CS230

It's a bird... It's a plane... It's a Type Ia supernova!
Astronomical Object Classification from Photometric Time-Series Data

Ryan W. Gao
Stanford University
rgao@stanford.edu

Abstract

Astronomical data volume is exploding due to improved telescope technology, and thus, the field requires novel methods for data processing. This paper focuses on automated object classification based on photometric time-series data and object metadata, such as sky position and redshift. We evaluated both convolutional and recurrent neural networks (CNNs and RNNs), both with and without object metadata. The best performing model - the CNN with metadata features - achieved an equal-class-weighted accuracy of 70% on the 14-class classification problem.

1 Introduction

Astronomy is a field that is ripe with potential for applications of deep learning. The amount of data collected by the world's telescopes is increasing exponentially, and there is a concurrent rise in the need for efficient and accurate automated data processing techniques. Going live in 2022, the Large Synoptic Survey Telescope (LSST) is expected to generate 20-40 TB of data per night of observation. The ongoing Kaggle competition PLAsTiCC (Photometric LSST Astronomical Time-Series Classification Challenge) [1] is just one example of the concerted effort by the field's experts to leverage the techniques of machine learning to process data of this scale. The LSST will be recording primarily photometric time-series data, which are discrete measurements of object brightness within a particular wavelength range (called the passband). The LSST will observe both galactic objects (within the Milky Way) like variable stars and binary star systems, and extragalactic sources like quasars and supernovae of various types (Ia, Ib/c, II), as well as potentially previously unobserved object classes. This project provides a solution to the PLAsTiCC challenge, by applying convolutional and recurrent neural networks (CNNs and RNNs, respectively) to this time-series classification problem.

2 Related Work

Historically, automated object classification in astronomy has been focused on identifying broad categories, such as galaxy vs. star using single-measurement spectroscopic data from sky surveys. The learning algorithms of choice were Naive Bayes [5] and decision trees [6]. Recent work has been done on classifying light curves (multiple measurements over time) with more sophisticated classification algorithms like random forests, SVMs, neural nets, boosted decision trees, and even bi-LSTMs [8,9,10,11,12,13]. However, these studies have all focused exclusively on classifying different types of supernovas, and most require first fitting the light curve to a supernova-specific parametric model called SALT2 [14].

CS230: Deep Learning, Autumn 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

3 Dataset

The dataset we use is sourced from the PLAsTiCC Kaggle competition, hosted by the LSST Transients and Variable Stars Collaboration team. The dataset is generated from simulations of variable and transient sources (objects whose brightnesses change over time), with noise characteristics consistent with current state-of-the-art telescope technology. Because this is from a Kaggle competition, only the training set has labels, so we will scope this paper to analysis using only that set of 7848 examples. Due to the small dataset size, we use a 75/25 training/dev set split, and choose to forgo a test set.

3.1 Features

In this paper, we will refer to features that have multiple instances per object as "data," and features that have one value per object as "metadata."

The input data features include the photometric time series of brightness measurements `flux` taken with 6 different color filters, or `passbands`. The measurements are taken aperiodically over a span of 2 to 3 years, with the datetime expressed as Modified Julian Date, `mjd`. Each flux measurement is accompanied by the estimated flux error `flux_err` and a boolean `detected`, indicating whether there was a change detected relative to the background "template" image.

Each object is also associated with the following metadata: the location in the sky (in both equatorial coordinates - `ra`, `decl` - and galactic coordinates - `gal_l`, `gal_b`), whether it was part of the "Wide-Fast-Deep" (WFD) or "Deep Drilling Fields" (DDF) survey `ddf`, the distance from Earth (measured in units of redshift, both photometrically `hostgal_photoz` and spectroscopically `hostgal_specz`, and in units distance modulus `distmod`), the photometric redshift error `hostgal_photoz_err`, and the extinction due to galactic dust `mwebv`.

Finally, there is a metadata field `target`, which is the ground-truth object class. There are 14 possible classes, but this field is deliberately obfuscated in the Kaggle dataset, so we don't know the underlying astrophysical object types.

