our text is structured in the following way: We talk about our dataset in Section 2, and about our preprocessing and deep learning techniques in Section 3. Finally, in Section 4 we talk about our results obtained using this model and conclude in Section 5.

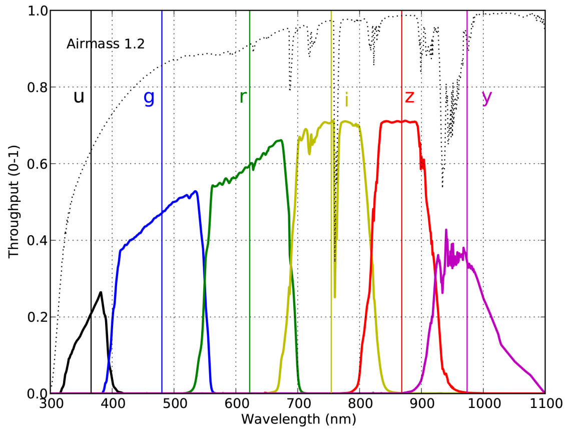

Figure 1. The LSST bandpasses. The vertical axis shows the total throughput. The computation includes the atmospheric transmission (assuming an airmass of 1.2, dotted line), optics, and the detector sensitivity. Source: Ivezić et al. (2019)

2 DATA

2.1 Overview

For the PLAsTiCC Challenge, we will be dealing with univariate time series data. *Time series data* is a series of observations of a variable of interest (in this case, brightness/intensity of light/flux) collected at multiple periods. In astronomy, such a series is called a *light curve*; a graph of light intensity of a celestial object or region, as a function of time. *Spectroscopy* is the measurement of flux emitted across all wavelengths by an object. Spectroscopic light curves are the ultimate tool for classification as they result in the most reliable predictions. However, spectroscopic observations come at the cost of tremendous telescope time, and it is not possible to obtain spectroscopic data for every object observed (LSST expects an average of 10×10^6 alerts per night)(Collaboration et al. 2009). The goal is to narrow down the number of objects of interest for spectroscopic follow-up so that resources can be allocated on a priority basis. Photometry is the measurement of light through optical filters ('passbands') that only permit a specific range of wavelengths to pass through. The LSST will make use of six passbands called *u, g, r, i, z and y* as shown in Figure 2 to measure the brightness of each object. The flux of light in each passband, measured as a function of time, is a light curve.

The PLAsTiCC dataset is a simulation of light curves under realistic observing conditions, as expected to be seen by LSST. The PLAsTiCC dataset consists of 3,500,734 light curves (7,846 in the training set and 3,492,888 in the test set) of various transient as well as variable objects from a mixture of different astrophysical classes, the full list of which is available in Table 1. PLAsTiCC takes the difficulty up a notch from SNPhotCC's supernova classification as it deals with this vast diversity of astrophysical classes.

2.2 The PLAsTiCC Dataset in detail

The PLAsTiCC data was simulated using 18 models developed by the members of the astronomy community at the behest of the

[1] https://www.kaggle.com/c/PLAsTiCC-2018

Table 1. Summary of all the different object types and their counts in our dataset. (Kessler et al. 2019)

Class Number	Class Name	Number of Training Samples	Number of Test Samples
90	Type Ia SN	2,313	1,659,831
67	Peculiar Type Ia SN—91bg-like	208	40,193
52	Peculiar Type Ia SN—SNIax	183	63,664
42	Type II SN	1,193	1,000,150
62	Type Ibc SN	484	175,094
95	Superluminous SN (Magnetar)	175	35,782
15	Tidal disruption event	495	13,555
64	Kilonova	102	133
88	Active galactic nuclei	370	101,424
92	RR Lyrae	239	197,155
65	M-dwarf stellar flare	981	93,494
16	Eclipsing binary stars	924	96,472
53	Mira variables	30	1,453
6	Microlens from single lens	151	1,303
991	Microlens from binary lens	0	533
992	Intermediate luminous optical transient	0	1,702
993	Calcium-rich transient	0	9,680
994	Pair instability SN	0	1,1172

PLAsTiCC team, 14 of which are based on enough observations to be represented in the training set (these form the 14 classes). Rest of the four were combined to form the 15th class found in the test set, as these objects have not been observed enough, or have never been observed but predicted to exist. This is the novel and truly challenging aspect of this competition - building classifiers that can identify unknown objects not present in the training set and correctly classify them into an unknown class. Using these models and three years of LSST observations, over 100 million transient and variable sources were simulated, of which 3.5 million satisfied the detection criteria explained in Kessler et al. (2019). These 3.5 million ugrizy light curves and the corresponding 453 million observations together form the PLAsTiCC dataset(Team & Modelers 2019).

The simulations assume that galactic objects have redshifts set to zero. For extragalactic objects, the simulation includes a model of a follow-up survey as described in Kessler et al. (2019). With this follow-up survey, 3.6% of the extragalactic objects have spectroscopic redshifts for their hosts. Meanwhile, extragalactic objects without spectroscopic redshifts were assigned photometric redshifts obtained using a model described in (Kessler et al. 2019) and were accompanied by their uncertainties. The training set consisted of 7,848 objects with spectroscopically confirmed types, representing a mere 0.2% of the total data (3,500,374 objects). The training set objects typically have a lower redshift than test set objects as spectra are necessary for type confirmation of each object, and it is easier to obtain spectroscopic data for nearby objects. Thus, the training set is biased towards brighter, closer objects.

