The Astrophysical Journal Letters, 837:L28 (6pp), 2017 March 10 https://doi.org/10.3847/2041-8213/aa603d

Deep Recurrent Neural Networks for Supernovae Classification

Tom Charnock and Adam Moss
School of Physics & Astronomy, University of Nottingham, Nottingham NG7 2RD, UK; tom.charnock@nottingham.ac.uk, adam.moss@nottingham.ac.uk
Received 2016 October 31; revised 2017 February 6; accepted 2017 February 7; published 2017 March 10

Abstract

We apply deep recurrent neural networks, which are capable of learning complex sequential information, to classify supernovae (code available at https://github.com/adammoss/supernovae). The observational time and filter fluxes are used as inputs to the network, but since the inputs are agnostic, additional data such as host galaxy information can also be included. Using the Supernovae Photometric Classification Challenge (SPCC) data, we find that deep networks are capable of learning about light curves, however the performance of the network is highly sensitive to the amount of training data. For a training size of 50% of the representational SPCC data set (around 10^4 supernovae) we obtain a type-Ia versus non-type-Ia classification accuracy of 94.7%, an area under the Receiver Operating Characteristic curve AUC of 0.986 and an SPCC figure-of-merit $F_1 = 0.64$. When using only the data for the early-epoch challenge defined by the SPCC, we achieve a classification accuracy of 93.1%, AUC of 0.977, and $F_1 = 0.58$, results almost as good as with the whole light curve. By employing bidirectional neural networks, we can acquire impressive classification results between supernovae types I, II and III at an accuracy of 90.4% and AUC of 0.974. We also apply a pre-trained model to obtain classification probabilities as a function of time and show that it can give early indications of supernovae type. Our method is competitive with existing algorithms and has applications for future large-scale photometric surveys.

Key words: methods: data analysis – supernovae: general – techniques: miscellaneous

1. Introduction

Future large, wide-field photometric surveys such as the Large Synoptic Survey Telescope (LSST) will produce a vast amount of data, covering a large fraction of the sky every few nights. The amount of data produced lends itself to new analysis methods that can learn abstract representations of complex data. Deep learning is a powerful method for gaining multiple levels of abstraction and has recently produced state-of-the-art results in tasks such as image classification and natural language processing (see Lecun et al. 2015 for an excellent overview of deep learning and references within for more details).

There are many applications of deep learning for large photometric surveys, such as: (1) the measurement of galaxy shapes from images; (2) automated strong lens identification from multi-band images; (3) automated classification of supernovae; (4) galaxy cluster identification. In this Letter, we will focus on supernovae classification using deep recurrent neural networks. The LSST, for example, is expected to find over 10^7 supernova (LSST Science Collaboration et al. 2009). However, it is estimated that only 5000 to 10,000 will be spectroscopically[1] confirmed by follow-up surveys (Matheson et al. 2013), so classification methods need to be developed for photometry. All previous approaches to automated classification (Newling et al. 2011; Karpenka et al. 2013; Lochner et al. 2016) have first extracted features from supernovae light curves before using machine-learning algorithms. One of the advantages of deep learning is replacing this feature extraction.

In this work, we will use *supervised* deep learning. During training, the machine is given inputs and produces a set of output predictions. It is also given the correct set of outputs. An objective loss function then measures the error between the predicted and target outputs, and the machine updates its adjustable parameters to reduce the error. It can then make predictions for unknown outputs.

Recurrent neural networks (RNNs) are a class of artificial neural network that can learn about sequential data (for an extremely comprehensive overview, see Medsker & Jain 1999). They are commonly used for tasks such as speech recognition and language translation, but have several possible applications in astronomy and cosmology for processing temporal or spatial sequential data. RNNs have several properties that make them suitable for sequential information. The inputs to the network are flexible, and they are able to recognize patterns with noisy data (for example, the context of a word in a sentence relative to others can vary, or a time stream can contain instrument noise).

The main problem with vanilla RNNs is that they are unable to store long-term information, so inputs at the end of a sequence have no knowledge of inputs at the start. This is a problem if the data have long-term correlations. Several types of RNNs have been proposed to solve this problem, including long short-term memory (LSTM) units (Hochreiter & Schmidhuber 1997) and gated recurrent units (GRU; Chung et al. 2014). These are similar in concept, in that information is able to flow through the network via a gating mechanism. Another problem with RNNs is that information can only flow in one direction. In *bidirectional* RNNs, information is able to pass both forward and backward. Bidirectional LSTM networks have been shown to be particularly powerful where sequential data are accompanied by a set of discrete labels.