3.2 Pre-processing

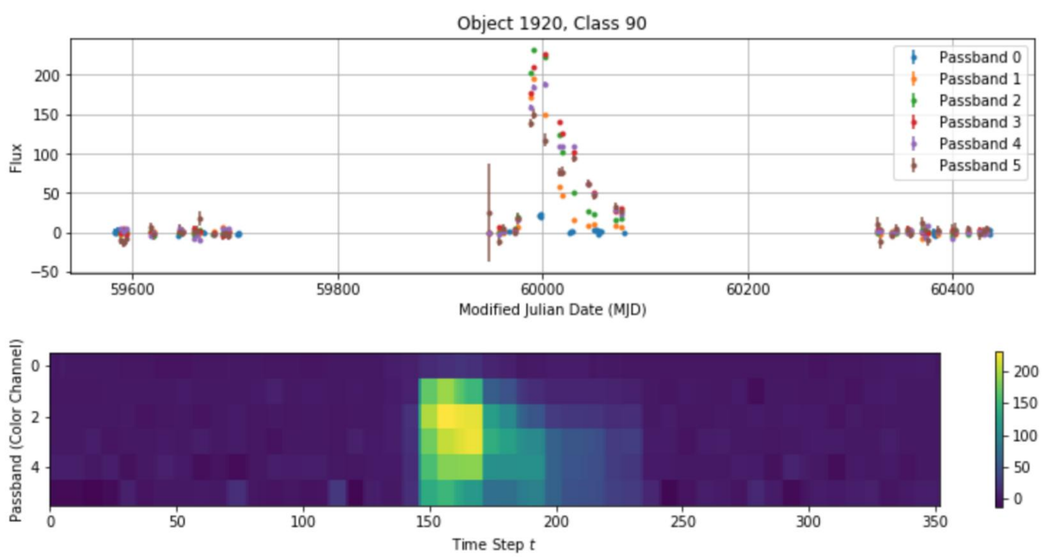

Figure 1: `flux` data transformation: a) Original data, with irregular intervals and length. b) Final input data, with missing data interpolated and length normalized to $T = 352$.

In order to convert the raw data streams into a format suitable for training our classifier, we pre-process the data using the following steps:

1. Observations are performed in only one passband at a time, so we linearly interpolate `flux` for the other 5 missing passbands at every measurement time `mjd`.

2. We replace measurement times `mjd` by regular time steps $t = 0, 1, 2, ...$ This ignoring irregular intervals between measurements, but standardizes the data input format.
3. Finally, we resize the data to the maximum length observed in the training set $T = 352$, using linear interpolation.

In addition to the 10 provided, we derive 6 more metadata features from the `flux` time series: min, max, mean, stdev, median, and skew (or third moment). We scale all 16 metadata features to zero mean and unit variance.

4 Methods

4.1 Loss Function

As is standard for multi-class classification problems, we use a cross-entropy loss function with weights

$$\mathcal{L} = -\frac{\sum_{i=1}^{C} \sum_{j=1}^{M} w_i y_{ij} \log p_{ij}}{\sum_{i=1}^{C} w_i},$$

where $C = 14$ is the number of classes, M is the total number of objects, y_{ij} is the indicator variable of object j belonging to class i, p_{ij} is the prediction probability of object j belonging to class i, and w_i is the weight of class i. For numerical stability, p_{ij} are clipped to fall within the interval $[10^{-15}, 1 - 10^{-15}]$.

The 14 `target` classes are unbalanced, with some classes having 80x the number of examples of other classes. Thus, we assign weights such that each class is equally important: $w_i = 1/M_i$, where M_i is the number of objects of class i in the training set.

Lastly, our optimization algorithm is Adam [2] with typical parameters of $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$. We find the optimal learning rate $\alpha = 0.01$ by training roughly 20 epochs and searching over the logarithmic space between $\alpha = 10^{-6}$ and $\alpha = 10^0$.