Another difference between objects arises due to the two distinct regions being monitored. LSST will probe two distinct regions called Deep Drilling Field (DDF) and Wide-Fast-Deep(WFD). DDF will cover ~50 deg^2 and DDF observations are effectively ~1.5 mag deeper and ~2.5 times more frequent than the WFD observations. Thus, objects in the DDF patches have well determined light curves with only small errors in flux. On the other hand, WFD observations will cover almost half of the sky, which will be observed less frequently, increasing uncertainties in the light-curves from objects belonging to this region. Our dataset contains a mix of DDF and WFD objects. While the relative areas of the DDF compared to the WFD for LSST will be 1/400, roughly 1% of the simulated PLAsTiCC data are from the DDF subset (Allam Jr et al. 2018).

The distribution of class sizes is wide, spanning from ~10^2 for the Kilonova class to ~10^6 for a few supernova types, which is quite apparent from Table 1. Many of the light curves are truncated because any given sky location is not visible (at night) from the LSST site for several months of the year. The data consists of irregular time series as it is not sampled at regular time intervals and different passbands are taken at different times, sometimes several days apart. The train: test split ratio is nearly 1:445, a highly skewed dataset with a significant variance in the number of examples of each class present in the training set. Thus our training set is highly imbalanced and non-representative, meaning that the distribution of objects in various classes is not uniform across the training set and test set. This imbalance is a massive departure from most deep learning problems, which tend to have representative datasets and extensive training sets.

The PLAsTiCC training set consists of two comma-separated values (CSV) files - a training set file which consists of light curves of all objects and a header file which consists of summary (astronomical) information available for each object. The PLAsTiCC test set is also similarly divided, with 11 light-curve files and one metadata file amounting to ~18 gigabytes of data.

2.3 MetaData

The header file lists each source in the data indexed by a unique identifier "object_id", that is an integer. Each row of the table lists the properties of the source as follows:

(i) object_id (*int32*): Unique Object Identifier.
(ii) ra (*float32*): right ascension, sky coordinate: longitude, given in degrees.
(iii) decl (*float32*): declination, sky coordinate: latitude, given in degrees.
(iv) gal_l (*float32*): Galactic longitude, given in degrees.
(v) gal_b (*float32*): Galactic latitude, given in degrees.
(vi) ddf (*bool*): A Boolean flag to identify the object as coming from the DDF survey area (with value ddf = 1 for the DDF). =

(vii) hostgal_specz (*float32*): The spectroscopic redshift of the source. This is an extremely accurate measure of redshift, provided for the training set and a small fraction of the test set.
(viii) hostgal_photoz (*float32*): The photometric redshift of the host galaxy of the astronomical source. A substitute for hostgal_specz, this is more error prone.
(ix) hostgal_photoz_err (*float32*): The uncertainty on the hostgal_photoz based on LSST survey projections.
(x) distmod (*float32*): The distance (modulus) calculated from the hostgal_photoz redshift.
(xi) MWEBV = MW E(B-V) (*float32*): This 'extinction' of light is a property of the Milky Way (MW) dust along the line of sight to the astronomical source, and is a function of the sky coordinates of the source ra, decl.
(xii) target (*int8*): The class of the astronomical source. This is provided only for the training data.

2.4 Light Curve Data

Each row of the light-curve table corresponds to an observation of the source at a particular time and passband. This includes the following information:

(i) object id (*int32*): Same as in the metadata table, given as numbers
(ii) mjd (*float64*): The time of the observation, measured in Modified Julian Date (MJD), given in days.
(iii) passband (*int8*): The specific LSST passband integer in which it was viewed, such that u, g, r, i, z, y = 0, 1, 2, 3, 4, 5.
(iv) flux (*float32*): the measured flux (brightness) in the passband of observation as listed in the passband column. The flux is corrected for MWEBV, but the error increases for larger MWEBV values. Note that the units for both flux and flux err are arbitrary.
(v) flux_err (*float32*): the uncertainty on the measurement of the flux listed above, given as float32 number.
(vi) detected (*bool*): This is given as a Boolean flag. The value is 1 if the object's brightness is significantly different at the 3σ level relative to the reference template, and 0 otherwise.

Each night LSST will produce 15 Terabytes of imaging data, and up to ~10^7 transient detections to sift through and find exciting candidates to analyze and to target for spectroscopic observations. This calls for a need to develop early epoch classification based on limited observations so that spectroscopic observations can be scheduled on interesting subsets. Keeping this in mind, we have tried to minimize the use of computational resources and optimize computation time to as great an extent as possible. We have dropped features like sky coordinates, galactic coordinates and the "detected" Boolean flag to reduce the dimensionality of our feature space. This also made it possible for us to handle such a large dataset with limited resources at our disposal.

3 METHODS

3.1 Overview

In this section, we have described the techniques used by us for the photometric classification of light curves. First, we preprocess the given data and extract features from it as described in Section 3.2. We have then described evaluation metrics considered in 3.3 and loss functions considered in 3.4. Our deep learning model consists of a dense-based submodel: **2DSubM** and a GRU-based submodel: **3DSubM**, each named after the shape of their input, respectively. This has been described in Section 3.5, with the hyperparameters in Section 3.6. We then describe our Ensemble technique in Section 3.7, which resulted in a significant boost to our model. We finally talk about the augmentation techniques we tried in Section 3.8, and why they were not useful.

3.2 Preprocessing

Our input data consists of 2 files each for training and testing - light curve data and metadata, as described in Section 2. We preprocess the available data in two separate ways based on the input requirements for the two types of submodels. Our preprocessing consists of the following steps:

3.2.1 2DSubM Data

(i) Adding noise to the flux in the form of a normal distribution having mean zero and standard deviation equal to the $\frac{2}{3}$ times the error in flux (i.e flux_err).
(ii) Scaling down the flux by multiplying every flux value by a quantity we define as:

$$flux\_scaler = \frac{1}{log_2(flux\_max - flux\_min)} \quad (1)$$

where $flux\_max$ and $flux\_min$ are the maximum and minimum flux present in the training set, respectively.
(iii) Aggregating a total of 156 features from the light curve data, based on the features provided by Siddhartha (meaninglesslives) on Kaggle[2].
(iv) Merging the above-aggregated features with the metadata, and dropping the features that are not required.
(v) Replacing all NaNs with zeros.
(vi) Power Transforming the 2D data using the method as described by Yeo & Johnson (Yeo & Johnson 2000).

