CS230

# Strongly-lensed quasar selection based on both multi-band tabular data

**Ji Won Park**
Department of Physics
Stanford University
jwp@stanford.edu

## Abstract

This paper presents deep neural networks (DNNs) applied to the binary classification problem of separating lensed quasar systems from everything else in a severely class-unbalanced, sparse dataset consisting of various multi-band properties measured over time. Several network architectures were attempted, including a shallow 1D convolutional neural network (CNN), long short-term memory (LSTM), 2D CNN, and an LSTM autoencoder simultaneously trained with the softmax classifier from the bottleneck layer. The autoencoder performed the best, achieving a final validation accuracy of 86%.

## 1 Introduction

Quasars are regions of high luminosity hosted at the centers of some galaxies. When a distant quasar is aligned closely with a foreground object such as a galaxy, the light from the quasar becomes gravitationally bent, in effect creating multiple images of the quasar as viewed from the Earth, as illustrated in Figure 1. This phenomenon is called strong gravitational lensing and we say that the quasar is (gravitationally) lensed. We also refer to the foreground object simply as a lens, and the aggregate system consisting of the lens and the quasar images as a lensed system.

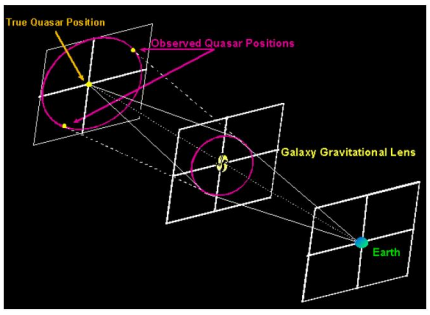


Figure 1: Illustration of a gravitationally lensed quasar. (Cornell Astrophysics, 2018)

Lensed quasars invite a host of cosmological applications. Most relevantly for this paper, however, the quasars can be used in what's called "time delay cosmography" (Treu, Marshall 2016). Monitoring the light curves, or the flux over some period of time, of a lensed quasar allows us to measure the time delay from the arrival of one quasar image to another. The time delays, in turn, can be used to constrain our measurement of the Hubble constant ($H_0$).

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

This endeavor requires a significant sample of lensed quasars, and suffers from the challenge that lensed quasars are very rare and difficult to find. The upcoming large-scale astronomical survey, called the Large Synoptic Survey Telescope (LSST), is predicted to detect billions of objects during its decade-long operation starting in year 2022. Of these, only around 8,000 will be lensed quasars (Gavazzi et al., 2014). Thus the challenge can be framed as a binary classification, i.e. lensed quasar systems (positive) vs. everything else (negative), of a severely class-unbalanced dataset. The LSST releases what is called a *source table* which stores important properties – such as shape, size, and brightness in five wavelength bands – of each observed object at various points in time.

The experiments introduced in this paper is part of an ongoing tools development for the LSST data analysis (Park, 2018a) All the network architectures attempted accept as input a time sequence of properties measured from an object and classifies the object as a lensed quasar system or not. Each object had 128 time slices of 55 properties (11 properties measured in 5 wavelength bands). The properties were hand-picked from a larger set of properties in the LSST source table.

It should be noted that this project is only an initial attempt at a method for classifying quasars based on multivariate time series data. LSST source tables are not yet available, as the telescope won't see light until 2022. Before beginning the deep learning portion of this project, I first had to complete the development of SLRealizer, a framework which emulates an LSST source table (Park, 2018b). Then I used SLRealizer to realize two mock LSST source tables, one consisting of lensed quasars and another of regular galaxies. Only then were the two source tables combined and processed to make an input for our DNNs. Thus any success of this method must be interpreted with caution, as I had control over both the data generation and model-training. It is possible that the distribution of lenses and non-lenses in my datasets were so different that the classification task was "too easy" to be applicable to real data, or that they were so similar that the Bayes error for the task is too high to meaningful. Details of datasets that went into the creation of the LSST source tables are in Section 4: Dataset and Features.

## 2 Related work

Quasars are not the only objects in the sky whose time variability distinguishes them from other objects. There is a wealth of literature on classifying the light curves, or the brightness-over-time plot, of stars. Previous attempts in astronomy are instructive because telescope data on the whole share three features: they are (1) noisy due to propagated errors from the optical system and the atmosphere; (2) sparse because no object is imaged by a telescope at all times and the frequency of imaging depends on the telescope schedule; and (3) class-imbalanced because, often, events of scientific interest such as planet transits happen very rarely. All of these apply to my quasar data.