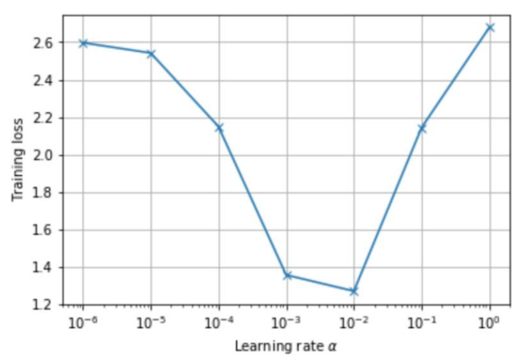

Figure 2: Hyperparameter tuning - training loss after 20 epochs, as a function of learning rate α.

4.2 Models

We train the following two neural network architectures on the time-series data:

- **1D CNN**: This convolutional neural network (CNN) is inspired by the VGGNet architecture [4]. Every CONV layer has filter size 3 and same padding ($p = 1$), and is followed by BatchNorm, ReLU, and MaxPool of filter size 2 and stride 2. Below is the architecture, along with the dimensions after each block (length $\times$ channels).

  INPUT $(352 \times 6) \rightarrow$ CONV16 $(176 \times 16) \rightarrow$ CONV32 $(88 \times 32) \rightarrow$ CONV32 (44×32)

  $\rightarrow$ CONV32 $(22 \times 32) \rightarrow$ CONV32 $(11 \times 32) \rightarrow$ FLATTEN (352)

- **RNN**: This recurrent neural network (RNN) architecture is single-layer, uni-directional, and uses long-short-term-memory (LSTM) cells [7] with 32 hidden nodes. The output is from the final time-step $T = 352$.

To incorporate the metadata features, we concatenate the output of the CNN or RNN to the 16 metadata features. Finally, the final output layer is fully connected with $C = 14$ nodes and softmax activation.

5 Results & Discussion

5.1 Evaluation Metric

Our evaluation metric is accuracy, weighted such that all classes are equally important:

$$A = \frac{1}{C} \sum_{i=1}^{C} \sum_{j=1}^{M} \frac{y_{ij}}{M_i} \hat{y}_{ij},$$

where C, M, M_i, and y_{ij} are defined above in Section 4.1. $\hat{y}_{ij}$ is the indicator variable for predicting object j belongs to class i.

5.2 Results

Here are the results after training each model for roughly 46 epochs:

Model	# Trainable Params	Training Accuracy	Test Accuracy
Metadata only	238	55.07%	48.40%
1D CNN only	16,414	66.26%	55.11%
LSTM only	5,582	48.77%	35.25%
CNN + Metadata	16,638	78.61%	70.09%
LSTM + Metadata	5,806	72.29%	51.58%

As a baseline, we trained a logistic regression model on only the metadata features, which achieved 48% accuracy. Adding the LSTM model marginally improves the accuracy to 52%, but adding the CNN model significantly improves accuracy to 70%. This can be explained by the CNN encoding significantly different information about the time-series than the 6 simple derived metadata features (min, max, mean, stdev, median, skew). Figure 3 illustrates the recall, or fraction of true class i items correctly labeled, for each of the 5 models. For example, the CNN-only model performs better than the metadata-only model on some classes (15, 53, 67) and worse on others (64, 95). Thus, the combined model can take advantage of information from both models.