3.2.2 3DSubM Data

(i) Adding noise to the flux in the form of a normal distribution having mean zero and standard deviation equal to the $\frac{2}{3}$ times the error in flux (i.e flux_err).
(ii) Scaling down the flux by multiplying every flux value by the flux_scaler as defined in (Eq. 1)
(iii) Calculating the difference in dates (MJDs) between observations and using that as a feature instead of the raw MJD value.
(iv) Grouping all observations occurring on the same night(i.e. within 8 hours of each other) together on the basis of their passbands. This gives us a table per object, consisting of various dates and the flux for each passband on that day. In case of multiple observations in a passband on the same night, the entry with the lower error is chosen. In case of no observations for any passband, the date is skipped.
(v) Filling all missing flux entries for a particular passband using linear interpolation(with respect to time) using the preceding and succeeding entries.
(vi) Replacing all NaNs with zeros.

[2] https://www.kaggle.com/meaninglesslives/simple-neural-net-for-time-series-classification#Extracting-Features-from-train-set

(vii) Padding each day's observations with zeroes. Since different objects have different number of observations, and thus different length, we pad them in order to make all sequences equal to the maximum length in the training set (162 in our case).

3.3 Evaluation Metrics

3.3.1 Accuracy:

Accuracy is given by the number of correctly classified examples divided by the total number of classified examples. In terms of the confusion matrix, it is given by:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

where,
TP: True Positive
TN: True Negative
FP: False positive
FN: False Negative

Accuracy is a useful metric when errors in predicting all classes are equally important. In our model, we have assumed equal weights for all classes.

3.3.2 Precision:

Precision is the ratio of correct positive predictions to the overall number of positive predictions. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative. It is given by:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3}$$

3.3.3 Recall:

Recall is the ratio of correct positive predictions to the overall number of positive examples in the dataset. The recall is intuitively the ability of the classifier to find all the positive samples. It is given by:

$$\text{Precision} = \frac{TP}{TP + FN} \tag{4}$$

In multiclass classification, precision/recall are evaluated for a particular class. This is done by considering all examples of the selected class as positives and all examples of the remaining classes as negatives. On Python, sk-learn provides various ways to evaluate precision scores by using the 'average' parameter. We will be using 'macro' precision/recall for our final model evaluation along with accuracy. Macro precision/recall calculates metrics for each label and finds their unweighted mean. This does not take label imbalance into account.

To get an understanding of the performance, we performed 5-fold cross-validation (Brownlee 2018) on the training set, and then selected the best model based on the average accuracy. Then, a random search of the hyperparameter space was performed, as described in Section 3.6. After obtaining the best model, it was retrained from scratch on the whole training set.

3.4 Loss Functions

We experimented with three different loss functions:

3.4.1 Categorical Cross Entropy

Categorical Cross entropy is the loss function used commonly when dealing with multi-class classification that requires output to be a probability value between 0 and 1. Cross entropy loss increases as the predicted probability diverges from the actual target/label.

3.4.2 Focal Loss

Focal Loss is a loss function generalizing binary and multiclass cross-entropy loss that penalizes hard-to-classify examples. It reshapes the loss function to down-weight easy examples and thus focus training on hard negatives by adding a modulating factor $(1 - p_t)^\gamma$ to the cross-entropy loss, with tunable focusing parameter $\gamma \geq 0$.

$$FL(p_t) = -\alpha_t (1 - p_t)^\gamma log(p_t) \tag{5}$$

Focal Loss is mainly used in classification problems dealing with class imbalance, with a sparse set of hard examples (Lin et al. 2017).

3.4.3 LogLoss

LogLoss is defined as the negative of log-likelihood and it is generally used for probabilistic classification rather than deterministic classification. Valid PLAsTiCC entries had to include probability scores for each class and each astronomical source, with the aim of minimizing the PLAsTiCC metric score which is defined as a weighted log-loss metric. We did not experience an improvement in performance with LogLoss and thus discarded it.

3.5 Model Architecture

We have used two types of submodels in our approach. The first is based on deeply connected dense layers, which we call *2DSubM*, while the second type, *3DSubM* is based on a combination of Gated Recurrent Unit (GRU; Cho et al. (2014)) layers and dense layers. The text in this section will describe the best 2 final model arrived at after hyperparameter tuning, whose details have been described in Section 3.6. The two different submodels are trained separately and then later combined using a method called stacking ensemble, in Section 3.7.

3.5.1 2DSubM Architecture

The core of this submodel network consists of fully connected dense layers, taking a two-dimensional array as an input and classification probability as an output. Our best performing network consists of four dense layers consisting of 512, 128, 128 and 64 units, respectively. All of the dense layers are activated using the tanh function, and a dropout of 0.1 and batch normalisation is applied after every dense layer. These help with regularisation and improve training. The output layer is a softmax layer of 15 units. A schematic representation of the 2DSubM model architecture is available in Fig. 2.

The *2DSubM* network takes the preprocessed *2DSubM* Data described in Section 3.2.1 as its input, which consists of metadata and aggregated features. This network was trained using Adam (Kingma & Ba 2014) as the optimizer (learning rate = 0.01), and categorical cross-entropy as the loss function. The average 5-fold cross-validation accuracy of the *2DSubM* architecture was found to be 0.773066.

