The Astrophysical Journal, 885:85 (18pp), 2019 November 1 https://doi.org/10.3847/1538-4357/ab48f4

DASH: Deep Learning for the Automated Spectral Classification of Supernovae and Their Hosts

Daniel Muthukrishna[1,4], David Parkinson[2,3], and Brad E. Tucker[4,5,6]
[1] Institute of Astronomy, University of Cambridge, Madingley Road, Cambridge CB3 0HA, UK; daniel.muthukrishna@ast.cam.ac.uk
[2] School of Mathematics and Physics, University of Queensland, Brisbane, QLD 4072, Australia
[3] Korea Astronomy and Space Science Institute, 776, Daedeokdae-ro, Yuseong-gu, Daejeon 34055, Republic of Korea
[4] Mt Stromlo Observatory, The Research School of Astronomy and Astrophysics, Australian National University, ACT 2601, Australia
[5] National Centre for the Public Awareness of Science, the Australian National University, Canberra, Australia
[6] The ARC Centre of Excellence for All-Sky Astrophysics in 3 Dimensions (ASTRO 3D), Australia
Received 2019 March 1; revised 2019 September 15; accepted 2019 September 27; published 2019 November 1

Abstract

We present DASH (Deep Automated Supernova and Host classifier), a novel software package that automates the classification of the type, age, redshift, and host galaxy of supernova spectra. DASH makes use of a new approach that does not rely on iterative template-matching techniques like all previous software, but instead classifies based on the learned features of each supernova's type and age. It has achieved this by employing a deep convolutional neural network to train a matching algorithm. This approach has enabled DASH to be orders of magnitude faster than previous tools, being able to accurately classify hundreds or thousands of objects within seconds. We have tested its performance on 4 yr of data from the Australian Dark Energy Survey (OzDES). The deep learning models were developed using TensorFlow and were trained using over 4000 supernova spectra taken from the CfA Supernova Program and the Berkeley SN Ia Program as used in SNID (Supernova Identification software). Unlike template-matching methods, the trained models are independent of the number of spectra in the training data, which allows for DASH's unprecedented speed. We have developed both a graphical interface for easy visual classification and analysis of supernovae and a Python library for the autonomous and quick classification of several supernova spectra. The speed, accuracy, user-friendliness, and versatility of DASH present an advancement to existing spectral classification tools. We have made the code publicly available on GitHub and PyPI (pip install astrodash) to allow for further contributions and development. The package documentation is available at https://astrodash.readthedocs.io.

Key words: methods: data analysis – methods: statistical – supernovae: general – surveys – techniques: spectroscopic

1. Introduction

Supernovae (SNe) have been pivotal to modern observational cosmology. The use of Type Ia SNe (SNe Ia) as standard candles has provided some of the most compelling evidence for the discovery that the expansion of the universe is accelerating (Riess et al. 1998; Schmidt et al. 1998; Perlmutter et al. 1999). However, the nature of dark energy and the value of many cosmological parameters are still under active consideration (Muthukrishna & Parkinson 2016; Zhang et al. 2017). To this end, several large-scale surveys, including the Dark Energy Survey (Dark Energy Survey Collaboration et al. 2016), the Supernova Legacy Survey (SNLS; Astier et al. 2006), and ESSENCE (Davis et al. 2007), have aimed to increase the total set of SNe in order to gain a better understanding of dark energy. Moreover, in the near future, projects such as the Large Synoptic Survey Telescope (LSST; LSST Science Collaboration et al. 2009) will substantially increase the transient catalog with the expectation to observe orders of magnitude more SNe than ever before.

The field of observational astronomy has reached a new era of "big data," where we are collecting more data than humans can possibly process and classify alone. Machine learning techniques have been a key driver in tackling these new large-scale problems, and many successful attempts have been used to solve large data astronomy problems (Ball & Brunner 2010). More recently, however, deep learning has gained a lot of popularity in the machine learning community for its accuracy, efficiency, and flexibility. In particular, convolutional neural networks (CNNs) have achieved remarkable results in a range of different applications, including image and speech recognition challenges, outperforming previous approaches (e.g., Krizhevsky et al. 2012; Razavian et al. 2014; Szegedy et al. 2014). Only after the Galaxy Zoo Challenge (Lintott et al. 2008; Dieleman et al. 2015), however, did it begin to gain a larger interest in the astronomy community (e.g., Aniyan & Thorat 2017; Cabrera-Vives et al. 2017).

While machine learning has been applied to photometric SN classification (e.g., Lochner et al. 2016; Möller et al. 2016; Charnock & Moss 2017; Moss 2018; Narayan et al. 2018; Muthukrishna et al. 2019), few attempts at spectral classification of any kind have been made. While this project was being developed, a paper by Sasdelli et al. (2016) applied deep learning to SN spectra: using it to explore the spectroscopic diversity in SNe Ia. Moreover, a recent thesis by Hála (2014) has applied a similar CNN approach to that described in this paper to the spectral classification of quasars, stars, and galaxies. SNe are inherently more complicated, however, due to the fact that they vary with time and have degeneracies in their type, age, and redshift, with often lower signal-to-noise ratio (S/N) caused by distortions from their host galaxy.

In fact, there are several factors that make SN classification a challenging problem. While different types of SNe are distinguished by the presence of particular absorption features in their spectra, the problem of spectral classification is made

difficult by the fact that the spectrum changes depending on the number of days since maximum light at which it was observed (defined as "age" in this paper). Each spectrum also has distortions due to contamination from host galaxy light. Moreover, the redshift at which the SN is observed impacts which spectral features are visible in the observed wavelength range and also affects the S/N, which decreases with redshift. Extinction from interstellar dust further impacts the spectra. Subtracting the continuum from each spectrum can limit this issue by placing more emphasis on the spectral features instead of the color information. Finally, issues with the telescope used to observe the spectrum, such as dichroic jumps being caused by miscalibrations between the two spectral arms using different CCDs, and also telluric features from Earth's atmosphere, are further problems that need to be accounted for when classifying spectra.

1.1. Prior Software

Due to these complications, existing SN spectral classifiers are not able to automate the classification process. Currently, the process of classifying SNe is very slow and labor-intensive, with the classification process for a single SN taking up to a few hours with the incessant input of an experienced astronomer. Surveys like the Australian Dark Energy Survey (OzDES; Yuan et al. 2015a; Childress et al. 2017) are observing thousands of transient objects that need to be classified, and current methods make this an enormously time-consuming process. SNID (Blondin & Tonry 2007) and Superfit (Howell et al. 2005) are the two main spectral classifier software packages used to classify SNe. SNID is a fast typing tool written in Fortran. It makes use of the cross-correlation algorithms of Tonry & Davis (1979) and has been effective in distinguishing SN subtypes at a range of redshifts. However, its accuracy drops significantly when there is host galaxy contamination or if the spectra have a low S/N. In such cases, Superfit acts as a better tool owing to its ability to classify host-contaminated spectra and account for extinction, and as such it is the primary tool used by large surveys such as OzDES and SNLS. Its main downfall, however, is that it is often very slow and requires a lot of user input to constrain priors on redshift, host, and SN type. Superfit is written in IDL and makes use of a chi-squared minimization approach to classify the spectra. It accounts for the SN type, age, host galaxy, and extinction in its minimization equation, which enables it to be a very effective tool. However, given the thousands of transient objects that are being detected by the latest era of SN surveys, a faster and more autonomous software is required.

DASH makes use of the techniques used in each of these previous tools. In particular, the spectra in DASH are processed in a very similar method to the log wavelength spectra developed by SNID (see Section 2.3). Moreover, the *rlap* ranking system developed by Blondin & Tonry (2007) is available in DASH and is used as a test for misclassifications (along with the machine learning scores) in much the same way as SNID.

All previous spectral tools for classification and redshifting make use of the Tonry & Davis (1979) cross-correlation technique (i.e., SNID, MARZ, Hinton et al. 2016; AUTOZ, Baldry et al. 2014; RUNZ) or a chi-squared minimization approach (i.e., Superfit). However, using either of these techniques means that the total computation time increases linearly with the number of spectra in the data set. Both SNID and Superfit can only compare an input spectrum with one other spectrum at a time, and their accuracy is highly reliant on their data set. DASH improves on this by using the aggregate features of a particular class of SN instead of comparing to a single spectrum. DASH is able to learn from the features of all spectra in an SN class and classify on that, instead of comparing to just one spectrum at a time like previous tools.

1.2. Overview

We have developed a new SN spectral classification tool, DASH (Deep Automated Supernova and Host classifier), to quickly and accurately determine the type, age, redshift, and host galaxy of SN spectra. We make use of a CNN, which greatly improves on many aspects of previous classification tools. In Section 2, we detail the data sets we have collated and the preprocessing techniques that are uniformly applied to the spectra. In Section 3, we describe the CNN architecture that we use. In Section 4, we outline the four different trained models that are available in the DASH release, before describing the algorithms used to redshift and to warn the user against possible misclassifications. In Appendix B, we outline how to use the Python library and graphical interfaces, as well as detail the platform requirements and the code development. Finally, in Section 5, we evaluate the performance of DASH on a validation set and the recent OzDES data.

2. Data

SNe are the result of either the core collapse of massive stars or the thermonuclear disruption of carbon-oxygen white dwarfs accreting matter from a binary companion. They are classified based on the presence of certain features in their optical spectrum taken near maximum light instead of their explosion mechanism. The presence or absence of hydrogen, silicon, and helium spectral features separates SNe into four broad types: Type Ia (SN Ia), Type Ib (SN Ib), Type Ic (SN Ic), and Type II (SN II). Within each of these, several subtypes have been defined owing to a range of peculiarities in their spectra. DASH makes use of 17 subtypes defined by Blondin & Tonry (2007), Modjaz et al. (2016), and Silverman et al. (2012):

SN Ia: Ia-norm, Ia-pec, Ia-91T, Ia-91bg, Ia-csm, Iax.
SN Ib: Ib-norm, Ib-pec, Ib-n, IIb.
SN Ic: Ic-norm, Ic-pec, Ic-broad.
SN II: IIP, II-pec, IIL, IIn.

In order to train the model, it was important that we collected a wide range of spectra encompassing each of these subtypes over a range of different ages. The quality of the classification model is highly dependent on the data that it was trained on, and hence in this section we detail how the data were collected, outline the decisions made that led to the final data set, and describe the systematic preprocessing techniques applied to the data before they were trained using a deep CNN.

2.1. Description

We collected labeled spectra from three main repositories: the SNID database, the Berkeley Supernovae Ia Program (BSNIP), and the releases from Liu & Modjaz in 2014–2016.

2.1.1. SNID Database

The latest version of the SNID database (Templates 2.0[7]) has compiled 3716 spectra from 333 different SNe obtained from 1979 to 2008 (Blondin & Tonry 2007; Blondin et al. 2012). These were collected from the SUSPECT public archive[8] (Yaron & Gal-Yam 2012), the CfA Supernova Archive,[9] and the CfA Supernova Program (Matheson et al. 2008; Blondin et al. 2012). The collected set was selected to have a high S/N and has been cleaned, de-redshifted, continuum-divided, smoothed, and processed onto a log wavelength scale by SNID in a process defined by Blondin & Tonry (2007). The spectra were classed into 14 different subtypes: Ia-norm, Ia-pec, Ia-91T, Ia-91bg, Ia-csm, Ib-norm, Ib-pec, IIb, Ic-norm, Ic-broad, IIP, II-pec, IIL, IIn. A detailed description of these subtypes can be found in Blondin & Tonry (2007).

We removed the SNe where the date of maximum light was unknown, and we were left with a total of 3618 spectra from 317 different SNe. This distribution comprised 2724 spectra from 283 SNe Ia, 223 spectra from 12 SNe Ib, 183 spectra from 11 SNe Ic, and 488 spectra from 11 SNe II.

2.1.2. Liu & Modjaz

In 2014–2016, Yuqian Liu and Maryam Modjaz released a series of papers (Liu & Modjaz 2014; Modjaz et al. 2014, 2016; Liu et al. 2016) that collected the largest set of stripped-envelope core-collapse SNe (SNe Ib and SNe Ic). The spectral database was downloaded from their GitHub repository[10] and contained 1045 spectra across 96 SNe Ib and SNe Ic. Within this set, Liu & Modjaz (2014) corrected 14 SNe also included in the SNID Templates 2.0 release, which had incorrect type or age information. In addition, they introduced two new subtypes called Ib-n (defined in Pastorello et al. 2008) and Ic-pec to better account for variations in some spectra.

We again removed the SNe where the date of maximum light was unknown, and we were left with a total of 571 spectra from 57 SNe. The distribution comprised 323 spectra from 27 SN Ib, 248 spectra from 30 SN Ic, and zero SN Ia or SN II spectra.

2.1.3. BSNIP

In 2012, Silverman et al. (2012) collated 1126 spectra from 277 SNe as part of the Berkeley SN Ia Program (BSNIP) in the BSNIP v7.0 release.[11] Many of these were, however, also part of the SNID Templates 2.0 set and the Liu & Modjaz updates. After removing exact duplicates in the BSNIP v7.0 database and also removing spectra with an unknown date of maximum light, we were left with 604 spectra across 133 SNe. This reduced set had 29 new SNe and 114 SNe that were common to the previously discussed data sets but included spectra at different ages. The distribution comprised 564 spectra from 131 SNe Ia, 40 spectra from two SNe Ic, and zero SN Ib or SN II spectra.

The BSNIP release also defined two new subtypes called Ia-02cx (renamed Iax) and Ia-99aa (defined in Silverman et al. 2012; Foley et al. 2013). We discussed these subtypes with the author, Jeffrey Silverman, and he believed that SNe Ia-99aa are a subset of the Ia-91T type and may not need their own category. Based on this discussion, and the fact that there were not enough SN Ia-99aa spectra to train its own subtype, we reclassified the SN Ia-99aa spectra as SNe Ia-91T.