The architecture of a typical bidirectional RNN for sequence labeling is shown in Figure 1, where the squares represent neurons. In this case, the inputs, which are vectors at each sequential step, are connected to two hidden RNN layers, either vanilla RNN or memory units. Each hidden layer contains a number of hidden units (capable of storing information), and in each layer information flows either forward or backward, but no information passes between the two directions. Several

[1] Although, these numbers are not guaranteed.

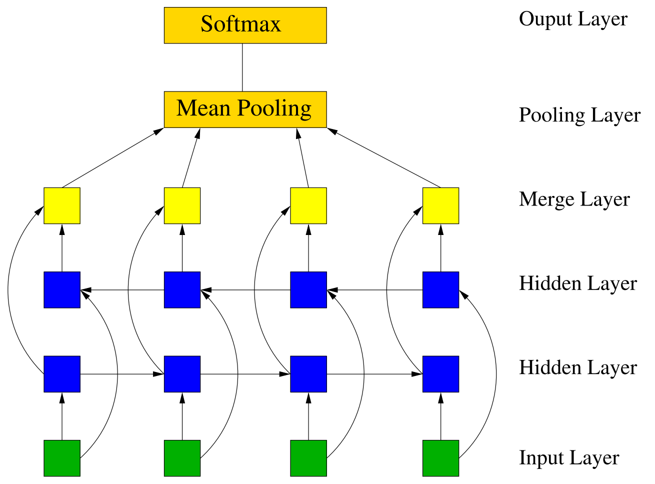

Figure 1. Bidirectional recurrent neural network for sequence classification. The input vectors at each sequential step are fed into a pair of bidirectional hidden layers, which can propagate information forward and backward. These are then merged to obtain a consensus view of the network, and finally a softmax layer computes classification probabilities.

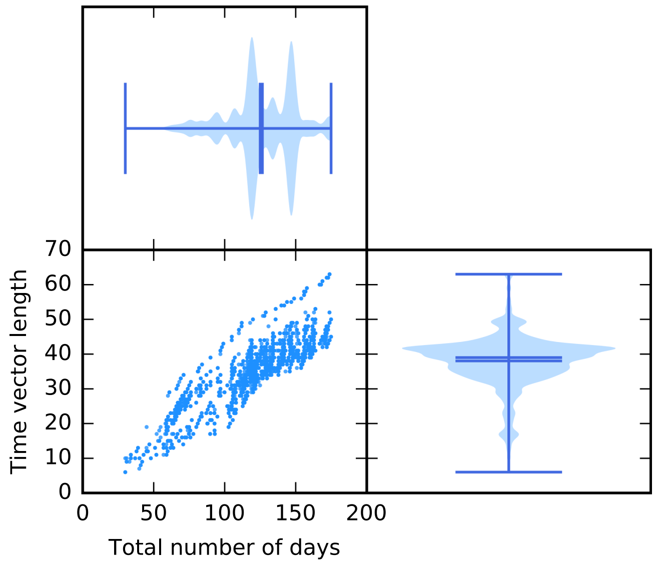

Figure 2. (Top) Distribution of the total number of days for each light curve with the minimum, maximum, mean, and median values indicated. (Bottom right) Distribution of the number of elements in the grouped time vector with the minimum, maximum, mean, and median values indicated. (Bottom left) The trend showing that more days in the light curve result in longer group time vectors.

hidden layers can be stacked to form *deep* neural networks. Deep networks are capable of learning higher-level temporal or spatial representations, and complex relationships between the inputs and outputs.

The output from the final set of hidden layers in each direction is merged at each sequential step and mean pooled (averaged) over all steps to obtain a consensus view of the network.[2] Finally, the mean output is fed to a *softmax* layer, taking an input vector z and returning normalized, exponential outputs for each class label i, $\exp(z_i)/\sum_i \exp(z_i)$, i.e., a vector of probabilities.

Each neuron is connected to another by a weight matrix, and the optimal weights are found by back-propagating the errors from a *loss function* of the output layer. For classification problems, this is typically the categorical cross-entropy between predictions and targets, defined as

$$L = -\sum_{i,j} t_{i,j} \log(p_{i,j}), \qquad (1)$$

where i, j run over the class labels, $t_{i,j}$ are the targets for each class (either 0 or 1), and $p_{i,j}$ are the predicted probabilities. Back-propagation takes the derivative of the loss with respect to the weights W of the output layer, $\partial L/\partial W$, and uses the chain rule to update the weights in the network.