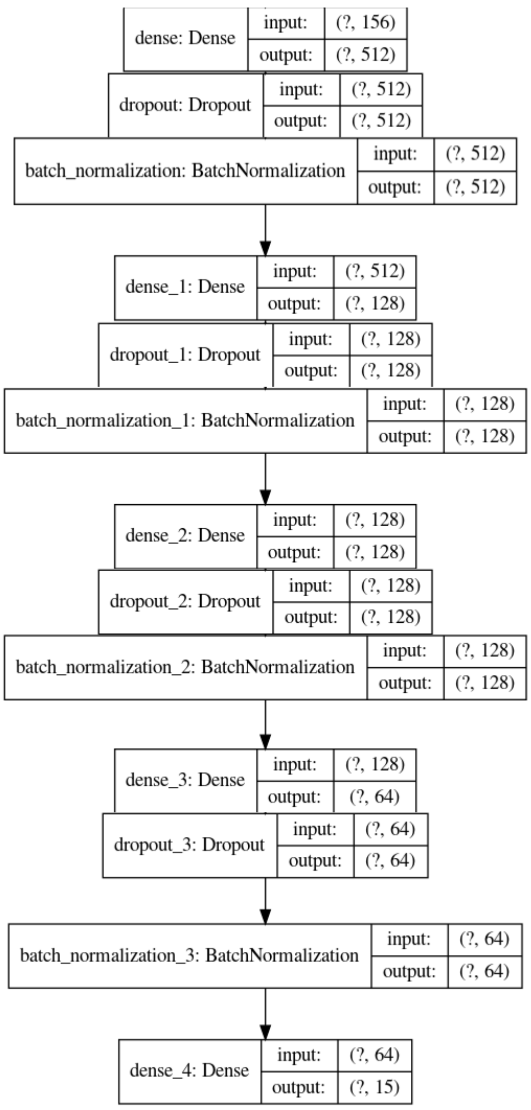
dense: Dense | input: (?, 156) | output: (?, 512)
dropout: Dropout | input: (?, 512) | output: (?, 512)
batch_normalization: BatchNormalization | input: (?, 512) | output: (?, 512)
dense_1: Dense | input: (?, 512) | output: (?, 128)
dropout_1: Dropout | input: (?, 128) | output: (?, 128)
batch_normalization_1: BatchNormalization | input: (?, 128) | output: (?, 128)
dense_2: Dense | input: (?, 128) | output: (?, 128)
dropout_2: Dropout | input: (?, 128) | output: (?, 128)
batch_normalization_2: BatchNormalization | input: (?, 128) | output: (?, 128)
dense_3: Dense | input: (?, 128) | output: (?, 64)
dropout_3: Dropout | input: (?, 64) | output: (?, 64)
batch_normalization_3: BatchNormalization | input: (?, 64) | output: (?, 64)
dense_4: Dense | input: (?, 64) | output: (?, 15)

Figure 2. A schematic representation of the best 2DSubM model architecture

3.5.2 3DSubM Architecture

This submodel network consists of a series of layers of Bidirectional GRUs and Dense layers. This network takes a three-dimensional array as an input, of shape (samples, timesteps, features). A GaussianNoise layer is applied at the start, which adds some noise to the inputs as a form of regularization. This is followed by a series of Bidirectional GRUs, with the output of the last GRU layer being downsampled to a two-dimensional array by the GlobalMaxPooling1D layer. These two-dimensional arrays are then fed to a series of dense layers, which finally returns an output layer consisting of the classification probability. Our best performing network consisted of a GaussianNoise layer with a standard deviation of 0.5, followed by four Bidirectional GRU layers consisting of 256, 64, 32 and 32 units, respectively. All of the GRU layers are activated using the tanh function, and a spatial dropout of 0.1 is applied after every GRU layer, for regularization. This is then followed by 4 densely connected layers consisting of 1024, 256, 64 and 32 units, respectively. All the dense layers are also tanh activated, and followed by a dropout of 0.1. Finally, the output layer is a softmax layer of 15 units.