While duplicate spectra between the data sets were removed, wherever there were discrepancies in the phase or subtype of an SN, we preferentially selected the Liu & Modjaz spectra because they intentionally improved upon the SNID Templates 2.0 release. There were a total of six discrepancies in the subtypes of SNe from the BSNIP v7.0 and the SNID Templates 2.0 data sets. The subtypes from the BSNIP data set were selected in favor of the Templates 2.0 data set because BSNIP intentionally improved on the SNID data set. The following changes were made: sn2002cx, sn2005hk, and sn2008A from Templates 2.0 were changed from Ia-pec to Iax subtypes; and the sn1995ac, sn2000cn, and sn2004aw from Templates 2.0 were changed to Ia-91T, Ia-91bg, and Ic-pec from the norm subtypes, respectively.

2.2. DASH Data Distribution

Combining the spectra from the SNID Templates 2.0 database, the Liu & Modjaz updates, and the BSNIP v7.0 release, and removing spectra with unknown ages, we were left with a total of 4831 unique spectra across 403 unique SNe. The distribution comprised 3288 spectra from 312 SNe Ia, 550 spectra from 40 SNe Ib, 505 spectra from 40 SNe Ic, and 488 spectra from 11 SNe II.

In general, SNe that are observed several weeks before or after maximum light are usually very dim, and their spectra are mostly dominated by host galaxy light. Thus, we only considered SNe in the range of -20 to $+50$ days since maximum light. After removing spectra outside this range, we were left with 3899 spectra from 403 SNe. In order to group the spectra into bins that can be trained on for the machine learning algorithm, we split the ages into 4-day intervals. Therefore, for each of the 17 SN subtypes, there are 18 age bins, leading to a total of 306 different classes to separate all of the spectra. The distributions of spectra across the SN subtypes and ages are illustrated in Figures 1 and 2, respectively. The complete distribution in each type and age classification bin is listed in Appendix A, Figure 7.

2.2.1. Data Augmentation

Figures 1, 2, and 7 illustrate two significant problems. First, there are several bins with zero spectra, meaning that DASH (and previous tools such as Superfit and SNID) will never be able to classify a spectrum into this bin. There is no way to fix this problem other than to observe a wider range of SNe. In fact, while the data set has proven to be sufficient for effective classification (see Section 5), it is expected that if a wider and deeper range of spectra is added to the training set, the accuracy —particularly for low-S/N spectra—will improve.

Second, the rarity of some SN types and the bias of cosmological surveys to preferentially observe SNe Ia near maximum light over other types mean that there is a large imbalance in the data set. In SNID and Superfit, this leads to a "type attractor" (Blondin & Tonry 2007), whereby low-S/N spectra will preferentially be classified as SNe Ia regardless of their actual type, simply because there are more SN Ia spectra to choose from. Moreover, while this is a large set of SNe, in

[7] https://people.lam.fr/blondin.stephane/software/snid/index.html
[8] https://www.nhn.ou.edu/~suspect/
[9] http://www.cfa.harvard.edu/supernova/SNarchive.html
[10] https://github.com/nyusngroup/SESNtemple/tree/master/SNIDtemplates
[11] https://people.lam.fr/blondin.stephane/software/snid/index.html

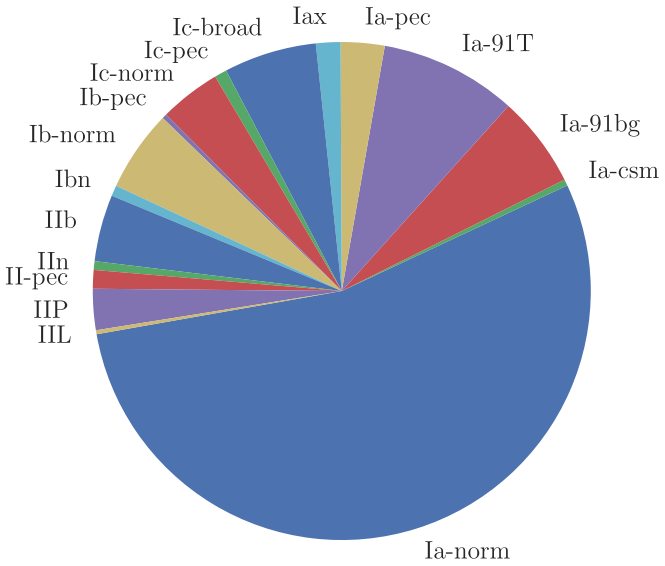

Figure 1. Fraction of spectra for each subtype in the final data set. The total distribution separated by subtype and age is listed in Appendix A, Figure 7.

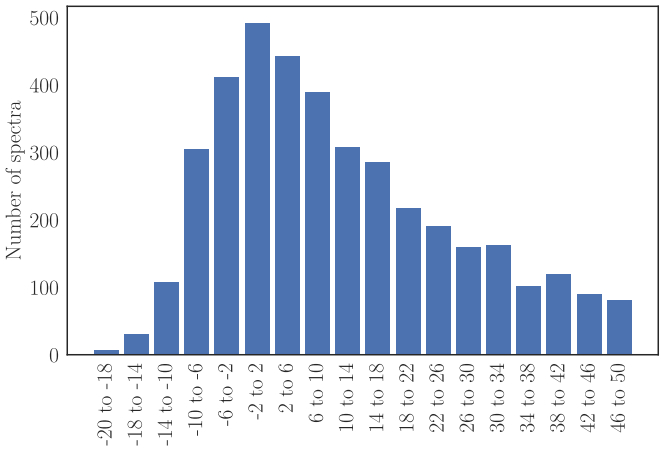

Figure 2. Distribution of spectra in the final data set across the different age bins.

terms of standard machine learning problems, this is a relatively small data set. In order to combat both of these issues, we have made use of an oversampling technique to greatly diminish the effect of these problems. The idea of oversampling is to repeat each spectrum in lowly populated bins until all classification bins have the same number of spectra. As an example, if a bin in the training set had 250 spectra in it, we would repeat each of those spectra 4 times, and if a bin had five spectra, we repeat each of those spectra 200 times, until all bins have an equal amount of 1000 spectra. However, instead of simply repeating spectra (which adds no information to a neural network), we perform three data augmentation techniques that can magnify the size of our training set by over 1000 times. The following data augmentation steps are used:

Adding noise: The easiest thing to do is to simply add random amounts of Gaussian noise to each spectrum while oversampling. In our case, we add Gaussian noise $\mathcal{N}(\mu, \sigma^2)$ with mean $\mu = 0$ and $\sigma = 0.05(f_{\max}–f_{\min})$, where $f_{\max}$ and $f_{\min}$ are the maximum and minimum flux values in the spectrum, respectively.

Adding host galaxy spectra: Second, so that we can also distinguish an SN spectrum that is contaminated by its host galaxy, we also add on varying amounts of host galaxy spectra. For each spectrum in the initial training set, we add a host galaxy spectrum in varying proportions from 1% to 99% and also make use of 11 different host types: E, S0, Sa, Sb, Sc, SB1, SB2, SB3, SB4, SB5, and SB6, which are taken from the BSNIP and `Superfit` data sets.

Cropping: We also crop each spectrum by varying amounts, such that instead of just training on an entire spectrum, we train on different wavelength segments of each spectrum. That is, we reduce the wavelength range of each spectrum by random amounts. As with all spectra that do not cover the full wavelength range used in our neural network, we set the points in the preprocessed and normalized spectra that do not have data to 0.5 (see Figure 3(d) for an example and Section 2.3 for more details).

Redshifting: Finally, for the unknown redshift models (see Section 4.1) we redshift each spectrum by a random amount from $z = 0$ to $z = 1$.

These processes increase the size of our training set considerably. Since we add on 11 different host galaxy spectra at over 10 different fractions, crop each spectrum at at least four different wavelength intervals, and add noise to each spectrum while oversampling by a minimum of 4 times (up to 1000 times depending on the number of spectra in the bin), we effectively increase our training set by at least $11 \times 10 \times 10 \times 4 \times 4 = 1760$ times the initial training set, but actually over around 100,000 times the initial data set size, given the amount of oversampling of lowly populated bins and random redshifting during training.

In this data augmentation process, we are enabling the neural network to find and train on the common features among the augmented spectra, allowing it to train only on the actual features that make up a spectrum instead of the noise, host light, or wavelength range of each spectrum. This significantly inhibits the imbalanced data set problem and allows the neural network to train on actual SN features of a particular classification bin rather than random distortions of a single spectrum.

Ultimately, this technique is very important and effective but cannot compete with actually having huge amounts of real observational data. In future, as more large-scale surveys work to increase the transient catalog, these CNN problems will be far more powerful than what can be made with current data sets.

Before augmentation, we split the total set of transients into two parts: 80% for the *training set* and 40% for the *testing set*. The *training set* is used to train the classifier to identify the correct SN class, while the *testing set* is used to test the performance of the classifier. We then apply the augmentation detailed previously to the training set only.

2.3. Preprocessing

Arguably one of the most important aspects in an effective learning algorithm is the quality of the training set. As such, a lot of the software effort in this project has been in ensuring that the data have been processed in a systematic and uniform way before we train the matching algorithm. In this section, we outline the processing techniques used to prepare the training set and the input spectra.

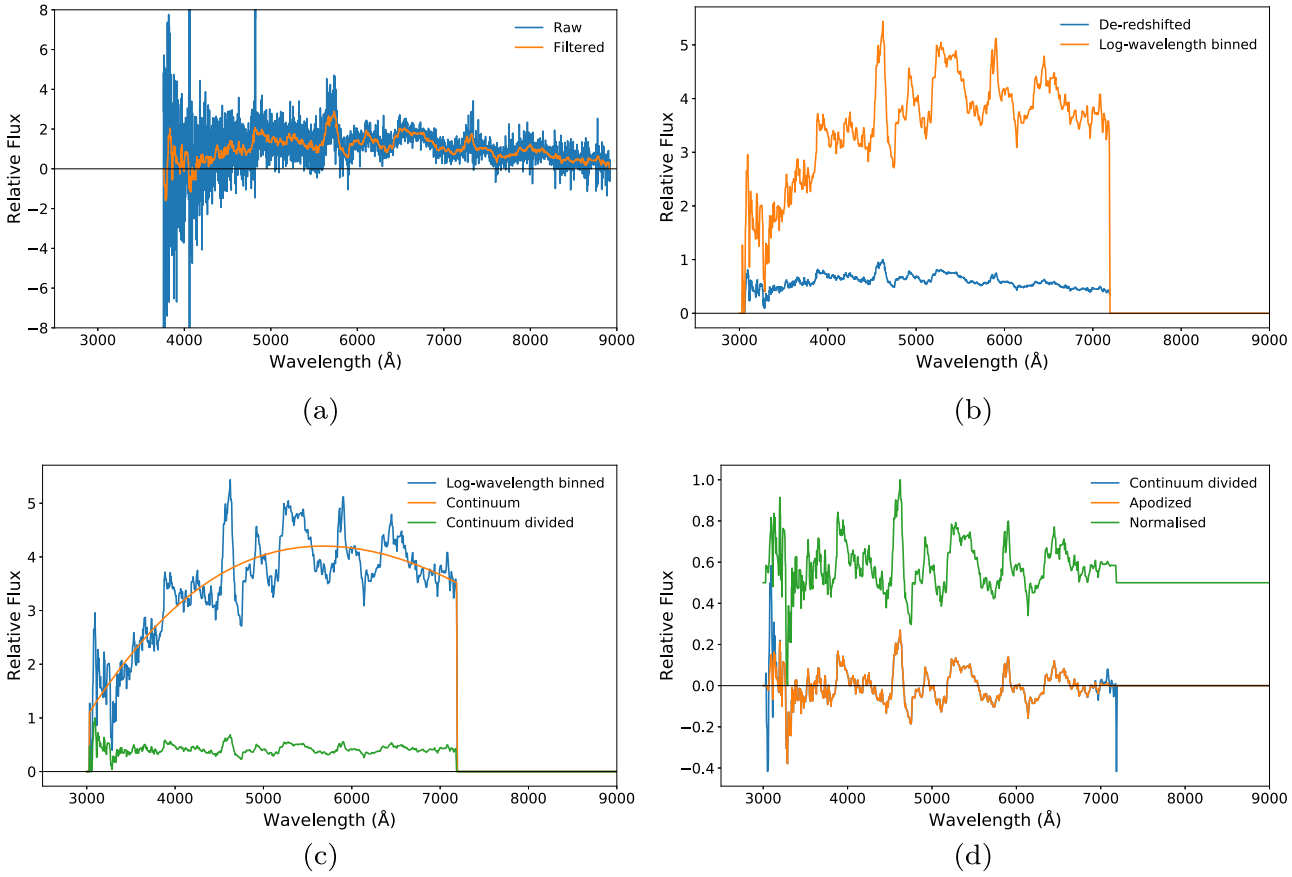

Figure 3. Spectral preprocessing steps before training using the SN Ia DES16C2ma spectrum as an example. (a) The blue line is the raw data spectrum, while the orange line shows the result after applying a low-pass median filter with a window size defined by Equation (3) and a smoothing factor of 5. (b) The smoothed spectrum is then de-redshifted to its rest frame based on the redshift obtained from its host lines from an external software. This step is not applied in DASH if the redshift-agnostic model is used. It is also binned into N_w points on a log wavelength scale. (c) The de-redshifted and smoothed spectrum is then binned onto a log wavelength scale as defined in Equation (4) (blue line). A 13-point cubic spline interpolation is used to model the continuum (orange line) before it is divided from the binned spectra to remove any spectral color information (green line). (d) The edge discontinuities on the previous spectrum are smoothed with a cosine taper (orange line). The flux is then normalized to values between 0 and 1 (green line).