2. Example Data

In this Letter, we will consider data from the Supernovae Photometric Classification Challenge (SPCC; Kessler et al. 2010a, 2010b), consisting of 21,319 simulated supernova light curves. Each supernovae sample consists of a time series of flux measurements, with errors, in the g, r, i, z bands (one band for each time step), along with the position on the sky and dust extinction. An example set of light curves is shown in Figure 3.

Due to the format of the input data, we first do a small amount of data processing to obtain values of the g, r, i, z fluxes and errors at each sequential step. We assume the time sequence begins at day 0 for each supernovae, rather than counting days forward and backward from the maxima of the light curve. For observations less than ~1 hr apart, we group the g, r, i, z values into a single vector, ensuring there is at most one filter-type in each group. If there is more than one filter-type, we further subdivide the group using a finer time interval. The group time is the mean of the times of each observation, which is reasonable, as the time intervals are small compared to the characteristic time of the light curve.

In Figure 2, we show how the length of the grouped time data vector is related to the duration of the light curve. The bottom left subplot shows that more total number of days since the beginning of observation of the light curve results in a greater number of grouped time elements in the vector. The upper subplot shows that the distribution of observation lengths in the SPCC data varies significantly with two distinct peaks. These are grouped into an average of 40-element data vectors as can be seen in the bottom right subplot.

Observations are of the form in Table 1, where any missing values are denoted by a dash. In order to impute the missing value of i, we use *data augmentation* and randomly select a value between i_1 and i_3. We make five random augmentations of all missing data, thereby increasing the size of the data set fivefold. We can test the importance of this by training each augmentation separately and comparing the change in accuracy, which we find is ~1%. Training with multiple augmentations at once gives the best performance since the network learns to ignore random-filled values.

The data come in two types, those with and those without the host galaxy photometric redshift. Each data set is split into a training and test set, with the training set containing a spectroscopically confirmed supernovae type and redshift. It is important that augmented data with the same supernovae ID go into either the training *or* test set; otherwise, they will not be independent. The original SPCC data consisted of 1103 training samples. The answer keys were subsequently made available for the test set (Kessler et al. 2010a).

[2] We find that obtaining a consensus view improves the performance of the network.

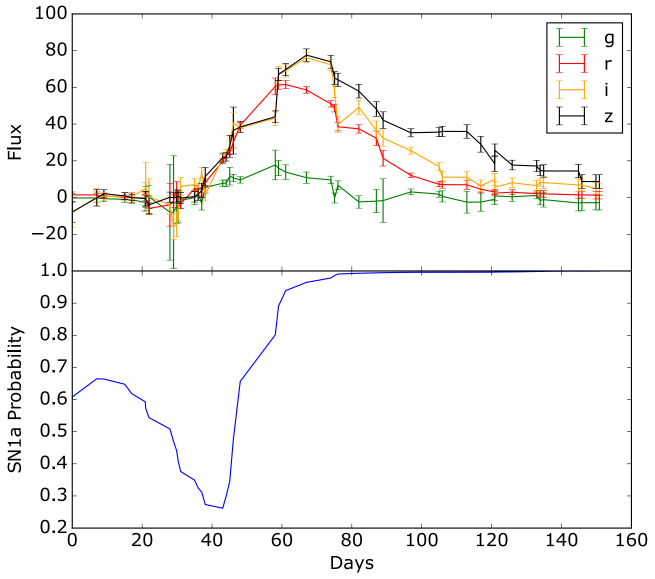

Figure 3. (Top) Example light curve in the four g, r, i, z bands for SN ID 551675 (a type-Ia) in the Supernovae Photometric Classification Challenge data (Kessler et al. 2010b). The data have been processed using augmentation so there is a g, r, i, z value at each sequential step. (Bottom) Type-Ia probability as a function of time from a two-layer LSTM model, trained with around 10^4 supernovae and SN 551675 excluded. The final probability gives 99.5% confidence that the supernovae is of type-Ia.

Table 1
Data Augmentation of Missing Observations

Time	g	r	i	z
t_1	g_1	r_1	i_1	z_1
t_2	g_2	r_2	…	z_2
t_3	g_3	r_3	i_3	z_3

Note. The missing data are replaced randomly by a value between i_1 and i_3.