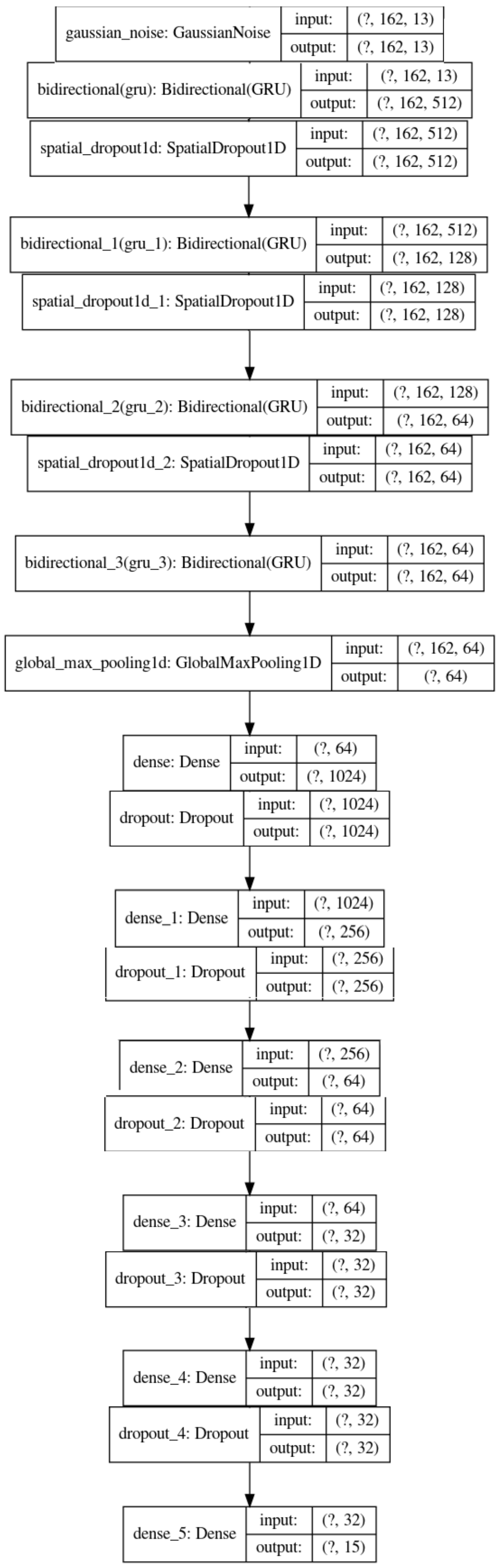
gaussian_noise: GaussianNoise | input: (?, 162, 13) | output: (?, 162, 13)
bidirectional(gru): Bidirectional(GRU) | input: (?, 162, 13) | output: (?, 162, 512)
spatial_dropout1d: SpatialDropout1D | input: (?, 162, 512) | output: (?, 162, 512)
bidirectional_1(gru_1): Bidirectional(GRU) | input: (?, 162, 512) | output: (?, 162, 128)
spatial_dropout1d_1: SpatialDropout1D | input: (?, 162, 128) | output: (?, 162, 128)
bidirectional_2(gru_2): Bidirectional(GRU) | input: (?, 162, 128) | output: (?, 162, 64)
spatial_dropout1d_2: SpatialDropout1D | input: (?, 162, 64) | output: (?, 162, 64)
bidirectional_3(gru_3): Bidirectional(GRU) | input: (?, 162, 64) | output: (?, 162, 64)
global_max_pooling1d: GlobalMaxPooling1D | input: (?, 162, 64) | output: (?, 64)
dense: Dense | input: (?, 64) | output: (?, 1024)
dropout: Dropout | input: (?, 1024) | output: (?, 1024)
dense_1: Dense | input: (?, 1024) | output: (?, 256)
dropout_1: Dropout | input: (?, 256) | output: (?, 256)
dense_2: Dense | input: (?, 256) | output: (?, 64)
dropout_2: Dropout | input: (?, 64) | output: (?, 64)
dense_3: Dense | input: (?, 64) | output: (?, 32)
dropout_3: Dropout | input: (?, 32) | output: (?, 32)
dense_4: Dense | input: (?, 32) | output: (?, 32)
dropout_4: Dropout | input: (?, 32) | output: (?, 32)
dense_5: Dense | input: (?, 32) | output: (?, 15)

Figure 3. A schematic representation of the best 3DSubM model architecture

Table 2. List of Hyperparameters tested with Keras-Tuner

Hyperparameters	Tested Settings
Number of Dense Layers	[1,2,3,4,5,6,7]
Number of GRU Layers	[0,1,2,3,4,5,6]
Number of Dense Units	[128,256,512,1024]
Number of GRU Units	[64,128,256]
Activation Function	[tanh,relu]
Loss Function	[categorical_crossentropy, focal, logloss]
Learning Rate	[0.1, 0.01, 0.001, 0.0001]
Dropout	[0, 0.01, 0.1, 0.2, 0.5]
SpatialDropout1D	[0, 0.01, 0.1, 0.2, 0.5]
GaussianNoise	[0, 0.25, 0.5, 0.75, 1, 2, 5]
Bidirectionality	[True, False]
BatchNormalization	[True, False]

A schematic representation of the 3DSubM model architecture is available in Fig. 3.

This network takes the preprocessed *3DSubM* Data described in Section 3.2.2 as its input, which consists of the light curve data. This network was also trained using Adam as the optimizer (but with the default learning rate = 0.001 this time), while focal loss was chosen as the loss function. The average 5-fold cross validation accuracy obtained was 0.764018.

3.6 Hyperparameter Tuning

We used Keras-Tuner for automating our hyperparameter tuning. We performed an automated random search on the hyperparameter for various parameters, of which we chose the best parameters as described in Section 3.5. The full list of the hyperparameters is in Table 2.

3.7 Stacking Ensemble

Instead of using a single model for prediction, we combined our best performing models by a technique called Stacking Ensemble (Wolpert 1992). The top two performing models each of type *2DSubM* and *3DSubM* (thus a total of 4 submodels) were stacked, and their outputs were combined to form the inputs of a meta learner neural network. A dense layer of 10 units followed this, and finally, a softmax layer of 15 units was used as the output. The weights of the original submodels were set to be untrainable, allowing us to modify only the weights of the last two dense layers. This ensemble was then finally trained on our original validation set, with the Adam optimiser and Categorical Cross-Entropy loss function for a total of 50 epochs. A schematic representation is available in Fig 4. This was then saved as the final model to be used on our test data. The presence of the two types of submodels gives our architecture more diversity.

Our stacking ensemble can thus be described as a neural network which combines our pre-trained neural networks in appropriate proportions to get the best result.

3.8 Data Augmentation

Data Augmentation is the technique of increasing training set examples or populating the data available for training the model without actually collecting any new raw data. This technique is especially useful in cases when the train data to test data ratio is skewed, with very few training examples available as well as cases of imbalanced data, when the distribution across classes is not uniform. It is challenging to devise data augmentation techniques for time series data, especially irregular time series, and this has been covered to a far lesser extent in the literature compared to data augmentation for images.