Many of the previous classification and redshifting tools (including SNID, Blondin & Tonry 2007; MARZ, Hinton et al. 2016; and AUTOZ, Baldry et al. 2014) preprocess their spectra in a similar way before cross-correlation and template matching. These methods are loosely based on the algorithms discussed by Tonry & Davis (1979). We implement a very similar processing technique to that used by Blondin & Tonry (2007) in SNID. Our processing algorithm is applied to both the training set and any input spectrum. It consists of the following steps:

1. Low-pass median filtering: The first step to processing is to apply a low-pass median filter to each spectrum in order to remove high-frequency noise and cosmic rays. We scale the amount of smoothing based on the average wavelength spacing of the spectrum, defined as $\lambda_{\rm density}$ below:

$$\lambda_{\rm density} = (\lambda_{\rm max} - \lambda_{\rm min})/N, \quad (1)$$

where $\lambda_{\rm min}$ and $\lambda_{\rm max}$ are the minimum and maximum wavelengths of the spectrum, respectively, and N is the number of points in the spectrum. We also define the wavelength density of the final spectra after processing as

$$w_{\rm density} = (w_1 - w_0)/N_w. \quad (2)$$

The window size of the median filter is then defined as

$$\text{window\_size} = \frac{w_{\rm density}}{\lambda_{\rm density}} \times \text{smooth}, \quad (3)$$

where smooth is the user-defined amount to scale the amount of filtering. Most of the spectra used in the training set have been preprocessed and smoothed by SNID, and as such we do not add any further smoothing, and we set the window size to 1. Input spectra in DASH have a default smoothing factor of smooth = 6, but they can be altered by a user. An example of this filtering step is illustrated in Figure 3(a).
2. De-redshifting: The next stage involves de-redshifting the spectrum to its rest frame (illustrated in Figure 3(b)). For input spectra, this is an optional stage depending on which redshift model is used (see Section 4).
3. Log wavelength binning: In the third step, we bin the spectra onto a log wavelength scale with a fixed number of points (N_w) between w_0 and w_1. These parameters can be changed by a user who wishes to retrain the CNN model. However, the default parameters are $N_w = 1024$, $w_0 = 3500$ Å, and $w_1 = 10000$ Å, which covers the optical spectral range at which most SN events are observed and has enough points to recover both narrow and broad spectral features, while not including too many

points to be computationally expensive. These parameters were further selected to match the default parameter values of the SNID data, so that we could directly use these in our training set.

This step is important for a few reasons. First, it ensures that each spectrum is a vector of exactly the same length and at the same wavelengths so that vectors from different spectra can be easily compared and trained on. Second, it is consistent with the SNID data and can make redshifting less computationally expensive (Blondin & Tonry 2007). However, perhaps most important is that we can make use of CNN's natural position invariance (Duda et al. 2012) during classification. By using a log wavelength scale, changes in redshift now become linear translations, and so the CNN's natural affinity for being invariant to small linear translations can be employed to allow classifications to also be invariant to redshift.

The log wavelength binning process follows the same method outlined in Blondin & Tonry (2007); some of the key steps are shown here. First, the log wavelength axis, $w_{\log,n}$, is defined as

$$w_{\log,n} = w_0 \ln e^{n \times dw_{\log}}, \tag{4}$$

where n is the index of each point in the vector and runs from 0 to N_w, and

$$dw_{\log} = \ln(w_1/w_0)/N_w \tag{5}$$

is the size of a logarithmic wavelength bin. The binned wavelength can then be translated from the normal wavelength with the following relationship:

$$\text{binned\_wave} = A \ln w_{\log,n} + B, \tag{6}$$

where $A = N_w/\ln(w_1/w_0)$ and $B = -N_w \ln w_0/\ln(w_1/w_0)$. Using this method, the input and training spectra were binned onto this scale. The binned spectrum is illustrated as the orange line in Figure 3(b). The points in the spectrum that do not have data in the range w_0 to w_1 are set to zero.

4. Continuum modeling with spline interpolation: The fourth step in preparing the spectra involves dividing the continuum. For galaxy spectra, the continuum is well defined and is easily removed using a least-squares polynomial fit. In SN spectra, however, the apparent continuum is ill-defined owing to the domination of bound–bound transitions in the total opacity (Pinto & Eastman 2001). For this reason, a 13-point cubic spline interpolation is used to model the continuum. A total of 13 points was considered to be sufficient to interpolate the spectrum. This is illustrated as the orange line in Figure 3(c).
5. Continuum division: This continuum is then divided from the spectrum (blue line). This step removes any spectral color information (including flux miscalibrations) and enables the correlation to rely purely on the relative shape and strength of spectral features in each spectrum. It also has the advantage of diminishing the effect of extinction from the remaining spectra. According to Blondin & Tonry (2007), the loss of color information has very little impact on the redshift and age determination.
6. Apodizing the edges: While the discontinuities at each end of the spectrum are limited by the continuum division, further discontinuities are removed by apodizing the spectrum with a cosine bell in the final step of processing. This involves multiplying 5% of each end of the spectrum by a cosine, to remove sharp spikes. This is illustrated as the orange line in Figure 3(d). Finally, the spectrum is renormalized to positive values between 0 and 1 (green line), so that it is ready for training in the CNN. As neural networks require regularly sampled data in a fixed grid, we set the points in the spectrum that do not have data in the range w_0 to w_1 to 0.5.

We then define two important properties for each processed spectrum for the supervised deep learning approach: its label and image data. The image data are composed of the 1024-point vector that corresponds to the preprocessed normalized flux values. The labels correspond to one of the 306 different classification bins outlined in Section 2.2. We represent these labels as 306-point one-hot vectors where each entry represents a different classification bin so that matrix multiplication can be more easily used when training. The labeled and preprocessed data are then passed into the deep learning model for training.

3. Deep Learning

Deep learning is a branch of machine learning that has recently gained a lot of popularity for its success in a range of different applications, including image, speech, and language recognition. The age of big data and advancements in computer hardware have enabled neural networks to be effective at solving these more complicated problems in reasonable amounts of time.

3.1. Convolutional Neural Networks

Convolutional neural networks are one of the most popular deep learning architectures and have been very successful at benchmark image classification problems. We have employed this architecture by phrasing the spectral classification problem as a one-dimensional image classification problem, where fluxes correspond to pixel intensities. This enables us to use a very similar method to that which is used to solve the benchmark MNIST classification problem (Li 2012). We have developed the CNN with TensorFlow's Python library owing to its convenient high-level library that avoids low-level details. It makes use of a highly efficient C++ back end to do its computations (Abadi et al. 2016).

In a deep neural network, each layer is in the form of a set of nodes or neurons that represent the data. In DASH, the first input layer is made up of 1024 neurons representing the fluxes of an input spectrum. Additional layers of neurons above the original input signal are built to ensure that each new layer captures a more abstract representation of the original input layer. Each new hidden layer identifies new features by forming nonlinear combinations of the previous layer (Cybenko 1989; Hinton & Salakhutdinov 2006). For example, the hidden layers in DASH represent abstract constructions of the input flux vector. The final output layer will then simply represent 306 different neurons corresponding to the 306 different classification bins of SN types and ages.

The output, $\hat{y}_i$, of each neuron in a neural network layer can be expressed as the weighted sum of the connections from the previous layer:

$$\hat{y}_i = \sum_{j=1}^{n} W_{i,j} x_j + b_i, \tag{7}$$

where x_j are the different inputs to each neuron from the previous layer, $W_{i,j}$ are the weights of the corresponding inputs,

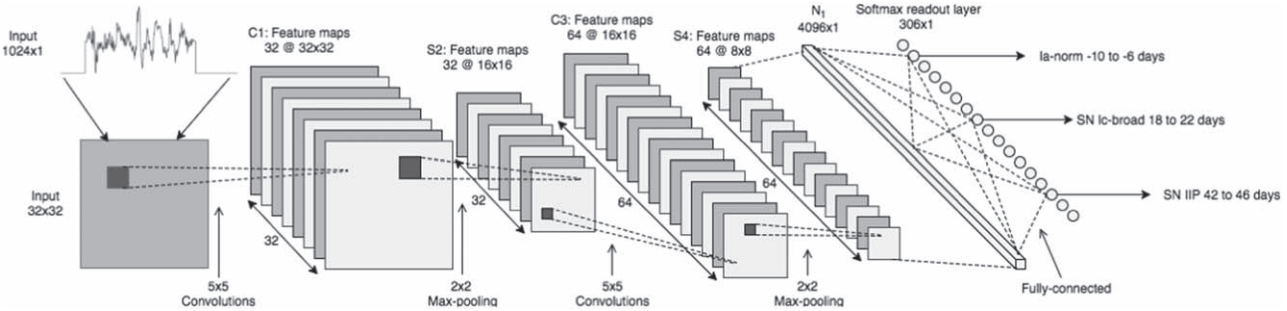

Figure 4. Visual representation of the multilayer convolutional neural network used in DASH. The 1024-point input flux, which has been processed following the method outlined in Figure 3, is reshaped into a 32 × 32 grid. The first convolutional layer computes 32 features for each 5 × 5 patch on the input. These 32 images are then subsampled using a standard max-pooling layer over 2 × 2 patches of each image, reducing the image sizes to 16 × 16. A second layer of convolution with 64 features for each 5 × 5 patch is applied to the previous layer before a 2 × 2 max-pooling layer is used to subsample the image size down to 8 × 8. The 64 images representing a 64 × 8 × 8 tensor are then flattened down to a 4096-point vector. A fully connected layer with 1024 neurons to allow processing on the entire image is added. Similar to the convolutional layers, weights and biases are computed before a readout and softmax regression layer are added to identify the best-matching classifications of the model. This final layer is a 306-point vector, with a score for each SN type and age bin. Three example classification bins have been listed on the right.

b_i is a bias that is added to allow some points in the vector to be more independent of the connections, j is an integer running from 1 to the number of connected neurons in a particular layer to sum over the connections from the previous layer, and i is an integer running from 1 to the number of neurons in the next layer. In the simple case, where we simply have a single layered dense neural network, x is simply the input flux, i runs from 1 to 1024 across the length of the input flux vector, and j runs from 1 to 306 across the number of classification bins. The weights and biases are free variables that are computed by TensorFlow during the training process.

In the final output layer, the values of $\hat{y}$ represent the "evidence" tallies for each classification bin. In order to be able to assign probabilities to each of the classification bins, we make use of a softmax regression model in the final layer. The softmax regression probabilities, y, are calculated by applying a softmax function on the evidence,

$$y = \mathrm{softmax}(\hat{y}), \tag{8}$$

where the softmax activation function is defined as

$$\mathrm{softmax}(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}. \tag{9}$$

This function generalizes a logistic regression to the case where it can handle multiple classes. It effectively normalizes the output layer of neurons so that the total probabilities of all classification bins sum to 1. These softmax probabilities are important in DASH, as they are used to rank the best-matching classification bins. It is important to note that these probabilities only provide the relative probability of a particular classification bin when compared to the other 306 different SN types and ages.

Before the training process can begin, we need to specify a loss function that indicates how accurately the model's prediction matches the true class for each input spectrum. We define the loss function to be the cross-entropy, $H_Y(y)$, between the actual classification bin, Y, and the model's prediction, y, as

$$H_Y(y) = -\sum_{i=1}^{306} Y_i \log(y_i). \tag{10}$$

Here Y is the label of the data that are made up of a 306-point one-hot vector with zeros in all entries except for one, which has a 1 to indicate the true classification bin. On the other hand, y is a 306-point vector where the sum of all entries is 1, and ideally for a good model the entry with the highest probability would be the same bin as the entry with a 1 in Y. Hence, the cross-entropy measures how inefficient the predictions are compared to the truth. We minimize the cross-entropy using a common but sophisticated gradient descent optimizer called the Adam optimizer (Kingma & Ba 2014). We feed in our training set defined in Section 2 in small batches and train the neural network such that the values for the weights and biases in each layer are computed to optimize the model.

Overall, the neural network model consists of six different layers: two convolutional layers with two max-pooling layers between them, one fully connected layer, and a readout layer before the softmax regression as illustrated in Figure 4. Each convolutional and fully connected layer has weights and biases that are initialized with a small amount of noise to avoid symmetry breaking and zero-gradients and a small positive bias to avoid "dead neurons," respectively. These layers use rectified linear units (Nair & Hinton 2010) as the activation function for each of the neurons in the layers. The max-pooling layers basically just subsample the input flux in a nonlinear fashion so as to reduce the computational complexity (Boureau et al. 2010; Aniyan & Thorat 2017). Following the fully connected layer, we implement dropout regularization to reduce overfitting during training. Effectively, this means that the neurons that have very small weight values, and hence do not strongly interact with other neurons, are discarded from the network iteratively during training.

4. Trained Models

4.1. Models

Using the machine learning architecture defined in Section 3, we have trained four different models that are available in the DASH release. All of these models use the same data set (described in Section 2) and follow the same data augmentation approaches outlined in Section 2.2.1, whereby various amounts of host galaxy light are added to the data. However, they differ in whether they classify into these hosts and on whether they calculate the redshift. They are listed as follows:

1. Known redshift, SN-only classification.
2. Unknown redshift, SN-only classification.
3. Known redshift, SN+host classification.
4. Unknown redshift, SN+host classification.

In the majority of SN spectra, the redshift can be accurately predetermined from host galaxy features using effective redshifting tools such as MARZ[12] (Hinton et al. 2016). As such, the "Known redshift, SN-only" model has been designed with the same CNN architecture illustrated in Figure 4 and ensures that each spectrum in the training set has been de-redshifted to its rest frame ($z = 0$). During classification, the redshift must be input as a prior by the user so that the input spectrum can also be de-redshifted to its rest frame.

However, in some SN spectra, the host galaxy is too faint compared to the SN spectrum, and hence the redshift cannot be easily determined from standard redshifting tools. For these cases, we have developed models that can classify SNe independent of the redshift and hence do not require a redshift prior. The "Unknown redshift, SN-only" model uses the same architecture as the "Known redshift, SN-only" model but differs by adding an extra data augmentation step (see Section 2.2.1), which involves iteratively redshifting each spectrum by varying amounts before training. This enables the trained model to learn the features of spectra independent of their redshift and hence be able to identify the classification bin regardless of whether the input spectrum is in its rest frame.