Hinners et al. attempted a binary classification of stellar light curves into those with a planet transit and those without. (Hinners, Tat & Thorp, 2017) They report little success (close to random guessing) with representation learning using LSTM and attribute this to all three factors mentioned above. But they find that classification accuracy improves significantly when employ the Synthetic Minority Over-sampling Technique (SMOTE) (Chawla et al., 2002) to oversample the minority class (the transit class) and use classical machine learning algorithms such as SVM and Random Forest to the boosted dataset. One drawback of SMOTE, however, is that features must be averaged across time, as it cannot be used on raw time series.

Nevertheless, applying classical machine learning algorithms to the time-averaged properties can be instructive. Before I took on this project, Kim et al. hand-engineered some quasar features and ran the Random Forest algorithm on the time-averaged properties of quasars (Kim et al., 2018). One advantage of random forest is that it can output a feature importance ranking. This exercise gave us a sense of which features will be highly activated. Also, it guided in the process of excluding non-lenses with extreme properties so that the classification task was not too easy.

Naul et al. seeks to classify light curves into various star classes (Naul et al., 2018). They address the uneven sampling problem by passing the light curves through an RNN autoencoder. The autoencoder takes as input the measurements of various features as well as the sampling times (more accurately, the difference between consecutive sampling times) and the error on the measurements. It serves to embed the light curves as a fixed-length vector. Moreover, the error on the measurements are input to the loss function so penalty on high-noise measurements can be downscaled by the size of the error. They report an impressive, near-perfect classification accuracy.

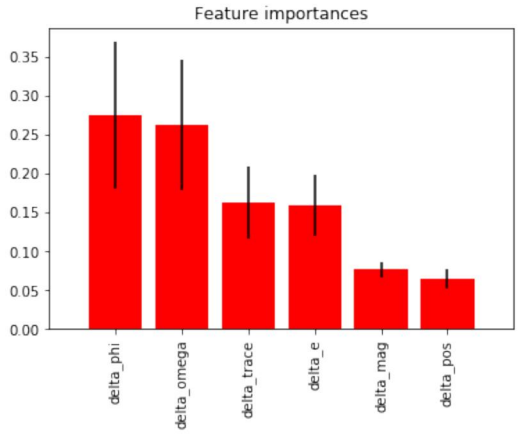


Figure 2: The time averaged object properties were passed through a random forest classifier for an analysis of feature importance. The bars correspond to the difference between the bluest band and the reddest band in the orientation angles, rotation vector, size, ellipticity (shape), brightness, and positional offset (Kim et al., 2018)

## 3 Dataset and Features

This project was intended as a proof of concept for a tools R&D for the LSST, so the dataset came not from real lensed quasars but simulated ones. The number of positive examples was 15631, and negative examples 20000 which resulted in a fairly even ratio of 0.78:1. The total dataset was split in a 8:2 ratio between training and validation sets, and no test set was assigned due to lack of data.

The framework which emulates the LSST source table, SLRealizer (Park, 2018b), took as input two catalogs to generate the source table of positive examples. One was a mock quasar catalog rendered by (drphilmarshall, 2018) and another was a catalog of observational history called "Twinkles" (LSSTDESC, 2018) which lists the observation conditions – such as sky brightness, wavelength band used (the telescope only uses one band at a given time), and the degree of atmospheric turbulence – as at each point in time. There were 128 time slices representing irregularly sampled observations in a span of approximately 2 years. For each object and for each point in time, SLRealizer computed what the 11 properties of the object would be as viewed through each of the five wavelength bands. The 11 properties were all floating-point values. I list the notable ones here:

- Size : more specifically, the trace of the second moment matrix of the object
- Degree of ellipticity : in other words, the degree of angular distortion
- Orientation angle : the angle of above distortion
- Time since last observed : the time in days since the object was last sampled
- Flux : a value related to brightness
- Degree of atmospheric turbulence : this is related to the error on the size, as more turbulence tends to stretch the image of the object

## 4 Methods

Four architectures were attempted: a 1D CNN where the convolution took place along the time axis and the features served as channels, a 2D CNN where the convolution took place along both the time and features axes, a simple LSTM, and an LSTM autoencoder.

For the 1D CNN and simple LSTM, I loosely followed the implementation by Burak Himmetoglu (Himmetoglu, 2018). My modification on the 1D CNN has seven 1D convolution layers (with stride 1 and the last 5 layers alternated with 1D max pooling with stride 2) and two fully-connected layers before a two-class softmax classification.