The input vector to each sequential step consists of: time in days since the first observation; flux in each of the four bands; flux errors in each of the four bands; R.A. and decl.; dust extinction; and host photo-z if relevant. While we do not expect some of these variables to impact the classifier accuracy, we do not attempt any feature engineering and leave it to the network to decide if they are relevant.

RNNs typically perform better with more training data, so we train using the SPCC test set with answer keys (which is a non-biased representational data set[3]), and select a random fraction to act as the training set. We consider 1103 supernovae (a training fraction of 0.052), the same size as the original challenge, and fractions of 0.25 and 0.5 (around 5000 and 10^4 supernovae, respectively), nearly an order of magnitude larger, and closer to the number likely to be followed up for the LSST. The training performance of RNNs is also improved if the data are processed in mini-batches. In order to do this, the input data must be of the same length, so we set the sequence length to be the maximum length over all supernovae observations and prepend the input with padding. In training the network, we ensure the padding is ignored by masking the padded input.

The times of the observations in the light curve are irregularly spaced and while this may not be optimal for the network, we find that it is better to use the data padded at the end of the sequence than to place observations at similar times in similar sequence positions. There may even be hidden connections between the clustering of observation times and supernovae type, although it is hard to test for this.

[3] The original SPCC training set was non-representational.

The goal of the classifier is to determine the supernovae type in the test set. We consider two problems, (1) to categorize two classes (type-Ia versus non-type-Ia), and (2) to categorize three classes (supernovae types 1, 2, and 3). We denote these as "SN1a" and "123" respectively. We also attempt the first two problems using only the first six observations with $S/N > 4$ and the data taken on the night of the sixth observation as described in Kessler et al. (2010b).

Several metrics are used to assess the classifier. The simplest is the accuracy, defined as the ratio between the number of correct predictions and total number of predictions. With two classes, a random classifier would have an accuracy of 0.5, and with three classes, an accuracy of $1/3$.

Next are a variety of metrics coming from the *confusion matrix* of predictions. For binary classification problems, the confusion matrix splits predictions into true positives (TPs), false positives (FPs), false negatives (FNs), and true negatives (TNs). We consider the purity and completeness of the classifier. These are defined as

$$\text{Purity} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Completeness} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{2}$$

We evaluate these for each class separately versus "the rest" (e.g., type-Ia versus non-type-Ia). The SPCC also defined the F_1 figure-of-merit for the SN1a classification problem. This is

$$F_1 = \frac{1}{\text{TP} + \text{FN}} \frac{\text{TP}^2}{\text{TP} + 3 \times \text{FP}}, \tag{3}$$

so incorrectly classifying a non-type-Ia supernovae as a type-Ia is penalized more heavily.

Finally, we calculate the area under the curve (AUC). The AUC is the area under the curve of the TP rate versus FP rate, as the threshold probability for classification is increased from 0 to 1. A perfect classifier has an AUC of 1, and a random classifier 0.5. For multi-class problems, we calculate the AUC for each class versus the rest and take an unweighted average to give the final AUC score.

3. Network Architecture

We consider several combinations of the network architecture. For the RNN type in the hidden layers, we test both vanilla RNN and long-term memory (LSTM and GRU) units. We also consider unidirectional and bidirectional networks. For unidirectional networks we fix the direction to be forward. For bidirectional networks, the number of hidden units in each RNN layer is equal in the forward and backward directions.

We also test stacking two sets of layers to form a deep network. In the unidirectional case, we stack two hidden layers. In the bidirectional case, the two stacks consists of a pair of forward and backward layers. We denote the number of hidden units in a network with a single stack by $[h_1]$, and the number of hidden layers in a two-stack model by $[h_1, h_2]$. We vary the number of hidden units, testing h = [4], [8], [16], [32], [4, 4], [8, 8], [16, 16], and [32, 32]. We do not go beyond a stack of two layers due to the limited size of the data set.