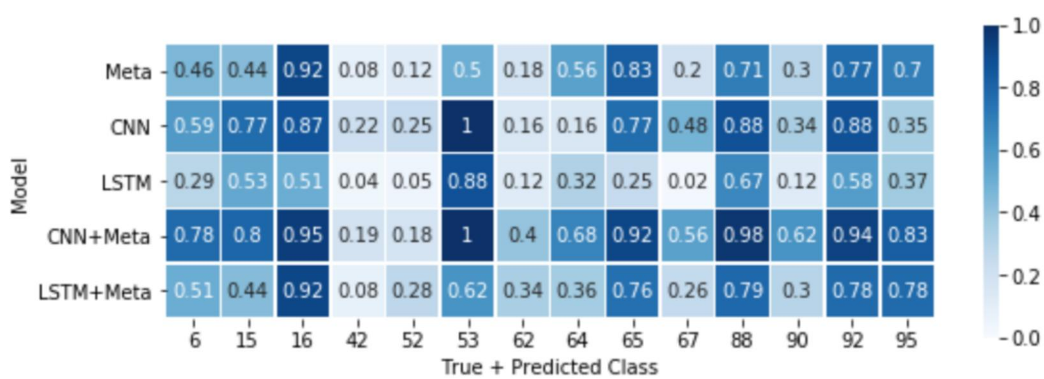

Figure 3: Recall matrix, showing the fraction of objects of class i correctly labeled by a given model.

From the confusion matrix in Figure 4, the five most difficult classes to predict, 42, 52, 62, 67, and 90, are often confused with each other. From the metadata features, these classes are always extra-galactic objects, and from visual inspection of the light curves, they all appear to be abrupt flux increases followed by exponential-like decay in brightness (example in Figure 1). This description matches that of a supernova, so there is potential in augmenting the model with a supernova-specific parametric model, such as the ones described in Section 2.

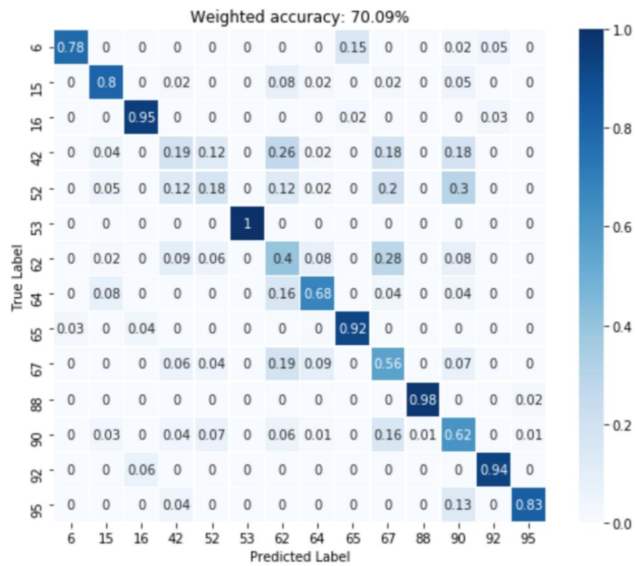

Figure 4: Confusion matrix for the CNN + Metadata model. Values are normalized such that diagonal elements are recall, or equivalently, such that each row sums to 1.

6 Conclusion

The 1D CNN combined with metadata features performed well on this 14-class classification problem, reaching 70% weighted accuracy. The RNN architecture was not as successful, only reaching 52% accuracy. Judging from the size of the training-test accuracy gap, the LSTM model is more prone to overfitting training data. The first future work item below would most benefit the LSTM model.

6.1 Future Work

- Better control of overfitting, especially via data augmentation based on the measurement errors `flux_err` and `hostgal_photoz_err`, or L_1 or L_2 regularization.
- Better pre-processing of the `flux` data. For example, fitting a supernova-specific parametric model, or using a more sophisticated representation of irregular time series data, such as a dm-dt grid [15]. This grid plots the incidence of (Δ magnitude, Δ time) pairs between all pairwise measurements.
- Other neural network architectures, such as ResNets, Inception, and multi-layer RNNs.

7 Contributions & Code

All original work was done by Ryan Gao. Source code for this project can be found at `https://github.com/rygao/Stanford-CS230-Project`.

References

[1] The PLAsTiCC team and Tarek Allam Jr. and Anita Bahmanyar and Rahul Biswas and Mi Dai and Lluís Galbany and Renée Hložek and Emille E. O. Ishida and Saurabh W. Jha and David O. Jones and Richard Kessler and Michelle Lochner and Ashish A. Mahabal and Alex I. Malz and Kaisey S. Mandel and Juan Rafael Martínez-Galarza and Jason D. McEwen and Daniel Muthukrishna and Gautham Narayan and Hiranya Peiris and

Christina M. Peters and Kara Ponder and Christian N. Setzer and The LSST Dark Energy Science Collaboration and The LSST Transients and Variable Stars Science Collaboration, “The Photometric LSST Astronomical Time-series Classification Challenge (PLAsTiCC): Data set,” 2018, *arXiv*, abs/1810.00001.