Once the best-matching classification bins have been identified, we determine the redshift of each of the top-ranking classification bins by making use of a cross-correlation technique with the input and the training data from the classification bin. This redshifting method is described in Section 4.2.

The SN+host classification models are designed with nearly the same architecture as the SN-only models, respectively. However, instead of just classifying into the 306 classification bins made up of SN type and age, we add an extra dimension with 11 host galaxies, making a total of (11×306) 3366 classification bins. In each of these bins we add varying proportions of a particular host galaxy spectrum. The 11 host galaxy types we used are taken from the SNID and BSNIP databases and follow the Hubble diagram naming convention, listed as follows: E, S0, Sa, Sb, Sc, SB1, SB2, SB3, SB4, SB5, SB6. The CNN then trains based on the presence of a combined SN and host galaxy.

4.2. Redshifting Methods

In the second and fourth models, we iteratively redshift each spectrum in the training set by varying amounts between $z = 0$ and $z = 1$ before it is trained with the neural network, hence enabling the model to learn features and classify spectra irrespective of redshift. The log wavelength scale means that redshifts are now linear translations and hence help us to employ the CNN's natural position invariance (Duda et al. 2012). Once the model has determined a best-matching classification bin, we calculate the redshift using a very similar cross-correlation technique to that used in SNID as defined by Blondin & Tonry (2007) and Tonry & Davis (1979). The preprocessed input spectrum, $s(n)$, is cross-correlated ($\star$) with each training set spectrum, $t(n)$, in the classification bin as follows:

$$c(n) = s(n) \star t(n) = \mathcal{F}(S(k)T(k)), \tag{11}$$

where n represents the log wavelength indexes, $c(n)$ is the cross-correlation function, $S(k)$ and $T(k)$ represent the fast Fourier transform of the input spectrum and a training set spectrum, respectively, and $\mathcal{F}$ is the fast Fourier transform function that enables us to calculate the cross-correlation. An example cross-correlation function of the spectrum used in Figure 3 and a spectrum from the training set is illustrated in Figure 5.

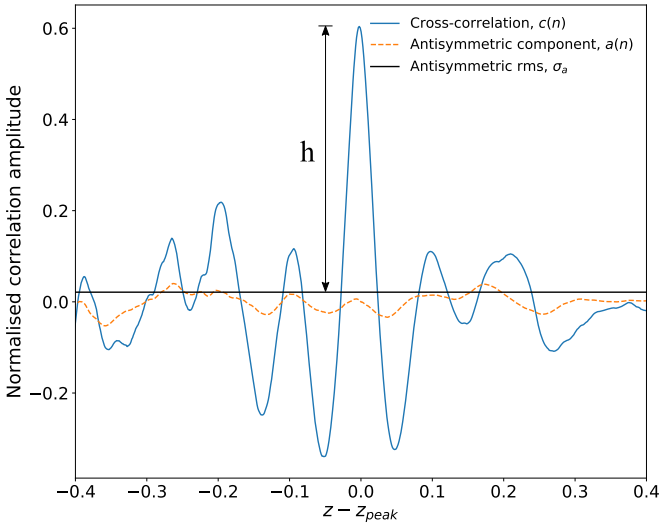

Figure 5. Example cross-correlation function of the DES16C2ma spectrum (Figure 3) with the best-matching spectrum from the training set determined by DASH. The position of the highest peak is used to determine the redshift with Equation (12). The antisymmetric component, $a(n)$, defined in Equation (14) is shown as the orange dashed line, and the rms of this is illustrated as the horizontal black line. The height of the correlation peak, h, is the difference between the highest value and the antisymmetric rms, σ_a.

Since the spectra have been processed onto a log wavelength scale defined in Section 2.3, the position of the peak cross-correlation score enables the redshift to be computed as

$$z = e^{\delta \times dw_{\log}}, \tag{12}$$

where $dw_{\log}$ was defined in Equation (5) and δ is the index value of the peak cross-correlation score in the range $-N_w/2 < \delta < N_w/2$. We calculate the redshift from the cross-correlation of the input spectrum with each training set spectrum in a particular classification bin and take the median value of all the redshifts.

The error in the calculated redshift is determined by simply calculating the standard deviation of the redshifts in a particular classification bin.

4.3. False-positive Rejection

As outlined in Section 3.1, the ranking system used by DASH only provides a relative measure of how closely an input spectrum matches a particular classification bin compared to all other classification bins. If an input spectrum happens to be a weird spectrum, then this ranking system will still choose the closest match, which may lead to false-positive classifications. To account for such cases, we have made use of two independent measures to flag potential misclassifications. The first rejection test makes use of a similar measure to that used in

[12] http://samreay.github.io/Marz/

SNID called the *rlap* score, and the second test compares the top-ranking DASH classifications to ascertain whether the matches are consistent with each other. This provides two independent warnings to a user: a "*low rlap warning*," and an "*inconsistent classification warning*." These are seen as a "reliability" label in the DASH interface, which act to inform a user that the automatic classification requires closer human inspection.

4.3.1. *Low* rlap *Warning*

In SNID, the *rlap* scores act as the primary method of comparing an input spectrum to each of the spectra in the training set: the training set spectrum with the highest *rlap* score is considered the best-matching spectrum.

Tonry & Davis (1979) first introduced the cross-correlation height-to-noise ratio, r, to quantify the significance of a cross-correlation peak. It is defined as

$$r = \frac{h}{\sqrt{2}\,\sigma_a}, \tag{13}$$

where h is the height of the cross-correlation shown in Figure 5 and $\sqrt{2}\,\sigma_a$ is the rms of the antisymmetric component of $c(n)$. The antisymmetric component is calculated by assuming that $c(n)$ is the sum of an autocorrelation of the training set spectrum, $t(n)$ with its shifted spectrum $t(n-\delta)$, and a random function, $a(n)$, that distorts the correlation peak (Tonry & Davis 1979):

$$c(n) = t(n) \star t(n-\delta) + a(n). \tag{14}$$

The autocorrelation term will give a peak correlation at the exact redshift given by the shift, δ, in logarithmic wavelength units, and it will be symmetrical about $n = \delta$. Assuming that the symmetric and antisymmetric parts of $a(n)$ have approximately the same amplitude and are uncorrelated, the rms of $a(n)$ is $\sqrt{2}$ times the rms of its antisymmetric component (Blondin & Tonry 2007).

While the r score alone is a sufficient measure of the similarity of two spectra if both the training set spectrum and input spectra cover a wide wavelength range, it provides a poor measure if the two spectra do not significantly overlap each other in their rest frame. This overlap can be quantified as

$$\mathrm{lap} = \ln \frac{w_a}{w_b}, \tag{15}$$

where w_a and w_b are the maximum and minimum wavelengths at which both spectra overlap each other, respectively. Combining the two scores in the product $\mathrm{rlap} = r \times lap$ provides a measurement of the similarity between two spectra.

After cross-correlating an input spectrum with each training set spectrum in the best-matching classification bin, we calculate the average value from each of these *rlap* scores. If the average *rlap* score is small (defined as $\mathrm{rlap} < 6$), then we output a low-reliability flag to the user to act as a warning that the automatic classification may not be accurate. This enables a user to more closely inspect the spectra with a low *rlap* warning. We note that the *rlap* scores used in DASH cannot be directly compared with the scores used in SNID.

4.3.2. *Inconsistent Classification Warning*

The second measure of warning a user about a potential misclassification is to compare the top-ranking classifications provided by DASH. If the top matches are not in neighboring classification bins, such that the broad SN type and the SN age are distinctly different from each other in the top few matches, then we list the classification with a warning label. More specifically, we check that the top two matches are the same broad SN type, and we also check whether the age bins of the top matches are neighboring each other (i.e., an example of neighboring bins would be "2–6 days" and "6–10 days"). If either of these checks fails, then we output a warning to signify that there may be a misclassification.

On the other hand, if the top matching classifications are in agreement, such that they represent the same type of SN and neighboring age bins, then we can actually combine the softmax probabilities together to provide a higher level of certainty on the classification. For example, in Figure 8 we can combine the top few classifications, as they are in neighboring bins, and output the combined probability, as is illustrated in the top right of the figure.

5. Performance

In this section we detail the performance of the main Model 1 (see Section 4.1) released in DASH. The matching algorithms are first validated against the testing set, and then it is tested against recent data taken from 3 yr of ATels (Astronomical Telegrams) released by OzDES (Bassett et al. 2015; Davis et al. 2015; Glazebrook et al. 2015; Lewis et al. 2015; Pan et al. 2015; Smith et al. 2015; Tucker et al. 2015; Yuan et al. 2015b; Hoormann et al. 2016; King et al. 2016; Moller et al. 2016; Mudd et al. 2016; O'Neill et al. 2016a, 2016b; Sommer et al. 2016; Muthukrishna et al. 2017; Sharp et al. 2017; Calcino et al. 2018a, 2018b; Macaulay et al. 2018).

5.1. *Testing Set*

From the total number of spectra described in Section 2.1, initially 80% was used for training the deep learning algorithm, and 20% was left for evaluating the matching performance. Once we were confident that the algorithm was effective, we retrained it using 100% of the data before testing its performance on the OzDES ATels. While it is generally not good practice to apply a model that has not been validated, we decided that in order to be able to classify into classes that are not well represented in the training set, it would be more beneficial to use as much data as we had available. We tested both the validated model (using just 80% of the data) and the unvalidated model (using all the available data) on the OzDES spectra and found that while the difference was marginal, the model using all the data produced results that more closely matched the OzDES ATels.

The normalized confusion matrix illustrating the classification performance on the validation set is illustrated in Figure 6.

The predicted classes are mostly consistent with the true classes, with most misclassifications occurring within the same broad SN type. For example, the SN Ia-91T misclassifications were all Ia-norm SNe, and all Ib-pec SNe were misclassified as SNe IIb. Similarly, there were some SN Ib and SN Ic misclassifications.

5.2. *OzDES ATels*

To give an indication of how DASH will perform on noisy host-contaminated spectra from large surveys based on fiber optics instead of just long-slit spectroscopy, we collected

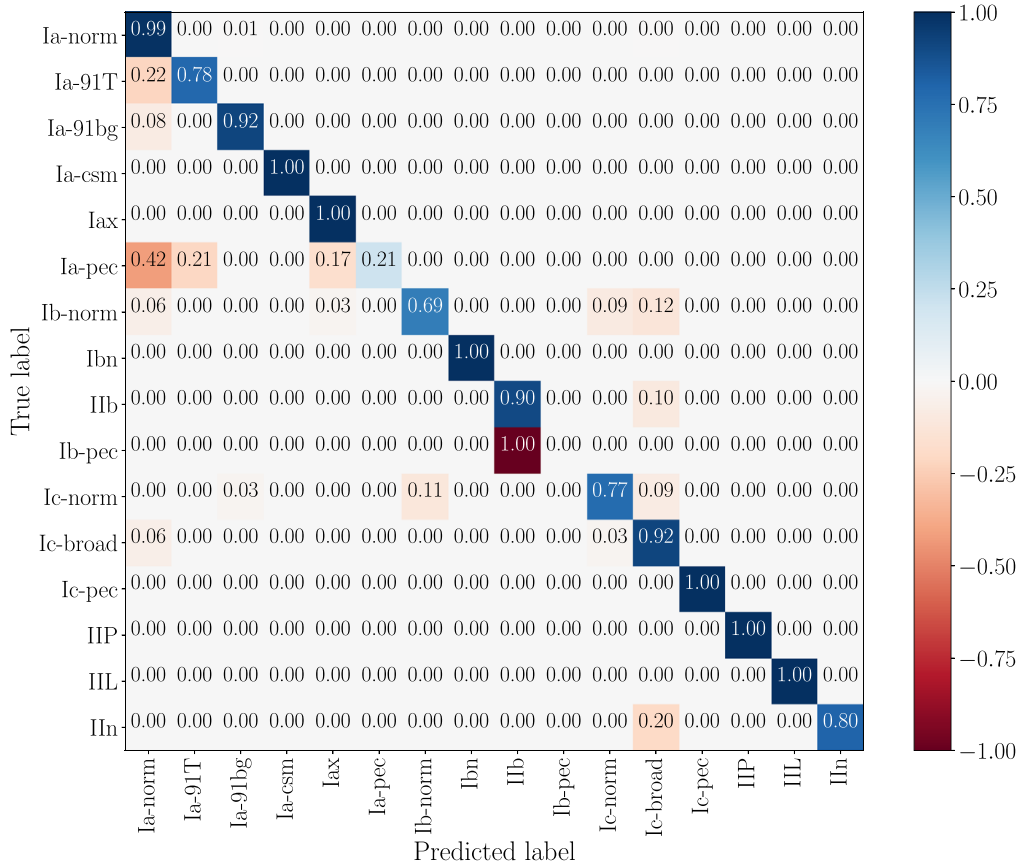

Figure 6. Normalized confusion matrix of the classifier trained on 80% of the data outlined in Section 2.1 and tested on the remaining 20% of spectra. The color bar and value in each cell indicate the fraction of each *True label* that was classified as the *Predicted Label*. The negative color bar values indicate misclassifications, while positive corresponds to correct classifications.

spectra that have been identified in all OzDES ATels from 2015 to 2017 and have compared whether DASH matches these classifications. This is listed in Table 2 in Appendix C. In the OzDES ATELs, objects are often not classified as precisely as DASH, whereby the age of the SN is not well constrained, and the SN may be listed with just a broad type without a specific subtype, and it may also often be listed with a trailing question mark to indicate that the classifiers were not confident on the classification. Moreover, it should be noted that these classifications were obtained not only by using data from the spectra but also by making use of the light-curve information. To this end, the classifications were completed by two or three experienced astronomers with the help of Superfit and SNID but were not autonomously classified like the DASH classifications.