3.8.1 *Gaussian Process*

A Gaussian Process(GP) is a probability distribution over possible functions. This method of augmentation was implemented in `Avocado`, the top-scoring model of PLAsTiCC (Boone 2019) and in `STACCATO` (Revsbech et al. 2017). It involves fitting GPs to the observations of each object in the training set. Then, synthetic light curves are produced for each object by downsampling and dropping random points from the GPs. This is repeated multiple times, and because of the stochastic nature, each iteration downsamples the GP differently. This gives us a set of new synthetic light curves which are consistent with the observations of the original object. By degrading well-sampled light curves, and adding noise, it is thus possible to generate light curves with similar features as the test set and ensures that the augmented training set is more representative of the test set. Further, a large number of features can then be extracted from these GP-fitted curves, to be fed to the machine learning model.

The chief difficulty that we faced in implementing GP was the lack of computational resources. Gaussian Process regressions are computationally expensive. K. Boone required ~100 core hours of computing time after using an Intel® Xeon® CPU, which was not an option for us.

3.8.2 *Redshift Modifications using Normal Distribution*

This method was implemented in the second-highest scoring model of PLAsTiCC[3]. The preprocessing part involved removing redshift dependency from all time and wavelength related features. This was done by assuming a linear relation on redshift on these features and then dividing them by (1 + hostgal_photoz). Then, up to 30% of the observations were dropped randomly. Redshift was then modified using a normal distribution with the standard deviation:

$$\sigma_{\text{redshift}} = \text{hostgal\_photoz\_err} \times \frac{2}{3} \tag{6}$$

All time and wavelength related features were modified accordingly. Finally, flux was modified using a normal distribution with standard deviation:

$$\sigma_{\text{flux}} = \text{flux\_err} \times \frac{2}{3} \tag{7}$$

When we implemented this augmentation, our model did not show any improvement. We also tried replicating the results achieved by the authors, but our accuracies did not match their results. We later came to know that they had implemented neural network models in conjunction with LightGBM models in an ensemble. Our implementation of this technique yielded accuracies in the 35-40% range and did not improve our performance. Hence, we discarded these data augmentation techniques, choosing instead to work with data modifications and scaling/normalising of features.

[3] `https://www.kaggle.com/c/PLAsTiCC-2018/discussion/75059#446148`

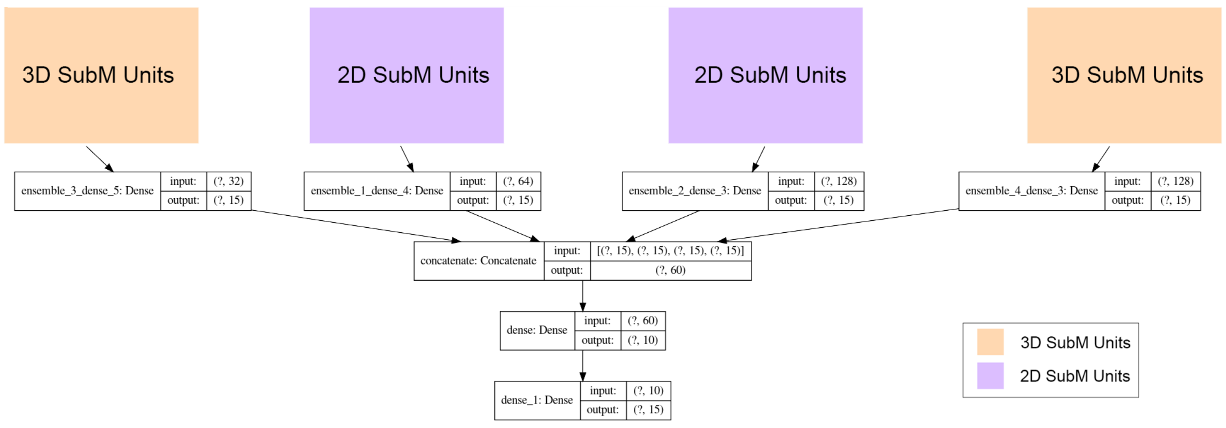
3D SubM Units
2D SubM Units
2D SubM Units
3D SubM Units
ensemble_3_dense_5: Dense | input: (?, 32) | output: (?, 15)
ensemble_1_dense_4: Dense | input: (?, 64) | output: (?, 15)
ensemble_2_dense_3: Dense | input: (?, 128) | output: (?, 15)
ensemble_4_dense_3: Dense | input: (?, 128) | output: (?, 15)
concatenate: Concatenate | input: [(?, 15), (?, 15), (?, 15), (?, 15)] | output: (?, 60)
dense: Dense | input: (?, 60) | output: (?, 10)
dense_1: Dense | input: (?, 10) | output: (?, 15)
3D SubM Units
2D SubM Units

Figure 4. A schematic representation of our stacking ensemble

Table 3. Table Summarising the Metrics Scores produced by our final stacked model on all of the test data.

Metric	Score
Accuracy	76.243 %
Precision	0.62266
Recall	0.50290

4 RESULTS

We ran our final stacked model as mentioned in Section **??**. The scores achieved by our model on the metrics explained in Section 3.3 are listed in Table 3. Preprocessing the entire test set (11 files, amounting to 18.43 gigabytes) took nearly 15 hours to complete. We used Kaggle Notebooks to train and evaluate our model. Training on an NVIDIA® TESLA® P100 GPU provided by Kaggle, took around 2 hours, while evaluation took just under 3 hours.