For each network we perform five randomized runs over the training data to obtain the classifier metrics. The loss function is

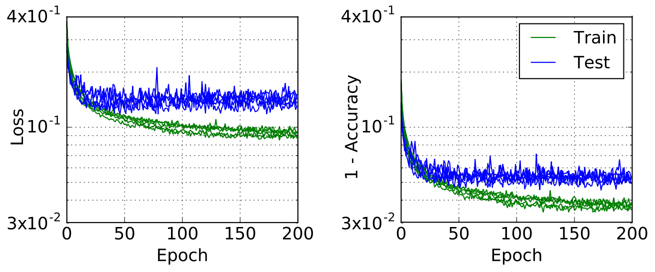

Figure 4. (Left) Training loss (green) vs. test loss (blue) for a unidirectional 2 layer LSTM network with 16 hidden units in each layer. (Right) Training accuracy (green) vs. test accuracy (blue) for the same network.

the categorical cross-entropy between the predictions and test data. The network weights ware trained using back-propagation with the `Adam` updater (Kingma & Ba 2014). Mini-batches containing 10 samples[4] were used throughout, and each model was trained for 200 *epochs*, where each epoch is a full pass over the training data.

4. Results

A data set of 21,319 is relatively small by deep learning standards. Furthermore, the "feature space" of supernovae light curves is significantly smaller than, say, using RNNs to learn about language. We therefore need to be careful about *overfitting*. Overfitting arises when the network learns about relations between the inputs and outputs of the training data, which do not exist in the test data. It can typically be detected by comparing the loss of the training and test data. If the loss of training data continues to decrease, but the loss of the test data increases, this is a sure sign of overfitting. If no sign of overfitting is observed, the network is not usually complex enough to fully learn the relationship between inputs and outputs (called *underfitting*).

For a training fraction of 0.5, we found the best architecture was a deep two-layer network with unidirectional LSTM units. Bidirectional units did not significantly improve the test accuracy and made the network more difficult to train. There was a marked improvement in test accuracy using 16 hidden units in each layer rather than 8, but too much overfitting occurred using 32 hidden units. Overfitting was still an issue for 16 hidden units, but a technique called *dropout* (Srivastava et al. 2014) could regularize this. Dropout sets a random fraction of connections to 0 at each update during training *only*, preventing the units from adapting too much. We apply dropout only to non-recurrent connections after each hidden layer.

In Figure 4, we show the training and tests losses for such a network, with a dropout of 0.5, applied to type-Ia versus non-type-Ia classification with host galaxy photo-z information. Without dropout, the training loss continues to fall and the test loss rises. For 5 randomized runs, training for 200 epochs, we obtain a classification accuracy of $94.9 \pm 0.2\%$, AUC of 0.986 ± 0.001, and $F_1 = 0.64 \pm 0.01$. The corresponding type-Ia purity and completeness are $87.3 \pm 0.8\%$ and $91.4 \pm 1.1\%$, respectively. A summary of results and comparisons can be found in Table 2. The inclusion of host galaxy photo-z marginally improves the classifier performance. The 1σ errors quoted in the table are the result of five runs where the training data is randomly chosen (and therefore different) each time. Some random choice of the set of light curves are more effective for training the network than others, but it is extremely difficult to optimize this.

To test the robustness of the time-grouping method, we remove 10% of the known filter values (and/or their errors) before grouping the data into a single vector and randomly augmenting the missing values. After training we find there is a small degradation in the results, i.e., for a training fraction of 0.5 using a deep two-layer, unidirectional network with 16 hidden units, a dropout of 0.5, and including the photo-z information the obtained results are very similar to the second line in Table 2. This shows that a reduction in 10% of the points is similar to the omission of the photo-z data, and therefore the data augmentation method is extremely robust.

One advantage of our approach is that light curve data can be directly input to a *pre-trained* model to give very fast evaluation (<1s) of supernovae type. In the lower panel of Figure 3, we input the light curve, as a function of time, of a type-Ia supernovae (excluded from training) to the pre-trained two-layer LSTM model discussed above. The classifier (type-Ia versus non-type-Ia) is initially unsure of classification, with a type-Ia probability of around 0.5. The probability then decreases slightly, but rapidly increases near the peak of the light curve. The classifier has high confidence that the supernovae is of type-Ia at around 60 days, and the final probability is in excess of 99.5%. This method could therefore be useful to give early indication of supernovae type in surveys.

We also test the same model using a training fraction of 0.25 (around 5000 supernovae), closer to the lower end of the number likely to be followed up for the LSST. After 5 randomized runs and training for 200 epochs, we obtain an accuracy of $92.9 \pm 0.6\%$, AUC of 0.975 ± 0.003, and $F_1 = 0.57 \pm 0.03$. The corresponding type-Ia purity and completeness are $86.6 \pm 2.0\%$ and $83.4 \pm 3.4\%$, respectively. The F_1 metric has degraded by $\sim$10% for a reduction in data of 50%.