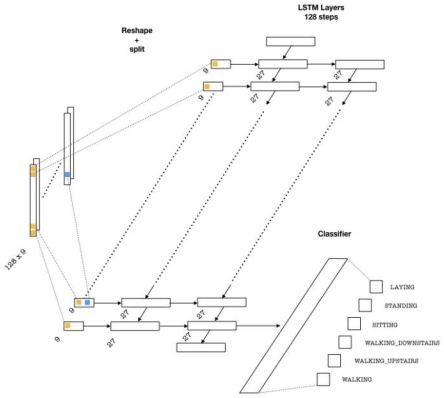


Figure 3: Schematic of the LSTM network. (Himmetoglu, 2018)

At each iteration, for each object, the LSTM network takes in 128 time slices of 55 floating-point properties. Figure 1 shows a schematic of the architecture (Himmetoglu, 2018). It shows two layers, but I used four for my application, as my features were more complex. It also shows the LSTM size as 27, but I used 165 because it should be a few times greater than the number of channels (55 in my case). I also used dropout with keep probability 0.5 and gradient clipping to prevent exploding gradients.

For the 2D CNN, I downsampled the time series from 128 to 65 via simple averaging and gathered two more features per band, to make $13 \times 5 = 65$ features. This yielded a 64-by-65 data array (64 on the time axis, 65 on the features axis) which could be passed through a 10-layer 2D CNN. The 2D CNN consisted of eight 2D convolution layers (all with stride 1 and the last 4 alternated with 2D max pooling with stride 2) and 2 fully-connected layers.

The autoencoder architecture was taken from the work of Naul et al. (Naul, Bloom, Perez & van der Walt, 2017). The only significant difference between their architecture and the one implemented in this paper is the input vector size, which was 200 for Naul et al., but 128 for our application. The output vector size was kept at 200 because the sampling intervals were highly irregular, and we wanted the network to extrapolate in the sparsely sampled portions of the time sequence. The autoencoder is a bidirectional gated LSTM; the encoder has size 64 and decoder size 2. The bottleneck layer has size 8. Following their implementation, I also applied dropout with 75% keep probability.

## 5 Experiments/Results/Discussion

The goal is to identify as many non-quasars as possible. But more important than minimizing false positives is to have little to no false negatives. This is because lensed quasars are very rare; we would not want to miss already-precious lensed quasars in the real application with LSST data. In other words, we can afford to misclassify non-quasars as quasars but not the other way around. Given these specifications, we penalize false negatives more harshly than we do false positives. This is achieved by using a cross-entropy loss as normal except with weighted classes. We define weights for Class 1 (quasar) as 2.0 and the weights for Class 0 (non-quasar) as 1.0 and weight the loss function accordingly, so that the loss contribution from quasars is higher than that of non-quasars.

At a given iteration, the loss was

$$Loss_{total} = \frac{1}{N} \sum_{i=1}^{N} CE \left( w(y^{(i)}) y_1^{(i)}, \hat{y_1}^{(i)} \right)$$

where $CE$ stands for cross entropy loss, $N$ is the number of examples in the batch, $y$ are the labels, $\hat{y}$ are the scores, and $w$ is the aforementioned class weighting such that $w(0) = 1.0$ and $w(1) = 2.0$. The sum iterates over all examples in the batch, denoted $i$.

Training was performed in a batch configuration, where the batch size was 600. Batch normalization was not used because the dataset was sparse and carried many null values. The null values were replaced with an arbitrary unique floating-point number -9999.0 so batch normalization would have introduced an artifact in the intermediate feature maps.

The initial learning rate was set to 0.01 and allowed to exponentially decay every 5 epochs with a decay parameter 0.1.

Since there is some class imbalance in our dataset, it was useful to define an accuracy measure other than a simple ratio between predicted correct and predicted total, i.e. (true positives + true negatives)/(positives + negatives). So, following (Hinners, Tat & Thorp, 2017), we defined the "balanced accuracy measure" as

$$accuracy_{balanced} = \frac{1}{2}\left(\frac{TP}{P} + \frac{TN}{N}\right).$$

Table 1: Performance comparison

| Result | 1D CNN | LSTM | 2D CNN | LSTM autoencoder |
|---|---|---|---|---|
| Training Accuracy (%) | 74.0 | 85.2 | 68.0 | 92.1 |
| Validation Accuracy (%) | 67.4 | 82.8 | 61.9 | 89.0 |
| Training Balanced Accuracy (%) | 68.5 | 81.3 | 64.7 | 88.8 |
| Validation Balanced Accuracy (%) | 63.9 | 79.0 | 55.0 | 85.8 |
| Proportion of FN in Validation (%) | 15 | 12 | 28 | 5 |
| Training Time (on NVIDIA GTX 1070 8GB) | 4 hr | 4 hr | 7 hr | 13 hr |

## 6 Conclusion/Future Work

I have presented the results of several experiments in using DNNs to classify quasars from other astronomical objects. A simple 2D CNN (first column of Table 1) performed the poorest, at only slightly better than guessing accuracy, most likely because it was not temporally aware. There is no reason to assume translational invariance across wavelengths or object properties, so the concept of convolution does not make physical sense along the features axis. Moreover, the features axis is sparse (at a given time slice, only one-fifth is filled) so convolutions across the features axis may have introduced high errors to the kernel.