Figure 5 shows the confusion matrix generated post evaluation. Our model classified classes 16, 65 and 92 exceptionally well with accuracies 97%, 99% and 98% respectively. These classes are eclipsing binary stars, M-dwarf stellar flares and RR Lyrae respectively which have their characteristic light curves and our model successfully learned to classify them using this information. The next best-classified objects were Active Galactic Nuclei (AGN) and Type Ia Supernovae. Our model performed the worst when classifying Peculiar Type Ia SN (both 91bg-like and SNIax) and Type Ibc SN, all three of which were predicted to be Type Ia SN most frequently. Kilonova events were most frequently misclassified as Type II SN. We will discuss the possible reasons behind these in Section 5.

5 DISCUSSION AND CONCLUSION

Our model performed poorly on some classes as mentioned in Section 4. Kilonova events were underrepresented in both the training set as well as the test set with a total of only 231 objects in the entire dataset. The light curves for Type Ib and Type Ic supernovae have been found to be nearly identical to Type Ia supernovae in some cases. Also, SN Ib do not differ much from SN Ia when comparing the absolute magnitude at peak brightness (Tsvetkov 1987). This would explain why the model poorly classifies Type Ibc SN, labelling them as Type Ia SN instead. Poor classification of these classes can be attributed to underrepresentation of class in the training set, rare instances of occurrence in test data (specially Kilonova) and similarity with the major class of the test set.

These drawbacks can be overcome by implementing computationally intensive data augmentation techniques like GP fits which have been shown to significantly improve results by generating a more representative training set. We could also calculate and include more relevant features obtained from the literature pertaining to astronomical objects. These may help to distinguish between similar classes. Another way to improve model performance would be to build a model that predicts hostgal specz as a meta encoder. Working with minimal computational resources, our model could achieve upto 76% accuracy. With more computational resources, we could implement different preprocessing strategies and include more models in the final stacked ensemble to boost performance.

ACKNOWLEDGEMENTS

We thank Dr. Parthiban Srinivasan for his guidance, support and invaluable inputs. We thank the LSST Team for coming up with this innovative challenge. We thank the PLAsTiCC organizers for the simulated data set and for organizing this challenge. We thank Kaggle for providing a platform to host PLAsTiCC. We also thank the Kaggle users over at the PLAsTiCC Discussion Boards for the helpful comments and ideas shared.

Software: Keras (Chollet et al. 2015), Keras-Tuner (O'Malley et al. 2019), NumPy(Oliphant (2006); van der Walt et al. (2011)), Jupyter (Kluyver et al. 2016), Matplotlib (Hunter 2007), scikit-learn (Pedregosa et al. 2011), SciPy (Virtanen et al. 2020), Pandas(pandas development team (2020); Wes McKinney (2010)), TensorFlow (Abadi et al. 2015) and Python3. (Van Rossum & Drake 2009)

REFERENCES

Abadi M., et al., 2015, TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, `http://tensorflow.org/`
Allam Jr T., et al., 2018, arXiv preprint arXiv:1810.00001
Bailey S., Aragon C., Romano R., Thomas R. C., Weaver B. A., Wong D., 2007, The Astrophysical Journal, 665, 1246
Batalha N. M., et al., 2011, The Astrophysical Journal, 729, 27
Boone K., 2019, The Astronomical Journal, 158, 257
Borucki W. J., et al., 2011a, The Astrophysical Journal, 728, 117