For 5.2% of the representative SPCC data, the training data set is so small that overfitting is more severe. Using the same two-layer LSTM network with 16 hidden units and dropout of 0.5, we find a notable increase in the test loss after $\sim$20 epochs, but the accuracy and other metrics remain relatively constant (F_1 values of 0.35 to 0.4 were obtained). The reason for this apparent discrepancy is that the accuracy, say, simply takes the maximum value of the softmax output layer. For example, a two-class problem with output probabilities [0.6, 0.4] and target [1, 0] has the same accuracy as one with output probabilities [0.8, 0.2]. The loss in the latter case would be lower, however, and represents increased confidence of the network in its predictions. We therefore reject models with severe overfitting and an increasing cross-entropy loss at the expense of metrics such as F_1, and decrease the model complexity.

For a training fraction of 5.2% we find a single-layer LSTM network, with four hidden units, and dropout of 0.5 satisfies this criteria. For 5 randomized runs, training for 200 epochs, we obtain a classification accuracy of $85.9 \pm 0.9\%$, AUC of 0.910 ± 0.012, and $F_1 = 0.31 \pm 0.03$. The corresponding type-Ia purity and completeness are $72.4 \pm 0.4\%$ and $66.1 \pm 6.0\%$, respectively.

It is difficult to directly compare the results from the SPCC challenge in Kessler et al. (2010a) with this work since the

[4] If training with a GPU, larger mini-batches are recommended to make use of the GPU cores.

Table 2
Summary of Results

Method	Training size	AUC	Accuracy (%)	F_1	Purity (%)	Completeness (%)	Host-z
A	10,660	0.986 ± 0.001	94.7 ± 0.2	0.64 ± 0.01	87.3 ± 0.8	91.4 ± 1.1	True
A	10,660	0.981 ± 0.001	93.6 ± 0.3	0.60 ± 0.02	87.4 ± 1.7	85.4 ± 2.6	False
A	5,330	0.975 ± 0.003	92.9 ± 0.6	0.57 ± 0.03	86.6 ± 2.0	83.4 ± 3.4	True
A	5,330	0.973 ± 0.002	92.3 ± 0.4	0.55 ± 0.02	86.2 ± 2.4	80.8 ± 3.8	False
B	1,103	0.910 ± 0.012	85.9 ± 0.9	0.31 ± 0.03	72.4 ± 0.4	66.1 ± 6.0	True
B	1,103	0.901 ± 0.016	84.6 ± 1.7	0.28 ± 0.05	68.2 ± 3.4	66.3 ± 5.5	False
C	∼10,660	…	…	0.58	85	88	True
C	∼10,660	…	…	0.51	82	85	False
C	1,045	…	…	0.33	70	75	True
C	1,045	…	…	0.29	67	71	False
D	∼8,000	…	…	0.55	…	…	True
D	∼2,000	…	…	0.45	…	…	True
E	1,103	0.94 ± 0.03	…	…	…	…	True
E	1,103	0.89 ± 0.53	…	…	…	…	False
E	1,103	…	…	…	90	85	True
E	1,103	…	…	…	87	90	True
F	10,660	0.974 ± 0.001	90.4 ± 0.3	…	90.6 ± 0.7	86.5 ± 0.7	True
F	10,660	0.959 ± 0.006	88.5 ± 1.1	…	87.6 ± 1.1	85.9 ± 4.1	False
G	1,103	0.868 ± 0.015	78.1 ± 0.9	…	70.8 ± 3.4	70.6 ± 4.1	True
G	1,103	0.865 ± 0.011	78.0 ± 1.2	…	66.9 ± 3.2	74.5 ± 4.2	False
A	10,660	0.977 ± 0.002	93.1 ± 0.4	0.58 ± 0.01	88.0 ± 1.1	82.2 ± 2.8	True
A	10,660	0.970 ± 0.001	92.0 ± 0.3	0.53 ± 0.01	86.0 ± 0.9	79.5 ± 2.2	False
B	1,103	0.902 ± 0.014	85.2 ± 1.2	0.29 ± 0.04	71.5 ± 1.6	62.8 ± 5.6	True
B	1,103	0.860 ± 0.017	81.6 ± 1.2	0.21 ± 0.02	62.6 ± 3.0	57.6 ± 2.7	False
A	10,660	0.960 ± 0.006	87.9 ± 0.9	…	86.4 ± 0.8	84.4 ± 3.5	True
A	10,660	0.948 ± 0.002	86.8 ± 0.3	…	84.1 ± 1.1	83.7 ± 1.4	False
B	1,103	0.851 ± 0.013	76.8 ± 1.3	…	64.7 ± 3.8	71.0 ± 4.1	True
B	1,103	0.819 ± 0.010	74.2 ± 1.0	…	58.1 ± 3.8	73.6 ± 6.6	False