The autoencoder-classifier performed the best, as the model could both express the input as a fixed-length vector representation and optimize for classification. This turned out to be optimal for a sparse yet irregularly sampled time series. Although omitted in this paper, the decoded output was instructive in seeing how the missing time steps were extrapolated by the network.

If I had more time, I would like to use the generative model introduced in (Dobler et al., 2015) to create more flux variations on the quasar examples. This would have the effect of regularizing the model. The bias could be reduced by making the model more complex. I have decided on a shallower architectural design due to GPU constraint, but using very deep state-of-the-art architectures such as ResNet would be help model the complex and nonlinear quasar features. If I had more time to gather more details about the data, it would be worthwhile to get the errors on all the measurement values (features) so that the errors could be fed into the autoencoder loss. Doing so could allow the model to

handle the measurement error internally, e.g. figure out for itself which time fluctuations are due to noise and which are significant and intrinsic to quasars.

## 7 Contributions

I thank Phil Marshall, Staff Scientist at SLAC National Accelerator Laboratory, for his mentorship on the project. Particularly helpful discussions included those on modeling the time variability of strongly-lensed quasars and interpreting the LSST table format. Mike Baumer, a Ph.D. student in the Stanford Physics Department, provided the raw catalog of negative examples (non-lenses) and made initial queries to exclude extreme examples. Jenny Kim, an undergraduate Physics major, had implemented a prototype of the data generation code before I took on this project.

## References

Astro.cornell.edu. (2018). Quasar as a Gravitational Lens. [online] Available at: http://www.astro.cornell.edu/academics/courses/astro201/g_lens_qso.htm [Accessed 11 Jun. 2018].

Chawla, N., Bowyer, K., Hall, L., & Kegelmeyer, W. (2002). SMOTE: synthetic minority over-sampling technique. Journal Of Artificial Intelligence Research, 16(1), 321-357. Retrieved from https://dl.acm.org/citation.cfm?id=1622416

Dobler, G., Fassnacht, C., Treu, T., Marshall, P., Liao, K., & Hojjati, A. et al. (2015). STRONG LENS TIME DELAY CHALLENGE. I. EXPERIMENTAL DESIGN. The Astrophysical Journal, 799(2), 168. doi:10.1088/0004-637x/799/2/168

drphilmarshall. drphilmarshall/OM10. (2018). GitHub. Retrieved 11 June 2018, from https://github.com/drphilmarshall/OM10

Gavazzi, R., Marshall, P., Treu, T., & Sonnenfeld, A. (2014). RINGFINDER: AUTOMATED DETECTION OF GALAXY-SCALE GRAVITATIONAL LENSES IN GROUND-BASED MULTI-FILTER IMAGING DATA. The Astrophysical Journal, 785(2), 144. doi:10.1088/0004-637x/785/2/144

Himmetoglu, B. healthDataScience/deep-learning-HAR. (2018). GitHub. Retrieved 11 June 2018, from https://github.com/healthDataScience/deep-learning-HAR

Hinners, T., Tat, K. and Thorp, R. (2017). Machine Learning Techniques for Stellar Light Curve Classification. Arxiv.org. Retrieved 11 June 2018, from https://arxiv.org/abs/1710.06804

Kim, J. et al. (2018). SLRealizer: LSST Catalog-level Realization of Gravitationally-lensed Quasars.

LSSTDESC. LSSTDESC/Twinkles. (2018). GitHub. Retrieved 11 June 2018, from https://github.com/LSSTDESC/Twinkles

Naul, B., Bloom, J., Pérez, F., & van der Walt, S. (2017). A recurrent neural network for classification of unevenly sampled variable stars. Nature Astronomy, 2(2), 151-155. doi:10.1038/s41550-017-0321-z

Park, J.W. jiwoncpark/deep-qso. (2018a). GitHub. Retrieved 11 June 2018, from https://github.com/jiwoncpark/deep-qso

Park, J.W. jiwoncpark/sl-realizer. (2018b). GitHub. Retrieved 11 June 2018, from https://github.com/jiwoncpark/sl-realizer

Treu, T., & Marshall, P. (2016). Time delay cosmography. The Astronomy And Astrophysics Review, 24(1). doi:10.1007/s00159-016-0096-8