DASH is able to provide a much more specific classification with the age and subtype constrained with useful probabilities to indicate the confidence of the fit. As a caveat, we note that objects flagged as *Reliable* should be considered strong classifications even if they have low probabilities because we can sum the probabilities of the next few similar classifications (see Section 4.3.2).

Furthermore, the speed of classification of DASH is significantly better than previous classification tools, whereby we were able to autonomously classify all 212 spectra in under 20 s, as opposed to the several days to weeks taken to originally classify the objects.

DASH was able to classify the entire set of OzDES spectra completely autonomously without any human visual inspection. It matched the ATel classification for 93% of the spectra, correctly classifying 197 out of the 212 SNe. These are listed in Table 2 in Appendix C and summarized in Table 1. OzDES is primarily a cosmological survey and thus is biased toward following up SNe Ia. The OzDES ATels are dominated by SNe Ia. Only a small fraction of SNe II are included in the ATels (perhaps because these can often be identified by the presence of hydrogen emission lines), and only three SNe Ib or SNe Ic have been included.

All but three of the mismatches were either flagged by the false-positive rejection scheme as *Unreliable* (indicating that the classification should be further checked by a human) or classified as a "Ic-broad." In general, we consider that

Table 1
Distribution of the 212 OzDES ATel Classifications Released between 2015 and 2017

ATel Class	No. of SNe	No. of DASH Matches
Ia	129	127
Ia?	43	34
II	28	25
II?	9	7
Ibc	1	1
Ibc?	2	2

Note. The "?" next to the SN type was used in the ATel classifications to indicate that the authors of the ATel were not confident in their classification. The third column lists how many of the objects in each class were also correctly classified by DASH.

classifications into the Ic-broad class are usually highly host-contaminated spectra and are not usually actually Type Ic-broad SNe. Two of the other three misclassfiications were typed as "SN Ia?," indicating that the ATel classifications were uncertain. The accuracy of the classifications, coupled with the false-positive rejection scheme, ultimately enables astronomers to only need to look at a very small subset of the entire testing set, with most spectra being classified autonomously.

5.3. Comparison to Previous Software

Overall, DASH is a more effective classification tool than previous tools for four important reasons: speed, accuracy, classification specificity, and its installation and ease of use.

The main improvement of DASH over current tools is its significant speed increase. The primary reason for the increase in speed is that machine learning does not iteratively compare with individual spectra, but instead classifies based on features in the spectrum. Thus, unlike SNID and Superfit, which increase their computation time linearly with the number of spectra in the training data, DASH is able to separate the training and testing stages. The classification of a single SN takes only a few seconds in DASH but can take several tens of minutes in Superfit. Moreover, while SNID is already a fast program, DASH is even faster, and this is particularly true when classifying several spectra at once. By making use of the DASH library functions, a user is able to classify hundreds or thousands of objects within seconds.

Unlike any other similar software, DASH does not iteratively search through and compare an input spectrum to each training set spectrum. Instead, it learns from the aggregate set of SNe in a particular classification bin and trains on the specific features that make up an SN type using a CNN. The advantage of this is that a classification is always made based on the entire set of spectra within a particular classification bin, rather than a single spectrum. This reduces the impact of spectra with incorrect classifications or unrepresentative spectra.

Finally, we have made the installation and usage very simple. It can be installed without having to worry about dependencies by making use of the Python Packaging Index. It also enables the simple classification of hundreds of spectra with just two lines of code (see Appendix B.3).

Nonetheless, software like Superfit and SNID still provide independent classification measures, and if used in conjunction with DASH, a robust classification scheme can be achieved.

6. Conclusions

We have developed a novel classification tool by using a contemporary CNN with advanced machine learning techniques. We have diverged from all similar tools that employ either a cross-correlation or chi-squared template-matching algorithm. By doing so, we have improved on previous work to enable DASH to be orders of magnitude faster than previous tools, autonomous, more accurate and precise in its classification, and much easier to install and use.

We have collated 4831 SN spectra from the CfA Supernova Program, BSNIP, and the stripped-envelope collection from Liu and Modjaz. Using this as a training set, we have validated the performance of our classifier on 3 yr of ATels from OzDES. The results indicate that DASH is well suited to classify the large number of spectra soon to be observed by upcoming large-scale spectroscopic surveys, such as DESI (DESI Collaboration et al. 2016) and 4MOST (de Jong et al. 2019).

Furthermore, these surveys will have less biased and much more complete samples of SNe that are better able to capture the diversity in the non-SN Ia populations. To improve the classification performance of DASH further, we can add this larger and more diverse range of SNe to our training set. Unlike previous classification tools, increasing the size of the training set does not decrease the classification time. However, the training of the classifier can be computationally expensive, and thus it will be most suitable to retrain DASH whenever more spectra that encompass a significantly deeper and wider range of spectral classes become available.

While this is an expansive set, as future surveys increase the SN catalog, we can increase the size of our training set to retrain and improve the performance of DASH even further.

A systematic preprocessing algorithm and data augmentation techniques have enabled us to train a robust learning algorithm. The training of four independent models has further allowed us to classify not only the SN type and age but also its host galaxy and redshift.

In a beta version of the DASH release, we have included extra superluminous SN (SLSN) classes as a new classification type, and we plan to release this in an upcoming version.

Moreover, while we have primarily developed this tool for SN classification, there is no significant reason why this approach cannot be extended to other types of spectra: from different types of stars, galaxies, or active galactic nuclei in the future.

We have publicly released the software with a graphical interface and a python library available on pip and GitHub, and it has already been used in several published SN classifications. Ultimately, the speed, accuracy, user-friendliness, and versatility of DASH present an advancement to existing spectral classification tools. As such, DASH is a viable alternative or complementary spectral classifier for the transient community.

D.M. was supported by an Australian Government Research Training Program (RTP) Scholarship and the Australian Research Council Centre for All-Sky Astrophysics (CAASTRO), through project No. CE110001020. The models were trained with the Obelix supercomputer from the School of Mathematics and Physics at the University of Queensland and the servers at the Research School of Astronomy and Astrophysics at the Australian National University.

Software: AstroPy (Astropy Collaboration et al. 2013), TensorFlow (Abadi et al. 2016), NumPy (van der Walt et al. 2011), SciPy (Jones et al. 2001), Qt.

Appendix A
Data Distribution

The final data set distribution in each type and age classification bin is illustrated in Figure 7.

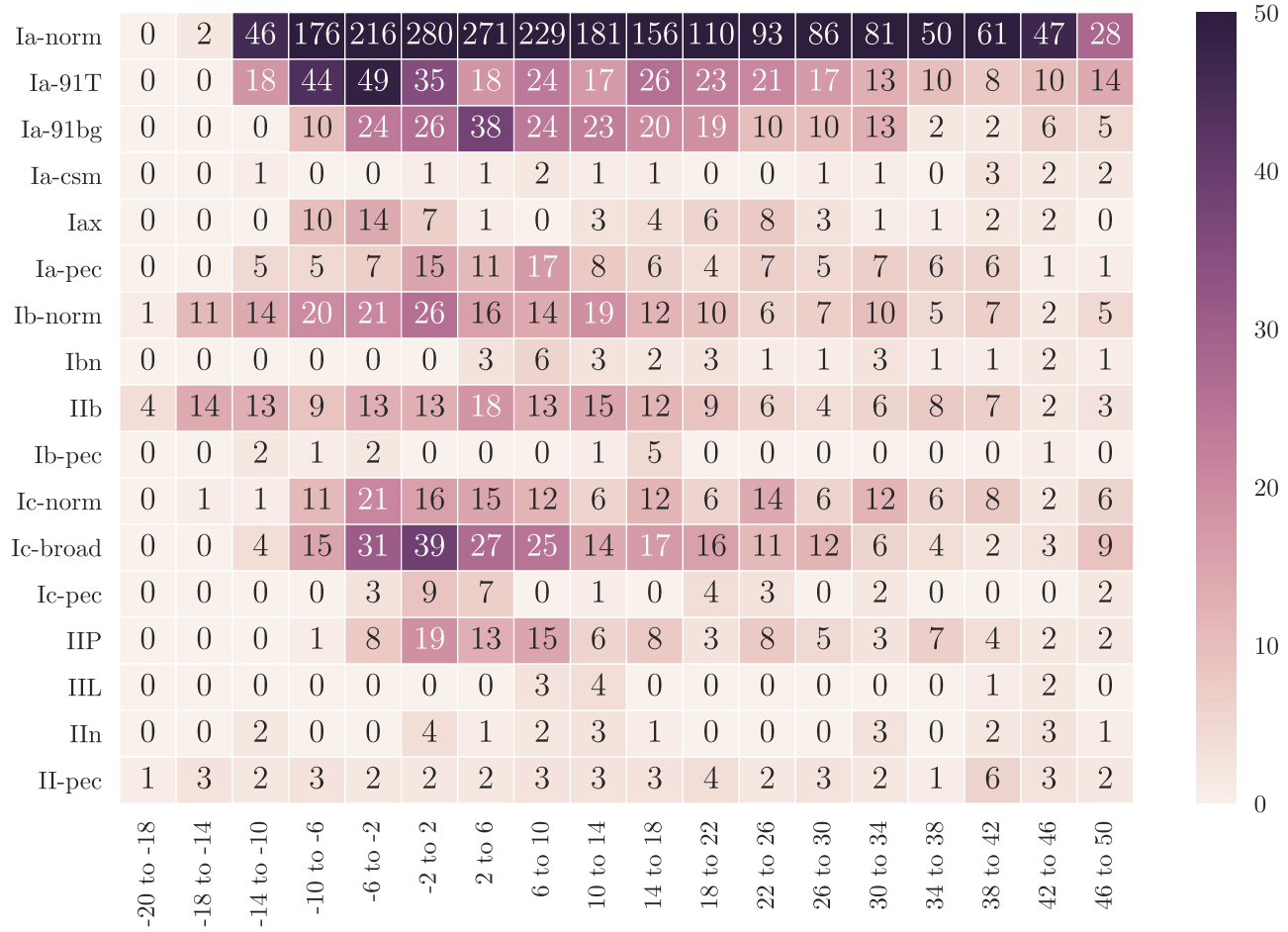

Figure 7. Distribution of the final data set used to train the machine learning model. The numbers of spectra for each subtype (rows) and each corresponding age in days since maximum (columns) are listed. The color bar ranges from 0 to 50 spectra.

Appendix B
Usage

DASH is intended to be an easy-to-use SN classification tool. It has the functionality to quickly classify a single spectrum, but its main advantage over existing tools lies in its ability to automatically classify hundreds or thousands of objects in just a few seconds. As such, it is intended to be used for large-scale transient surveys, and it is currently being used in the Australian sector of the Dark Energy Survey (OzDES).

B.1. Platform

We developed DASH (Deep Automatic Supernova and Host classifier) as an offline, cross-platform, and standalone program based in Python. It has been tested to effectively run on most Mac, Linux, and Windows distributions, with stringent testing on Mac Sierra and Ubuntu. We have ensured that the installation process is extremely simple and does not require the messiness of worrying about installing dependencies. The easiest way to install DASH is to run `pip install astrodash-upgrade` in the command line, which will automatically install nearly every dependency. This simplicity in installation, and the fact that it uses Python, which is currently the most popular programming language among astronomers (Momcheva & Tollerud 2015), is a huge advantage compared to previous SN classification tools.

There are six Python-based dependencies used in DASH, which are all automatically updated and installed with pip. We make significant use of Google Brain's new TensorFlow Python library (Abadi et al. 2016) to develop the CNNs, but we also make considerable use of NumPy (van der Walt et al. 2011), SciPy (Jones et al. 2001), AstroPy (Astropy Collaboration et al. 2013), and Qt with PyQt and PyQt-graph for the design of the graphical user interface. The code base is open-source and publicly available on GitHub[13] and is well documented.[14]

B.2. Interfaces

Two different interfaces are available in the DASH package: a graphical interface and a Python library. We detail these in the following subsections.