Note. (Top section) Summary of results for type-Ia vs. non-type-Ia classification with a training fraction of 0.5, 0.25, and 0.052 with comparisons to similar methods in Karpenka et al. (2013) and Newling et al. (2011). (Second section) Summary of results for types I, II, and III classification. (Third section) Summary of results for SPCC early-epoch challenge. (Bottom section) Summary of the results for the SPCC early-epoch challenge when classifying between Type I , II, and III supernovae. The models used are (A) unidirectional LSTM, [16, 16] with 0.5 dropout, (B) unidirectional LSTM, [4] with 0.5 dropout, (C) Karpenka et al. (2013), (D) Newling et al. (2011), (E) Lochner et al. (2016) SALT2 fits averaged over machine-learning architecture (F) bidirectional LSTM, [16, 16] with 0.5 dropout, and (G) bidirectional LSTM, [4] with 0.5 dropout. Errors on results are the mean and standard deviation values from five randomized runs.

figure of merit is quoted as a function of redshift and a non-representative set of light curves was originally used. In Kessler et al. (2010a), the method of Sako et al. (2008) had the highest average F_1, with 79% purity and 96% accuracy. This is a somewhat confusing average as $F_1 \sim 0.4$ at a redshift $z \sim 0.1$ up to $F_1 \sim 1$ at $z \sim 0.9$. Other methods performed similarly.

It is better to consider comparison with other methods using post-SPCC data, for we obtain results that are competitive with previous approaches. The analyses by Karpenka et al. (2013) and Newling et al. (2011) are easier to compare. Along with Lochner et al. (2016), these employ a two-step process, where features are first extracted by various methods before machine-learning classification. The results obtained for similar sized training sets are comparable, as can be seen in the top section of Table 2. When using half the data set to train on we get a higher F_1 value, $F_1 = 0.64$, compared to $F_1 = 0.58$ in Karpenka et al. (2013). The value in Newling et al. (2011) is also similar given that the sample size is smaller. For a smaller sample training set of 5.2% of all the data we again perform similarly to Karpenka et al. (2013), but underperform compared to Newling et al. (2011), taking into account the slightly larger sample size in the latter case. In Lochner et al. (2016), using the SALT2 fits provided the best average AUC over a range of machine-learning techniques. By imposing a purity of 90%, a completeness of 85% was achieved, while requiring a completeness of 90% reveals a corresponding purity of 85%.

In the second section of Table 2, the three-class categorization is shown. There are no available data for comparison of this problem, but compared to classification between type-Ia versus non-type-Ia, bidirectional recurrent neural networks do well. The AUC and accuracy remain high, still above 90%, when the host-z is included using a training fraction of 0.5. Using a smaller training fraction of 0.052, the results are worsened, similar to the two-class categorization in the top section of Table 2.

The third section of Table 2 shows the results of the early-epoch challenge from SPCC. Here, only the data before the night of the sixth observation with $S/N > 4$ for each light curve can be used—a great reduction from the use of the full light curve. We do surprisingly well in this case obtaining an accuracy of $93.1 \pm 0.4\%$, AUC of 0.977 ± 0.002, and an $F_1 = 0.58 \pm 0.01$ with a training fraction of 0.5 and including host-z. These values are not far from those obtained using the whole light curve and are equivalent to the full results of Karpenka et al. (2013). The results are not as good with a training fraction of 0.052, but still comparable to our results using the whole light curve. The network trained on the partial light curves does better than suggested from feeding the early-epoch light curve through a network trained on the full sequence. This is due to the later parts of the light curve influencing the weights of the network while training. Training on only the initial part of the light curve optimizes the network weights such that early sequence features have more effect, resulting in better accuracy, AUC, and F_1 values than expected.