[2] Diederik P. Kingma and Jimmy Ba, “Adam: A Method for Stochastic Optimization,” 2014, *CoRR*, abs/1412.6980.

[3] Paszke, Adam and Gross, Sam and Chintala, Soumith and Chanan, Gregory and Yang, Edward and DeVito, Zachary and Lin, Zeming and Desmaison, Alban and Antiga, Luca and Lerer, Adam, “Automatic differentiation in PyTorch,” 2017, *NIPS-W*.

[4] Karen Simonyan and Andrew Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” 2014, *CoRR*, abs/1409.1556.

[5] C. Wolf, K. Meisenheimer, and H.-J. Röser, “Object classification in astronomical multi-color surveys,” 2001, *Astronomy & Astrophysics* **365**(1):660–680.

[6] Nicholas M. Ball, Robert J. Brunner, Adam D. Myers, and David Tcheng, “Robust Machine Learning Applied to Astronomical Data Sets. I. Star-Galaxy Classification of the Sloan Digital Sky Survey DR3 Using Decision Trees,” 2006, *The Astrophysical Journal* **650**:497–509.

[7] Sepp Hochreiter; Jürgen Schmidhuber, “Long short-term memory,” 1997, *Neural Computation* **9**(8): 1735–1780.

[8] James Newling, Melvin Varughese, Bruce A. Bassett, Heather Campbell, Renée Hlozek, Martin Kunz, Hubert Lampeitl, Bryony Martin, Robert Nichol, David Parkinson, Mathew Smith, “Statistical Classification Techniques for Photometric Supernova Typing,” 2010, *arXiv*, abs/1010.1005.

[9] N. V. Karpenka, F. Feroz, M. P. Hobson, “A simple and robust method for automated photometric classification of supernovae using neural networks,” 2012, *arXiv*, abs/1208.1264.

[10] Michelle Lochner, Jason D. McEwen, Hiranya V. Peiris, Ofer Lahav, Max K. Winter, “Photometric Supernova Classification With Machine Learning,” 2016, *arXiv*, abs/1603.00882.

[11] A. Möller, V. Ruhlmann-Kleider, C. Leloup, J. Neveu, N. Palanque-Delabrouille, J. Rich, R. Carlberg, C. Lidman, C. Pritchet, “Photometric classification of type Ia supernovae in the SuperNova Legacy Survey with supervised learning,” 2016, *arXiv*, abs/1608.05423.

[12] Mi Dai, Steve Kuhlmann, Yun Wang, Eve Kovacs, “Photometric classification and redshift estimation of LSST Supernovae,” 2017, *arXiv*, abs/1701.05689.

[13] Tom Charnock, Adam Moss, “Deep Recurrent Neural Networks for Supernovae Classification,” 2016, *arXiv*, abs/1606.07442.

[14] J. Guy, P. Astier, S. Baumont, D. Hardin, R. Pain, N. Regnault, S. Basa, R. G. Carlberg, A. Conley, S. Fabbro, D. Fouchez, I. M. Hook, D. A. Howell, K. Perrett, C. J. Pritchet, J. Rich, M. Sullivan, P. Antilogus, E. Aubourg, G. Bazin, J. Bronder, M. Filiol, N. Palanque-Delabrouille, P. Ripoche and V. Ruhlmann-Kleider, “SALT2: using distant supernovae to improve the use of type Ia supernovae as distance indicators,” 2007, *Astronomy and Astrophysics* **466** (1):11–21.

[15] Ashish Mahabal, Kshiteej Sheth, Fabian Gieseke, Akshay Pai, S. George Djorgovski, Andrew Drake, Matthew Graham, the CSS/CRTS/PTF Collaboration, “Deep-Learnt Classification of Light Curves,” 2017, *arXiv*, abs/1709.06257.