B.2.1. Graphical User Interface

The graphical interface enables users to visually inspect the DASH classifications while being able to tune various parameters. It has been designed to be user-friendly, to be intuitive, and to contain minimal clutter as illustrated in the example screenshot in Figure 8. More detailed instructions on

[13] https://github.com/daniel-muthukrishna/DASH/
[14] https://astrodash.readthedocs.io

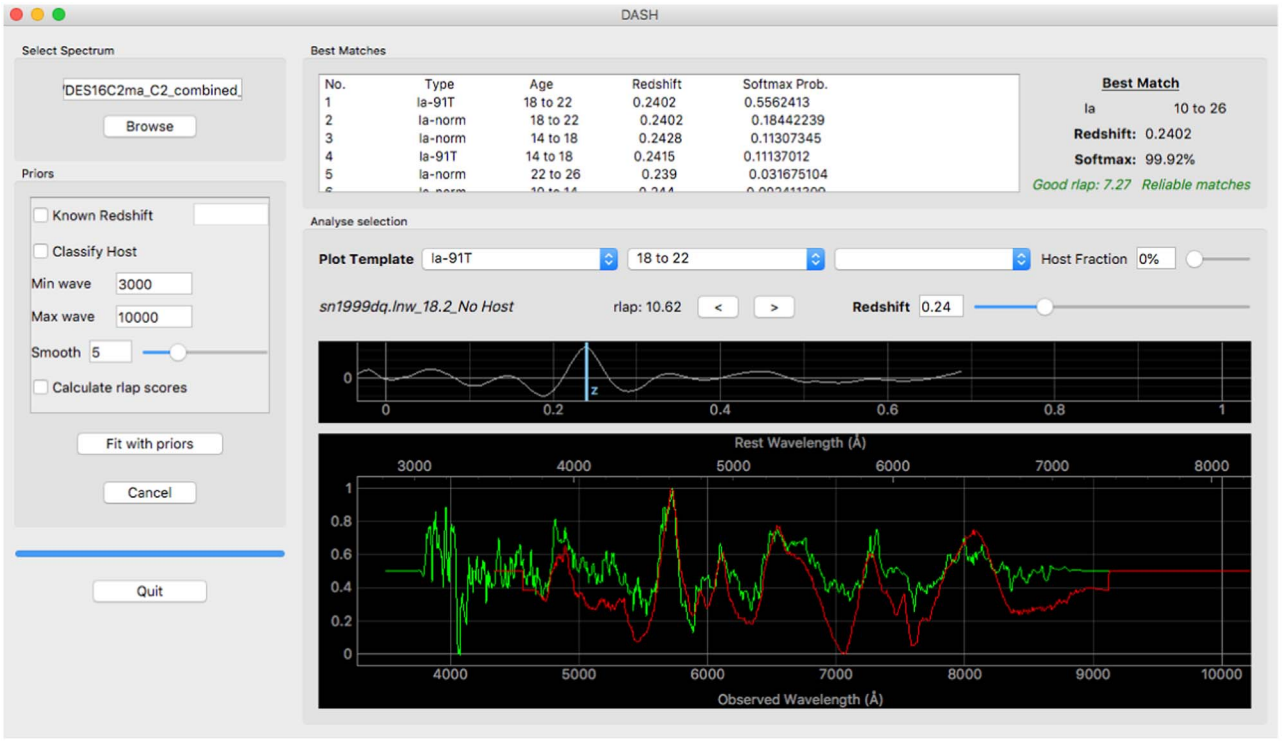

Figure 8. DASH graphical interface. An example classification of the OzDES DES16C2ma spectrum (as also illustrated in Figure 3) is shown. Using the agnostic redshift model, the software predicts that the input spectrum is a Type Ia-91T SN at 18–22 days past maximum with a 55.6% softmax regression confidence. The input spectrum is plotted in the bottom panel (green) against one of the example spectra from the training set (red). The cross-correlation is plotted in the smaller graph, with the predicted redshift being $z = 0.24$. The probabilities of the top six classifications can be combined, because they are all consistent with each other, to give a combined softmax regression confidence of 99.92% that the SN is an SN Ia between 10 and 26 days past maximum. Both the rlap and reliable matches flags (see Section 4.3) have passed and are written in green text to indicate that DASH is confident about the classification.

the usage have been provided in the online documentation, but we briefly outline the main components in the following.

On the left panel under the *Priors* header, the user can make a series of selections that alter the spectrum that is passed into the classification algorithm. The first selection enables the user to choose from one of the four models listed in Section 4.1 by selecting a combination of the two check boxes. Next, if the user wishes to avoid bad parts of the spectrum caused by excessive noise, dichroic jumps, or otherwise, the wavelength range of the input spectrum can be changed. In the case of very noisy spectra, a smoothing option, which applies a low-pass median filter at varying window sizes (as defined in Equation (3)), has also been provided. Finally, as well as the softmax probabilities used as a ranking system in DASH, users who are familiar with SNID may also choose to display *rlap* values that can act as a second measure of the quality of each classification (see Section 4.3.1). As cross-correlations are relatively slow, checking this box will significantly increase the total classification time. The listed *rlap* scores are calculated by averaging the scores from the cross-correlation of the input spectrum with each training set spectrum in a particular classification bin.

Once the priors have been chosen and the best-matching classifications have been filled, the right section of the interface will update to include a few important sections. On the bottom panel, we make use of PyQtgraph to plot the preprocessed input spectrum against different training set spectra. Above this, we also plot the cross-correlation function against redshift for each spectrum similarly to Figure 5. Under the *Best Matches* header, the top-ranking classification bins are shown with columns for the type, age, host galaxy, softmax probability, redshift, and *rlap* score. Depending on the *Priors* selections and the chosen model, only some of these headers will be displayed. On the top right, the best-matching classification will be listed by combining the top-ranked classifications (as detailed in Section 4.3.2). A flag indicating whether the match should be considered reliable or not is also shown based on the false-positive rejection tests outlined in Section 4.3. Under the *Analyze selection* header, a user can choose to plot a different classification bin, by selecting the type, age, and host of an SN. Clicking the arrows will switch between the different spectra in a particular classification bin. Finally, the user also has the option to change the fraction of host galaxy light displayed in the training set spectrum and the redshift of the input to visualize how this affects the spectral features. By default, a spectrum from the best-matching classification bin is plotted first.

B.3. Python Library

A Python library has also been developed so that several classifications can be made autonomously without the requirement of visual inspection. Classification of multiple spectra is very simple, requiring only a couple of lines of code:

```
import astrodash
classify = astrodash.Classify(filenames, redshifts)
print(classify.list_best_matches())
astrodash.plot_with_gui(indexToPlot = 0)
```

The only inputs required are a list of filenames containing the spectra that are to be classified and an optional list of corresponding known redshifts. More optional arguments selecting which model should be applied, the amount of smoothing, and whether rlap scores should be calculated can also be specified. The details of these optional arguments are outlined in the documentation. However, it should be noted that with just those two lines of code, several hundreds of spectra can be classified automatically and within just a few seconds or minutes, with the best matches being saved to a human-readable text file. The final line enables the first spectral file in the input list to be plotted and analyzed on the graphical interface.

B.4. Usage with Open Supernova Catalogs

DASH also interfaces with the online Open Supernova Catalog[15] (Guillochon et al. 2017). Changing the filename input in either interface with something in the format osc-name-age_index (e.g., osc-sn2002er-10) will download the spectrum from the OSC and classify it.

B.5. Development and Contribution

The DASH source code currently consists of several thousand lines of code across more than 30 Python files that are open-source and publicly hosted on a git repository on GitHub at https://github.com/daniel-muthukrishna/DASH.

GitHub provides issue tracking to keep track of open issues and feature requests. Users are encouraged to report bugs or issues and to request new useful features with this issue tracker. Moreover, this project has been developed in an object-oriented fashion, so that different code implementations can be relatively easily changed. One such example is the ability to easily change the deep learning architecture by just replacing one Python file. To this end, as more advanced neural network architectures become available, the learning algorithm can be improved or replaced.

Furthermore, as more SN spectra are observed by large-scale surveys, the training set should be updated. In fact, the more spectra that we can train the CNN with, the better the classification algorithm will become. To this end, if any users of the software would like to increase the size of the training set, they should contact us so that better models can be trained. Alternatively, simply updating the spectra in the training_set directory on GitHub and carefully running the "create_and_save_data_files.py" file will begin to train a new model. It should be noted that this training process may take a significant amount of computation time: usually on the order of hours depending on the computational resources available.

Finally, at the time of writing, the project has just the lead author as the sole active developer of the software. However, if users of the software would like to implement their own features that may be useful to others, we encourage them to contact us so that we can add them to the GitHub collaborators.

Appendix C
OzDES ATel Classification Comparison

In Table 2, we compare DASH classifications to OzDES ATels from 2015 to 2017. This is summarized in section 5.2.

Table 2

Classification of Supernovae Released in the Past 3 yr of ATels by OzDES (Bassett et al. 2015; Davis et al. 2015; Glazebrook et al. 2015; Lewis et al. 2015; Pan et al. 2015; Smith et al. 2015; Tucker et al. 2015; Yuan et al. 2015b; Hoormann et al. 2016; King et al. 2016; Moller et al. 2016; Mudd et al. 2016; O'Neill et al. 2016a, 2016b; Sommer et al. 2016; Muthukrishna et al. 2017; Sharp et al. 2017; Calcino et al. 2018a, 2018b; Macaulay et al. 2018)

Name	Redshift	ATel Classification	DASH			Match?
			Classification	Probability	Reliability	
DES15X3hp	0.236	Ia +3 weeks	Ia-norm (18 to 22)	0.835	Reliable	✓
DES15X3dyu	0.425	Ia max	Ia-norm (−10 to −6)	0.938	Reliable	✓
DES15X3auw	0.151	Ia +2 weeks	Ia-norm (10 to 14)	0.994	Reliable	✓
DES15X1bw	0.13	Ia +5 weeks	Ia-91T (42 to 46)	0.984	Reliable	✓
DES15E2nk	0.308	Ia +1 week	Ia-91T (6 to 10)	0.947	Reliable	✓
DES15E2atw	0.147	Ia +1 week	Ia-norm (18 to 22)	0.911	Reliable	✓
DES15C3fx	0.2	Ia +3 weeks	Ia-csm (10 to 14)	0.999	Reliable	✓
DES15S2dye	0.26	Ia +1 week	Ia-norm (−2 to 2)	0.953	Reliable	✓
DES15S1by	0.129	II post-max	Ic-broad (−6 to −2)	0.999	Unreliable	x
DES15C3edd	0.36	Ia max	Ia-91T (−2 to 2)	0.707	Reliable	✓
DES15C2dyj	0.395	Ia +1 week	Ia-norm (2 to 6)	0.955	Reliable	✓
DES15C2eaz	0.062	II max	IIP (2 to 6)	0.905	Reliable	✓
DES15C2aty	0.149	Ia +2 weeks	Ia-norm (18 to 22)	0.405	Reliable	✓
DES15C1atm	0.207	Ia +2 weeks	Ia-norm (18 to 22)	0.59	Reliable	✓
DES15X3kqv	0.142	Ia at max	Ia-norm (2 to 6)	0.996	Reliable	✓
DES15E1kwg	0.105	Ia at max	Ia-norm (−6 to −2)	0.709	Reliable	✓
DES15X1ith	0.16	Ia +1 week	Ia-norm (6 to 10)	0.905	Reliable	✓
DES15E1kvp	0.442	Ia At max	Ia-norm (2 to 6)	1.0	Reliable	✓
DES15C3efn	0.077	Ia +2 weeks	Ia-norm (14 to 18)	0.964	Reliable	✓
DES15X3iv	0.018	Ia +1 month	Ia-norm (46 to 50)	1.0	Reliable	✓

[15] https://sne.space/

Table 2
(Continued)

Name	Redshift	ATel Classification	DASH Classification	DASH Probability	DASH Reliability	Match?
DES15X3itc	0.338	Ia +2–5 days post-max	Ia-91T (−2 to 2)	1.0	Reliable	✓
DES15X3kxu	0.345	Ia At max	Ia-norm (−2 to 2)	0.94	Reliable	✓
DES15E1iuh	0.105	II at max	IIP (6 to 10)	0.918	Reliable	✓
DES15X2asq	0.28	Ia +7 days	Ia-91T (6 to 10)	0.655	Reliable	✓
DES15S2ar	0.247	Ia? +16 days	Ia-norm (10 to 14)	0.977	Reliable	✓
DES15C1eat	0.45	Ia? +7 days	Ia-norm (2 to 6)	0.612	Reliable	✓
DES15X1ebs	0.58	Ia? pre-max	Ic-broad (2 to 6)	0.605	Reliable	x
DES15C3bj	0.287	II? post-max	Ic-broad (−6 to −2)	0.807	Reliable	✓
DES15C3axd	0.42	Ia? max	Ia-pec (2 to 6)	0.997	Reliable	✓
DES15C1ebn	0.41	Ia? max	Ic-broad (2 to 6)	0.904	Reliable	✓
DES15C3lvt	0.4	Ia? post-max	Ic-broad (2 to 6)	0.596	Unreliable	x
DES15E2cwm	0.291	Ia? +10 days	Ia-91T (10 to 14)	0.964	Reliable	✓
DES15S2og	0.38	Ia? post-max	Ia-norm (6 to 10)	0.744	Reliable	✓
DES15S1cj	0.166	II? post-max	IIP (6 to 10)	0.832	Reliable	✓
DES15S1ebd	0.408	Ia? max	Ia-91bg (2 to 6)	0.504	Unreliable	✓
DES15S2dyb	0.56	Ia? +7 days	Ic-broad (−10 to −6)	0.514	Reliable	x
DES15X3flq	0.368	Ia? +10 days	Ia-91bg (−2 to 2)	0.5952	Unreliable	✓
DES15E2kvn	0.208	Ia? max	Ia-norm (−2 to 2)	0.971	Reliable	✓
DES15C2lpp	0.181	II? post-max	Ic-broad (2 to 6)	1.0	Unreliable	x
DES15C2lna	0.069	II post-max	IIn (10 to 14)	0.897	Unreliable	✓
DES15X2lxw	0.197	Ia +1 week	Ia-norm (2 to 6)	0.984	Reliable	✓
DES15X2mei	0.248	Ia max	Ia-norm (−6 to −2)	0.527	Reliable	✓
DES15X3lya	0.29	Ia max	Ia-norm (2 to 6)	0.709	Reliable	✓
DES15S1mjm	0.26	Ia? +1 week	Ia-norm (2 to 6)	0.574	Reliable	✓
DES15S1lyi	0.359	Ia? +3 weeks	Ia-pec (10 to 14)	0.602	Reliable	✓
DES15S2mpl	0.257	Ia +1 week	Ia-norm (2 to 6)	0.998	Reliable	✓
DES15S2mpg	0.186	Ia max	Ia-91T (2 to 6)	0.402	Reliable	✓
DES15E1neh	0.39	Ia? max	Ic-norm (2 to 6)	1.0	Unreliable	x
DES15X3mpq	0.188	II +1 month	IIP (10 to 14)	0.832	Unreliable	✓
DES15X2mpm	0.235	Ia +2 week	Ia-pec (10 to 14)	0.964	Reliable	✓
DES15X2mku	0.09	II +1 month	IIP (6 to 10)	0.999	Reliable	✓
DES15C1mvy	0.32	Ia max	Ia-91T (6 to 10)	0.98	Reliable	✓
DES15C1mqf	0.111	Ia +3 weeks	Ia-norm (14 to 18)	0.959	Reliable	✓
DES15X3naa	0.331	Ia −4 days	Ia-91T (6 to 10)	0.724	Reliable	✓
DES15X3nad	0.1	II max	IIP (2 to 6)	0.508	Unreliable	✓
DES15X2mzv	0.313	Ia max	Ia-91T (6 to 10)	0.929	Reliable	✓
DES15X2nkl	0.304	Ia max	Ia-norm (−2 to 2)	0.797	Reliable	✓
DES15C2njv	0.181	Ia max	Ia-norm (2 to 6)	0.559	Reliable	✓
DES15C1nfb	0.13	Ia max	Ia-norm (2 to 6)	0.57	Reliable	✓
DES15E2nlz	0.41	Ia −5 days	Ia-norm (−2 to 2)	0.789	Reliable	✓
DES15E1nei	0.313	Ia max	Ia-norm (2 to 6)	0.97	Reliable	✓
DES15C3mpk	0.182	Ia +10 days	Ia-91T (6 to 10)	1.0	Reliable	✓
DES15C2oxo	0.336	Ia? +9 days	Ia-norm (6 to 10)	0.69	Reliable	✓
DES15C3orz	0.18	Ia +5 days	Ia-91bg (10 to 14)	0.844	Reliable	✓
DES15C3olc	0.067	Ia +24 days	Ia-norm (18 to 22)	1.0	Reliable	✓
DES16X1ey	0.076	SN II post-max	IIn (6 to 10)	0.818	Unreliable	✓
DES16C3ea	0.217	SN Ia post-max	Ia-norm (14 to 18)	0.925	Unreliable	✓
DES16E1ah	0.149	SN II post-max	Ia-norm (26 to 30)	0.999	Reliable	x
DES16E1md	0.178	SN Ia max	Ia-91T (−2 to 2)	0.982	Reliable	✓
DES16C3bq	0.241	SN Ia max	Ia-norm (−2 to 2)	1.0	Reliable	✓
DES16C3fv	0.322	SN Ia −6 days	Ia-norm (−10 to −6)	0.546	Reliable	✓
DES16X3jj	0.238	SN II? post-max	IIL (10 to 14)	1.0	Unreliable	✓
DES16X3es	0.554	SN Ia? max	Ia-pec (2 to 6)	0.996	Unreliable	✓
DES16X3hj	0.308	SN Ia max	Ia-91T (−2 to 2)	0.899	Reliable	✓
DES16X3er	0.167	SN Ia +2 days	Ia-91T (−2 to 2)	1.0	Reliable	✓
DES16X3km	0.054	SN II post-max	IIP (6 to 10)	1.0	Reliable	✓
DES16E2dd	0.075	SN Ia +3 days	Ia-norm (2 to 6)	0.964	Reliable	✓
DES16E1de	0.292	SN Ia? +2 days	Ia-norm (−10 to −6)	0.568	Unreliable	✓
DES16X2auj	0.144	Ia max	Ia-norm (2 to 6)	0.94	Reliable	✓
DES16X1ge	0.25	Ia post-max	Ia-norm (22 to 26)	0.988	Reliable	✓
DES16C2ma	0.24	Ia post-max	Ia-norm (18 to 22)	0.658	Reliable	✓
DES16C2aiy	0.182	Ia post-max	Ia-norm (2 to 6)	0.993	Reliable	✓