Finally, the bottom section of Table 2 has the results of the three-class categorization when using the early-epoch data. The results are similar to the difference between the full light curve and early-epoch data SN1a categorization when comparing with the full light curve 123 categorization. It should be noted that the bidirectional network used for the 123 categorization using the full light curve revealed sizable overfitting when using the early-epoch data, and so a unidirectional network was used instead.

5. Conclusions

We have presented a new method for performing photometric classification of supernovae. Machine-learning methodology has previously been applied to SPCC classification (Newling et al. 2011; Karpenka et al. 2013; Lochner et al. 2016). Instead of performing feature extraction before classification, our approach uses the light curves directly as inputs to a recurrent neural network, which is able to learn information from the sequence of observations.

Although we have trained the network on the cross-entropy loss and not the F_1 score, for the same sized data set of $\sim 10^3$ (10^4) supernovae (including host galaxy photo-z), Karpenka et al. (2013) obtained F_1 values of 0.33 (0.58), and Newling et al. (2011) values of 0.42 (0.57), compared to our 0.31 (0.64). Recurrent neural networks therefore compare well with other methods when a larger training set is available. The performance is not quite as good with a smaller training set, possibly due to the network having to learn from no prior information about (noisy) light curves. The current state of the art for a small training set ($\sim 10^3$ supernovae) comes from a combination of Spectral Adaptive Light curve Template 2 (SALT2) template fits and boosted decision trees (Lochner et al. 2016). It would be interesting to check how how deep learning compares to this with a larger training set.

As well as finding competitive results for the final metrics, we have shown that it is possible to give fast, early evaluation of supernovae type using pre-trained models. This is possible since the light curve can be fed to the model directly without needing any feature extraction.

Most interestingly, we have found that training a network only on the early-epoch light curve data results in a better early-time predictor than using a network trained on entire light curve data. Our results using only the early-epoch data are close to those using the entire light curve data for both SN1a and 123 categorization with both large and small training fractions.

There are several possibilities for future work. One of the advantages of recurrent neural networks is that inputs are agnostic, so the impact of any additional inputs could be explored. It would be possible, for example, to even pass the raw images in each filter though a convolutional network and use those as inputs. We have considered a representative training sample, but spectroscopic follow-up surveys may be biased. The performance of the network could be measured against selection bias, and the results used to inform the best follow-up strategy. Further work could also be performed to optimize the early detection probability of the network. Finally, to improve performance in the small data regime, one can use *transfer learning*. Here, a more complex network is pre-trained on simulations or existing data from other surveys, then the weights of the network are fine-tuned on the new, smaller data set. The simulated SPCC data used in this work are based on the DES instrument, and we are applying transfer learning to real DES data for publication in future work.

We appreciate helpful conversations with Steven Bamford, Simon Dye, Mark Sullivan, and Michael Wood-Vasey, and Natasha Karpenka, Richard Kessler, and Michelle Lochner for help with data acquisition. T.C. is supported by an STFC studentship, and A.M. is supported by a Royal Society University Research Fellowship.

References

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. 2014, arXiv:1412.3555
Hochreiter, S., & Schmidhuber, J. 1997, Neural Comput., 9, 1735
Karpenka, N. V., Feroz, F., & Hobson, M. P. 2013, MNRAS, 429, 1278
Kessler, R., Bassett, B., Belov, P., et al. 2010a, PASP, 122, 1415
Kessler, R., Conley, A., Jha, S., & Kuhlmann, S. 2010b, arXiv:1001.5210
Kingma, D., & Ba, J. 2014, arXiv:1412.6980
Lecun, Y., Bengio, Y., & Hinton, G. 2015, Natur, 521, 436
Lochner, M., McEwen, J. D., Peiris, H. V., Lahav, O., & Winter, M. K. 2016, arXiv:1603.00882
LSST Science Collaboration, Abell, P. A., Allison, J., et al. 2009, arXiv:0912.0201
Matheson, T., Fan, X., Green, R., et al. 2013, arXiv:1311.2496
Medsker, L., & Jain, L. C. 1999, Recurrent Neural Networks: Design and Applications (Boca Raton, FL: CRC Press)
Newling, J., Varughese, M., Bassett, B., et al. 2011, MNRAS, 414, 1987
Sako, M., Bassett, B., Becker, A., et al. 2008, AJ, 135, 348
Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. 2014, J. Mach. Learn. Res., 15, 1929