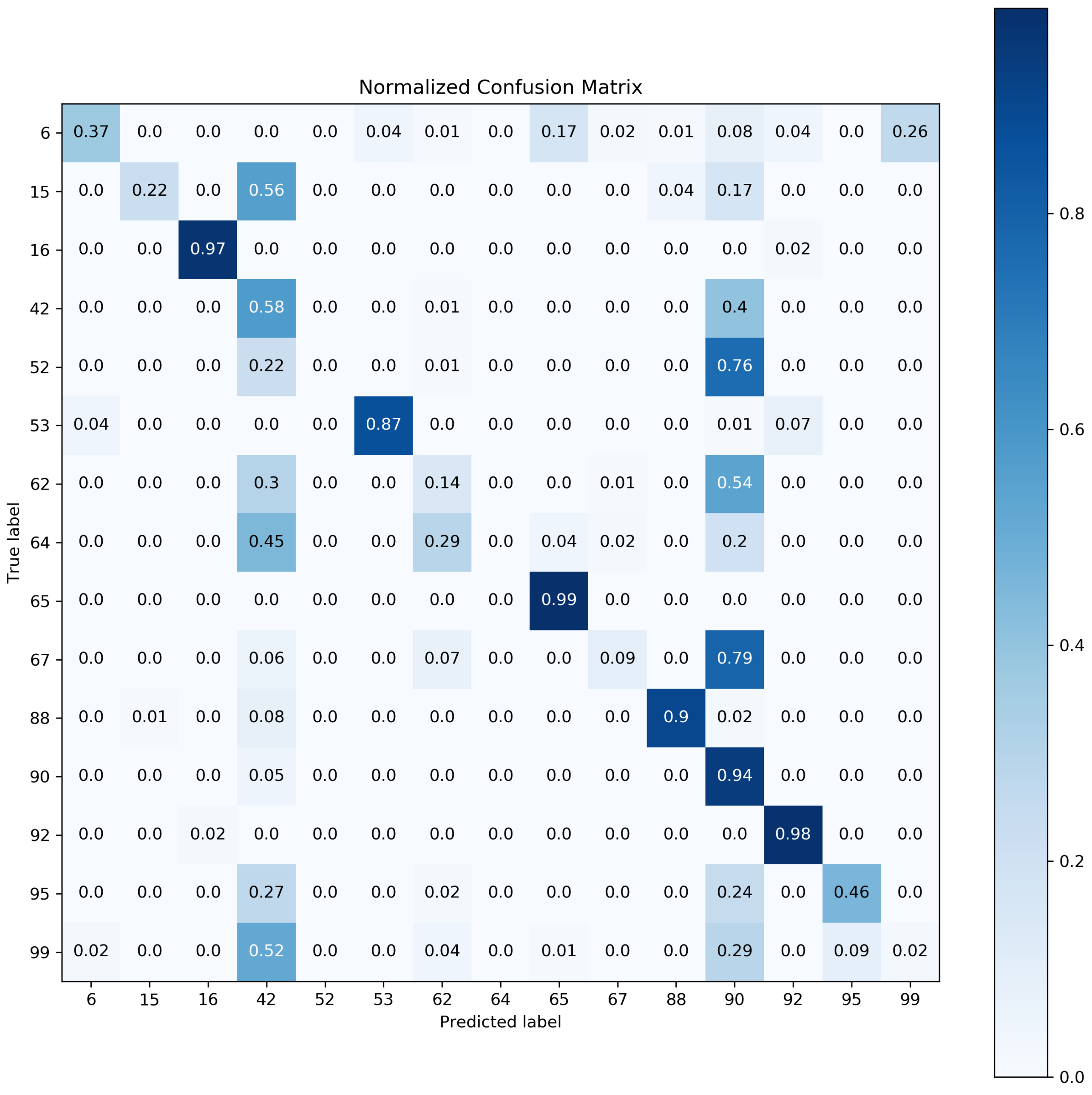

Figure 5. Confusion Matrix of our final stacked model on all of the test data: All the counts have been normalised to unity.

Borucki W. J., et al., 2011b, The Astrophysical Journal, 736, 19
Brownlee J., 2018, Machine Learning Mastery, 2019
Burke C. J., et al., 2014, The Astrophysical Journal Supplement Series, 210, 19
Charnock T., Moss A., 2017, The Astrophysical Journal Letters, 837, L28
Cho K., van Merrienboer B., Gulcehre C., Bahdanau D., Bougares F., Schwenk H., Bengio Y., 2014, Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation (`arXiv:1406.1078`)
Chollet F., et al., 2015, Keras, `https://keras.io`
Christiansen J. L., et al., 2015, The Astrophysical Journal, 810, 95
Collaboration L. S., et al., 2009, LSST Science Book, Version 2.0 (`arXiv:0912.0201`)
Collaboration: D. E. S., et al., 2016, Monthly Notices of the Royal Astronomical Society, 460, 1270
Frieman J. A., et al., 2007, The Astronomical Journal, 135, 338
Hinners T. A., Tat K., Thorp R., 2018, The Astronomical Journal, 156, 7
Hunter J. D., 2007, Computing in Science & Engineering, 9, 90
Ivezić , et al., 2019, The Astrophysical Journal, 873, 111
Jenkins J. M., et al., 2010, The Astrophysical Journal, 713, L120
Kessler R., Conley A., Jha S., Kuhlmann S., 2010, arXiv preprint arXiv:1001.5210
Kessler R., et al., 2019, Publications of the Astronomical Society of the Pacific, 131, 094501
Kingma D. P., Ba J., 2014, arXiv preprint arXiv:1412.6980
Kluyver T., et al., 2016, in Loizides F., Schmidt B., eds, Positioning and Power in Academic Publishing: Players, Agents and Agendas. pp 87 – 90
Koch D. G., et al., 2010, The Astrophysical Journal, 713, L79
Kulkarni S. R., 2016, in American Astronomical Society Meeting Abstracts #227. p. 314.01
Lin T., Goyal P., Girshick R. B., He K., Dollár P., 2017, CoRR,

abs/1708.02002
Masci F. J., et al., 2018, Publications of the Astronomical Society of the Pacific, 131, 018003
Muthukrishna D., Narayan G., Mandel K. S., Biswas R., Hložek R., 2019, Publications of the Astronomical Society of the Pacific, 131, 118002
Möller A., de Boissière T., 2019, Monthly Notices of the Royal Astronomical Society, 491, 4277
O'Malley T., Bursztein E., Long J., Chollet F., Jin H., Invernizzi L., et al., 2019, Keras Tuner, `https://github.com/keras-team/keras-tuner`
Oliphant T. E., 2006, A guide to NumPy. Vol. 1, Trelgol Publishing USA
Pasquet, Johanna Pasquet, Jérôme Chaumont, Marc Fouchez, Dominique 2019, A&A, 627, A21
Pedregosa F., et al., 2011, Journal of Machine Learning Research, 12, 2825
Revsbech E. A., Trotta R., van Dyk D. A., 2017, Monthly Notices of the Royal Astronomical Society, 473, 3969
Rowe J. F., et al., 2014, The Astrophysical Journal, 784, 45
Team P., Modelers P., 2019, Unblinded Data for PLAsTiCC Classification Challenge, doi:10.5281/zenodo.2539456, `https://doi.org/10.5281/zenodo.2539456`
Tsvetkov D. Y., 1987, Soviet Astronomy Letters, 13, 376
Van Rossum G., Drake F. L., 2009, Python 3 Reference Manual. CreateSpace, Scotts Valley, CA
Virtanen P., et al., 2020, Nature Methods, 17, 261
Wes McKinney 2010, in Stéfan van der Walt Jarrod Millman eds, Proceedings of the 9th Python in Science Conference. pp 56 – 61, doi:10.25080/Majora-92bf1922-00a
Wolpert D. H., 1992, Neural Networks, 5, 241
Yeo I., Johnson R. A., 2000, Biometrika, 87, 954
pandas development team T., 2020, pandas-dev/pandas: Pandas, doi:10.5281/zenodo.3509134, `https://doi.org/10.5281/zenodo.3509134`
van der Walt S., Colbert S. C., Varoquaux G., 2011, Computing in Science Engineering, 13, 22

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