Table 2
(Continued)

Name	Redshift	ATel	DASH			Match?
		Classification	Classification	Probability	Reliability	
DES16X3biz	0.24	Ia pre-max	Ia-norm (−6 to −2)	0.982	Reliable	✓
DES16X3aqd	0.033	IIP post-max	IIb (−14 to −10)	0.961	Reliable	✓
DES16E2aoh	0.403	Ia post-max	Ia-91T (−6 to −2)	0.864	Reliable	✓
DES16C3bq	0.237	Ia post-max	Ia-norm (2 to 6)	0.868	Reliable	✓
DES16E2bht	0.392	SN Ia +3 days	Ia-norm (−2 to 2)	0.988	Reliable	✓
DES16E2bkg	0.478	SN Ia max	Ia-norm (−2 to 2)	0.599	Reliable	✓
DES16X2crt	0.57	SN Ia? near-max	Ia-norm (−2 to 2)	0.622	Unreliable	✓
DES16E2cjg	0.48	SN Ia near-max	Ia-91T (−6 to −2)	0.563	Reliable	✓
DES16X3cpl	0.205	SN II? near-max	IIn (−2 to 2)	0.991	Unreliable	✓
DES16C3at	0.217	SN II +60 days	Ic-broad (2 to 6)	0.881	Reliable	x
DES16C1bnt	0.351	SN Ia +1 month	Ia-norm (22 to 26)	0.994	Reliable	✓
DES16C2cbv	0.109	SN II near-max	IIP (2 to 6)	0.822	Reliable	✓
DES16C1cbg	0.111	SN II post-max	IIP (−2 to 2)	0.999	Reliable	✓
DES16X2bvf	0.135	SN Ib post-max	Ib-norm (14 to 18)	0.65	Reliable	✓
DES16X2cpn	0.28	SN Ia +1 week	Ia-norm (6 to 10)	1.0	Reliable	✓
DES16X2crr	0.312	SN Ia near-max	Ia-norm (−2 to 2)	0.996	Reliable	✓
DES16X2bkr	0.159	SN II post-max	IIP (22 to 26)	0.953	Reliable	✓
DES16X2ceg	0.335	SN Ia near-max	Ia-norm (6 to 10)	0.709	Unreliable	✓
DES16E2cqq	0.426	SN Ia −1 week	Ia-norm (−2 to 2)	0.362	Reliable	✓
DES16E2clk	0.367	SN Ia near-max	Ia-91T (−2 to 2)	0.99	Reliable	✓
DES16E2crb	0.229	SN Ia near-max	Ia-norm (2 to 6)	0.998	Reliable	✓
DES16S1cps	0.274	SN Ia −1 week	Ia-91T (−6 to −2)	0.718	Reliable	✓
DES16E1ciy	0.174	SN Ia near-max	Ia-norm (2 to 6)	0.992	Reliable	✓
DES16X2dqz	0.204	SN Ib/c? max	Ic-norm (−2 to 2)	0.993	Reliable	✓
DES16X1der	0.453	SN Ia +1 week	Ia-norm (2 to 6)	0.982	Reliable	✓
DES16S2drt	0.331	SN Ia max	Ia-91T (−10 to −6)	0.737	Reliable	✓
DES16E1eef	0.32	SN Ia max	Ia-91T (−2 to 2)	0.582	Reliable	✓
DES16E1eae	0.534	SN Ia max	Ia-norm (−2 to 2)	0.994	Unreliable	✓
DES16X1dbx	0.345	SN Ia +1 week	Ia-91T (6 to 10)	0.942	Reliable	✓
DES16S2dfm	0.3	SN Ia near-max	Ia-norm (2 to 6)	1.0	Reliable	✓
DES16S2ean	0.161	SN Ia pre-max	Ia-norm (−6 to −2)	0.772	Reliable	✓
DES16X1dbw	0.336	SN Ia +1 week	Ia-91T (6 to 10)	1.0	Reliable	✓
DES16X1drk	0.463	SN Ia near-max	Ia-norm (−2 to 2)	1.0	Reliable	✓
DES16E2drd	0.27	SN Ia near-max	Ia-norm (−2 to 2)	0.927	Reliable	✓
DES16E2cxw	0.293	SN Ia +2 weeks	Ia-norm (10 to 14)	0.826	Reliable	✓
DES16C3dhv	0.3	SN Ia near-max	Ia-norm (2 to 6)	0.999	Reliable	✓
DES16X3dfk	0.15	SN Ia near-max	Ia-norm (2 to 6)	0.986	Reliable	✓
DES16E1dic	0.207	SN Ia max	Ia-norm (2 to 6)	0.985	Reliable	✓
DES16E1dcx	0.453	SN Ia +2 weeks	Ia-norm (10 to 14)	0.928	Reliable	✓
DES16S2ffk	0.373	SN Ia? −1 week	Ia-91T (−2 to 2)	0.754	Reliable	✓
DES16X1chc	0.043	SN Ia +2 months	Ic-norm (34 to 38)	0.965	Reliable	✓
DES16X1few	0.311	SN Ia −1 week	Ia-norm (−6 to −2)	0.946	Reliable	✓
DES16X2dzz	0.325	SN Ia? +2 weeks	Ia-norm (10 to 14)	0.771	Reliable	✓
DES16C1fgm	0.361	SN Ia −4 days	Ia-91T (−2 to 2)	0.979	Reliable	✓
DES16S1ffb	0.164	SN Ia near-max	Ia-norm (−6 to −2)	0.552	Reliable	✓
DES16X3enk	0.331	SN Ia? +1 week	Ia-norm (10 to 14)	0.725	Reliable	✓
DES16X3eww	0.445	SN Ia? max	Ia-norm (−2 to 2)	0.995	Reliable	✓
DES16C2ege	0.348	SN Ia? +1 month	Ic-norm (10 to 14)	0.999	Reliable	x
DES16X3dvb	0.329	SN II near-max	Ic-broad (−10 to −6)	0.895	Unreliable	✓
DES16C3elb	0.429	SN Ia +1 week	Ic-norm (10 to 14)	0.666	Unreliable	x
DES17E2ci	0.127	SN II post-max	IIn (42 to 46)	0.764	Reliable	✓
DES17E2ce	0.269	SN Ia +3 weeks	Ia-norm (18 to 22)	0.969	Reliable	✓
DES17E1by	0.287	SN Ia −1 week	Ia-norm (−2 to 2)	0.7	Reliable	✓
DES17E2bx	0.272	SN Ia at max	Ia-91T (−2 to 2)	0.999	Reliable	✓
DES17E2bw	0.147	SN Ia +2 weeks	Ia-norm (10 to 14)	0.998	Reliable	✓
DES17E2ar	0.513	SN Ia +10 days	Ia-csm (6 to 10)	0.944	Unreliable	✓
DES17E2aq	0.352	SN Ia −1 week	Ia-norm (−10 to −6)	0.848	Reliable	✓
DES17E2b	0.227	SN Ia +1 month	Ia-norm (18 to 22)	0.518	Reliable	✓
DES17E2a	0.295	SN Ia? +1 month	Ia-norm (22 to 26)	0.989	Reliable	✓
DES17X3ct	0.206	SN Ibc? post-max	Ib-norm (−6 to −2)	0.983	Unreliable	✓
DES17X3cb	0.317	SN Ia at max	Ia-norm (−2 to 2)	0.981	Reliable	✓
DES17X3ca	0.198	SN Ia? +6 weeks	Ia-norm (38 to 42)	0.585	Unreliable	✓

Table 2
(Continued)

Name	Redshift	ATel Classification	DASH			Match?
			Classification	Probability	Reliability	
DES17X3bd	0.141	SN II? post-max	IIP (26 to 30)	0.986	Reliable	✓
DES17X3az	0.56	SN Ia? +1 week	Ia-91T (6 to 10)	0.397	Unreliable	✓
DES17C3eg	0.117	SN Ia +3 weeks	Ia-norm (22 to 26)	0.977	Reliable	✓
DES17C3de	0.107	SN II post-max	IIP (38 to 42)	0.656	Reliable	✓
DES17E2cc	0.149	SN II post-max	IIP (18 to 22)	0.985	Reliable	✓
DES17S2byx	0.31	SN Ia? pre-max	Ia-norm (−10 to −6)	0.997	Reliable	✓
DES17E2bhj	0.186	SN II? post-max	Ib-norm (−6 to −2)	1.0	Unreliable	x
DES17X3bhi	0.39	SN Ia −1 week	Ia-norm (−6 to −2)	0.922	Reliable	✓
DES17X3btv	0.407	SN Ia near-max	Ia-91T (−6 to −2)	0.51	Reliable	✓
DES17S2als	0.388	SN Ia +1 month	Ia-norm (18 to 22)	0.885	Unreliable	✓
DES17E1byv	0.378	SN Ia pre-max	Ia-norm (−2 to 2)	1.0	Reliable	✓
DES17E2bro	0.223	SN Ia −1 week	Ia-91T (−2 to 2)	0.554	Reliable	✓
DES17X1boi	0.565	SN Ia near-max	Ib-norm (−6 to −2)	0.999	Unreliable	✓
DES17E1bmf	0.566	SN Ia near-max	Ic-broad (−10 to −6)	0.98	Reliable	x
DES17C3biz	0.23	SN Ia pre-max	Ia-norm (−10 to −6)	0.94	Reliable	✓
DES17E1axa	0.237	SN Ia +2 weeks	Ia-norm (14 to 18)	0.931	Reliable	✓
DES17X1ayb	0.292	SN Ia +2 weeks	Ia-91T (6 to 10)	0.702	Reliable	✓
DES17X1axb	0.139	SN II +10 days	IIP (18 to 22)	0.947	Reliable	✓
DES17X1aow	0.139	SN II post-max	IIP (18 to 22)	0.99	Reliable	✓
DES17X1alj	0.24	SN Ia? +3 weeks	Ia-norm (22 to 26)	0.946	Unreliable	✓
DES17E2sp	0.312	SN Ia +1 month	Ia-norm (18 to 22)	1.0	Reliable	✓
DES17C3dw	0.17	SN II post-max	IIP (6 to 10)	1.0	Reliable	✓
DES17X1gd	0.189	SN II? post-max	IIP (6 to 10)	0.651	Unreliable	✓
DES17S2bph	0.362	SN Ia? near-max	Ia-91T (−2 to 2)	0.981	Reliable	✓
DES17S2bop	0.385	SN Ia near-max	Ia-norm (2 to 6)	0.859	Reliable	✓
DES17E2boo	0.288	SN Ia near-max	Ia-csm (6 to 10)	0.858	Reliable	✓
DES17X2bmp	0.466	SN Ia? +1 week	Ia-norm (2 to 6)	0.801	Reliable	✓
DES17E2bmb	0.44	SN Ia near-max	Ia-norm (−2 to 2)	0.992	Reliable	✓
DES17X2blx	0.344	SN Ia max	Ia-norm (2 to 6)	0.64	Reliable	✓
DES17C1azd	0.338	SN Ia max	Ia-csm (6 to 10)	0.737	Reliable	✓
DES17X2bfi	0.34	SN Ia pre-max	Ia-norm (−2 to 2)	0.801	Reliable	✓
DES17E2arn	0.38	SN Ia +2 weeks	Ia-pec (2 to 6)	1.0	Reliable	✓
DES17X2alq	0.38	SN Ia? +2 weeks	Ia-norm (18 to 22)	0.997	Unreliable	✓
DES17X2agh	0.306	SN Ia? +2 weeks	Ia-norm (18 to 22)	0.946	Reliable	✓
DES17S2oo	0.23	SN II post-max	IIP (34 to 38)	0.861	Unreliable	✓
DES17S2lg	0.339	SN Ia? +1 month	Ia-norm (22 to 26)	0.621	Unreliable	✓
DES17X2abj	0.252	SN II? post-max	IIP (2 to 6)	0.984	Reliable	✓
DES17E1bud	0.552	SN Ia? near-max	Ia-norm (−10 to −6)	0.998	Unreliable	✓
DES17C2bqz	0.61	SN Ia? near-max	Ia-norm (−6 to −2)	0.971	Unreliable	✓
DES17E1bqq	0.463	SN Ia max	Ia-norm (−2 to 2)	0.558	Reliable	✓
DES17S1bof	0.226	SN Ia pre-max	Ia-91T (−6 to −2)	1.0	Reliable	✓
DES17E1bis	0.251	SN Ia +1 week	Ia-91T (6 to 10)	0.996	Reliable	✓
DES17E1beg	0.222	SN Ia +1 week	Ia-norm (6 to 10)	0.915	Reliable	✓
DES17S1aya	0.306	SN Ia? pre-max	Ia-csm (−14 to −10)	0.985	Reliable	✓
DES17S1bch	0.136	SN Ia max	Ia-norm (2 to 6)	0.995	Reliable	✓
DES17C2acb	0.35	SN Ia? +2 weeks	Ia-norm (18 to 22)	0.898	Reliable	✓
DES17C2pf	0.135	SN II post-max	II-pec (38 to 42)	1.0	Reliable	✓
DES17C2ou	0.103	SN Ia +2 months	Ia-norm (46 to 50)	0.991	Unreliable	✓
DES17S1lu	0.084	SN II post-max	IIP (38 to 42)	0.998	Reliable	✓
DES17C1bql	0.195	SN Ia −1 week	Ia-norm (−6 to −2)	0.761	Reliable	✓
DES17C3blq	0.511	SN Ia? max	Ic-broad (2 to 6)	0.344	Unreliable	x
DES17C3bei	0.103	SN II near-max	IIb (−2 to 2)	0.665	Reliable	✓
DES17C1bat	0.197	SN Ia +2 weeks	Ia-norm (6 to 10)	1.0	Reliable	✓
DES17C3aye	0.157	SN II post-max	IIb (−18 to −14)	0.977	Reliable	✓
DES17C1ayc	0.435	SN Ia +2 weeks	Ia-91bg (−2 to 2)	0.93	Unreliable	✓
DES17C1ald	0.131	SN Ia post-max	Ia-norm (22 to 26)	0.822	Reliable	✓
DES17S1emx	0.185	SN Ia? −1 week	Ia-91T (−10 to −6)	0.549	Reliable	✓
DES17S2ebs	0.304	SN Ia at max	Ia-norm (−2 to 2)	0.535	Reliable	✓
DES17C3dxw	0.622	SN Ia? near-max	Ic-norm (2 to 6)	1.0	Reliable	x
DES17X1dyt	0.33	SN Ia −1 week	Ia-91T (−6 to −2)	0.992	Reliable	✓
DES17X2dwm	0.3	SN Ia near-max	Ia-norm (2 to 6)	1.0	Reliable	✓
DES17X1dwi	0.252	SN Ia at max	Ia-91T (−2 to 2)	0.593	Reliable	✓

Table 2
(Continued)

Name	Redshift	ATel Classification	DASH Classification	DASH Probability	DASH Reliability	Match?
DES17X1diq	0.625	SN Ia? near-max	Ib-norm (−6 to −2)	0.744	Unreliable	x
DES17X1cuy	0.55	SN Ia? +1 week	Ib-norm (−6 to −2)	0.842	Unreliable	x
DES17X3dub	0.123	SN II near-max	IIP (22 to 26)	0.696	Unreliable	✓
DES17E1dgn	0.453	SN Ia near-max	Ia-norm (−6 to −2)	0.96	Reliable	✓
DES17C3doq	0.32	SN Ia at max	Ia-91T (−2 to 2)	0.909	Reliable	✓
DES17C1cpv	0.19	SN Ia +1 week	Ia-norm (6 to 10)	0.992	Reliable	✓

Note. The first column is the name of the observed object. The second column is the redshift determined by MARZ. The third column is the classification given in the ATel by OzDES. It details the type and age from maximum. A question mark after the classification type indicates that the ATel was not certain on the classification. Most of these ATel classifications were made by the OzDES team with the help of Superfit or SNID. The fourth, fifth, and sixth columns are the classification, softmax regression probability, and reliability from DASH, respectively. The final column has a tick if the ATel and DASH agree on the type of the SN, and a cross if they disagree.

ORCID iDs

Daniel Muthukrishna https://orcid.org/0000-0002-5788-9280
David Parkinson https://orcid.org/0000-0002-7464-2351
Brad E. Tucker https://orcid.org/0000-0002-4283-5159

References

Abadi, M., Barham, P., Chen, J., et al. 2016, Proc. 12th USENIX Conf. Operating Systems Design and Implementation, OSDI'16 (Berkeley, CA: USENIX Association), 265
Aniyan, A. K., & Thorat, K. 2017, ApJS, 230, 20
Astier, P., Guy, J., Regnault, N., et al. 2006, A&A, 447, 31
Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., et al. 2013, A&A, 558, A33
Baldry, I. K., Alpaslan, M., Bauer, A. E., et al. 2014, MNRAS, 441, 2440
Ball, N. M., & Brunner, R. J. 2010, IJMPD, 19, 1049
Bassett, B., Kasai, E., Crawford, S., et al. 2015, ATel, 8164, 1
Blondin, S., Matheson, T., Kirshner, R. P., et al. 2012, AJ, 143, 126
Blondin, S., & Tonry, J. L. 2007, ApJ, 666, 1024
Boureau, Y., Ponce, J., & Lecun, Y. 2010, in Proc. 27th Int. Conf. Machine Learning, ed. J. Fürnkranz & T. Joachims (Madison, WI: Omnipress), 111
Cabrera-Vives, G., Reyes, I., Förster, F., Estévez, P. A., & Maureira, J.-C. 2017, ApJ, 836, 97
Calcino, J., Davis, T. M., Hoormann, J. K., et al. 2018a, ATel, 11146, 1
Calcino, J., Davis, T. M., Hoormann, J. K., et al. 2018b, ATel, 11147, 1
Charnock, T., & Moss, A. 2017, ApJL, 837, L28
Childress, M. J., Lidman, C., Davis, T. M., et al. 2017, MNRAS, 472, 273
Cybenko, G. 1989, Mathematics of Control, Signals and Systems, 2, 303
Dark Energy Survey Collaboration, Abbott, T., Abdalla, F. B., et al. 2016, MNRAS, 460, 1270
Davis, T. M., Kim, A. G., Macualay, E., et al. 2015, ATel, 8367, 1
Davis, T. M., Mörtsell, E., Sollerman, J., et al. 2007, ApJ, 666, 716
de Jong, R. S., Agertz, O., Berbel, A. A., et al. 2019, Msngr, 175, 3
DESI Collaboration, Aghamousa, A., Aguilar, J., et al. 2016, arXiv:1611.00036
Dieleman, S., Willett, K. W., & Dambre, J. 2015, MNRAS, 450, 1441
Duda, R. O., Hart, P. E., & Stork, D. G. 2012, Pattern Classification (New York: Wiley), 654
Foley, R. J., Challis, P. J., Chornock, R., et al. 2013, ApJ, 767, 57
Glazebrook, K., Amon, A., Lidman, C., et al. 2015, ATel, 8413, 1
Guillochon, J., Parrent, J., Kelley, L. Z., & Margutti, R. 2017, ApJ, 835, 64
Hála, P. 2014, arXiv:1412.8341
Hinton, G. E., & Salakhutdinov, R. R. 2006, Sci, 313, 504
Hinton, S. R., Davis, T. M., Lidman, C., Glazebrook, K., & Lewis, G. F. 2016, A&C, 15, 61
Hoormann, J. K., Asorey, J., Carollo, D., et al. 2016, ATel, 9855, 1
Howell, D. A., Sullivan, M., Perrett, K., et al. 2005, ApJ, 634, 1190
Jones, E., Oliphant, T., Peterson, P., et al. 2001, SciPy: Open Source Scientific Tools for Python, http://www.scipy.org/ version 1.1.0
King, A., Moller, A., Sommer, N. E., et al. 2016, ATel, 9570, 1
Kingma, D. P., & Ba, J. 2014, arXiv:1412.6980
Krizhevsky, A., Sutskever, I., & Hinton, G. E. 2012, in Proc. 25th Int. Conf. Neural Information Processing Systems 1, NIPS'12 (Redhook, NY: Curran Associates Inc.), 1097
Lewis, G. F., Mould, J., Lidman, C., et al. 2015, ATel, 8167, 1
Li, Deng 2012, ISPM, 29, 141
Lintott, C. J., Schawinski, K., Slosar, A., et al. 2008, MNRAS, 389, 1179
Liu, Y., & Modjaz, M. 2014, arXiv:1405.1437
Liu, Y.-Q., Modjaz, M., Bianco, F. B., & Graur, O. 2016, ApJ, 827, 90
Lochner, M., McEwen, J. D., Peiris, H. V., Lahav, O., & Winter, M. K. 2016, ApJS, 225, 31
LSST Science Collaboration, Abell, P. A., Allison, J., et al. 2009, arXiv:0912.0201
Macaulay, E., Allam, S., Tucker, D., et al. 2018, ATel, 11148, 1
Matheson, T., Kirshner, R. P., Challis, P., et al. 2008, AJ, 135, 1598
Modjaz, M., Blondin, S., Kirshner, R. P., et al. 2014, AJ, 147, 99
Modjaz, M., Liu, Y. Q., Bianco, F. B., & Graur, O. 2016, ApJ, 832, 108
Möller, A., Ruhlmann-Kleider, V., Leloup, C., et al. 2016, JCAP, 12, 008
Moller, A., Tucker, B. E., Yuan, F., et al. 2016, ATel, 8673, 1
Momcheva, I., & Tollerud, E. 2015, arXiv:1507.03989
Moss, A. 2018, arXiv:1810.06441
Mudd, D., Martini, P., Lewis, G. F., et al. 2016, ATel, 9742, 1
Muthukrishna, D., Narayan, G., Mandel, K. S., Biswas, R., & Hložek, R. 2019, PASP, 131, 118002
Muthukrishna, D., & Parkinson, D. 2016, JCAP, 11, 052
Muthukrishna, D., Sharp, R. G., Tucker, B. E., et al. 2017, ATel, 10759, 1
Nair, V., & Hinton, G. E. 2010, in Proc. 27th Int. Conf. Machine Learning, ImageNet Classification with Deep Convolutional Neural Networks, ICML'10 (Madison, WI: Omnipress), 807
Narayan, G., Zaidi, T., Soraisam, M. D., et al. 2018, ApJS, 236, 9
O'Neill, C. R., Moller, A., Sommer, N. E., et al. 2016a, ATel, 9636, 1
O'Neill, C. R., Moller, A., Sommer, N. E., et al. 2016b, ATel, 9637, 1
Pan, Y.-C., Foley, R. J., Galbany, L., et al. 2015, ATel, 8460, 1
Pastorello, A., Quimby, R. M., Smartt, S. J., et al. 2008, MNRAS, 389, 131
Perlmutter, S., Aldering, G., Goldhaber, G., et al. 1999, ApJ, 517, 565
Pinto, P. A., & Eastman, R. G. 2001, NewA, 6, 307
Razavian, A. S., Azizpour, H., Sullivan, J., & Carlsson, S. 2014, in Proc. 2014 IEEE Conf. Computer Vision and Pattern Recognition Workshops, CVPRW '14 (Washington, DC: IEEE Computer Society), 512
Riess, A. G., Filippenko, A. V., Challis, P., et al. 1998, AJ, 116, 1009
Sasdelli, M., Ishida, E. E. O., Vilalta, R., et al. 2016, MNRAS, 461, 2044
Schmidt, B. P., Suntzeff, N. B., Phillips, M. M., et al. 1998, ApJ, 507, 46
Sharp, R., Zhang, B., Sommer, N. E., et al. 2017, ATel, 9961, 1
Silverman, J. M., Foley, R. J., Filippenko, A. V., et al. 2012, MNRAS, 425, 1789
Smith, M., Sullivan, M., Childress, M., et al. 2015, ATel, 8176, 1
Sommer, N., Tucker, B. E., Moller, A., et al. 2016, ATel, 9504, 1
Szegedy, C., Liu, W., Jia, Y., et al. 2014, arXiv:1409.4842
Tonry, J., & Davis, M. 1979, AJ, 84, 1511
Tucker, B. E., Sharp, R., Yuan, F., et al. 2015, ATel, 8137, 1
van der Walt, S., Colbert, S. C., & Varoquaux, G. 2011, CSE, 13, 22
Yaron, O., & Gal-Yam, A. 2012, PASP, 124, 668
Yuan, F., Lidman, C., Davis, T. M., et al. 2015a, MNRAS, 452, 3047
Yuan, F., Tucker, B. E., Lidman, C., et al. 2015b, ATel, 8464, 1
Zhang, B. R., Childress, M. J., Davis, T. M., et al. 2017, MNRAS, 471, 2254