Classifying Astronomical Objects Using Machine and Deep Learning Methods

Matthew Bundas
Department of Computer Science
New Mexico State University
Las Cruces, USA
bundasma@nmsu.edu
https://github.com/bundasma/CS579_project

***Abstract*—This project involves using machine learning and deep learning methods to classify astronomical objects given time-series observations and positional data relating to sources. Specifically, a Kaggle competition titled PLAstiCC Astronomical Classification was followed [4]. Simulated data of a soon to be opened Vera C. Rubin Observatory is given, and competitors are asked to create a model which can provide insight into which of 15 classes the objects may belong to. Fourteen of the fifteen total classes are present in the training data, while one class is present in the test data, but not the training data. Competitors are evaluated based on how accurate their predictions are using a weighted log loss formula. The solution to this competition in this project is a deep neural network. This deep neural network learns about hundreds of features engineered from the original data which attempt to capture astronomical properties which distinguish classes. These features include statistical calculations applied to raw flux values with consideration of both time and wavelength of observation, as well as intrinsic positional features. The structure of the deep neural network includes several fully connected layers making use of dropout and batch normalization concepts. While able to learn a fair amount from the training data, the model struggled to make predictions regarding object classification. The model in this project receives a competition score of 1.53. This score is better than the average score, but far from the winning scores.**

I. Introduction

Understanding the vast universe beyond our home of Earth is a fascinating and challenging task. Much of our understanding comes through the use of telescopes, where light is collected and focused onto a sensor where it can be measured, and later analyzed. Since their development, telescopes have only become more impressive both in terms of their complexity and raw size.

One observatory currently under construction set to be completed in 2021 is the Vera C. Rubin Observatory. The Vera C. Rubin Observatory (VCRO) is a cutting edge survey telescope, meaning that its function is to scan vast patches of sky on a given night rather than spend a longer period of time on a handful of objects. As a result of it being a survey telescope, its world class 8.4m diameter mirror, and 3.2 Gigapixel camera, the VCRO produces 20-40 Terabytes of data every night it is in operation [1]. This is of course far too much data for humans to process on their own.

To explore how computers might best be able to handle processing the data, the VCRO enlisted the machine learning and data science community by creating a Kaggle Competition. In this competition, competitors are given simulated VCRO data, both time series and static, and asked to create a model which can give insight into the classification of objects. My solution is a deep neural network whose features are engineered from the raw data sets provided. While I do have a bit of formal background knowledge related to astronomy, this was my first time exploring the field of deep learning and creating a deep neural network. This project was guided by my previous astronomy knowledge, scientific papers[6,7] which have explored astronomical classification, comprehensive documents given by the VCRO [1], and discussions between fellow Kaggle competitors [2-5].

Much time was spent feature engineering, as abundant meaningful features are essential in distinguishing classes of objects from one another. Classes of objects can be distinguished through observation by consideration of many factors. There is wide variation among different types of objects such as stars, galaxies, supernova etc. The most obvious is the range of which their brightness operates. By nature some phenomena such as supernova are much more bright than other objects like main sequence stars. Astronomical objects also differ in the way their brightness changes over time, some stars retain the same brightness over the time span of thousands of years, while others can vary orders of magnitude in a matter of minutes. Different classes of objects also vary in the wavelength of light their characteristics are expressed. Depending on the phenomena at hand, mechanisms within them cause different types of light to be emitted compared to others. All of these ideas are essential when distinguishing astronomical object classes. Luckily for this project, the data set given by VCRO contains all of the information needed to obtain values which represent these ideas and allow for a neural network to distinguish between objects.

However, due to the shear amount of data needed to be processed, the features used in the model must be obtainable in a relatively quick manner, which was a key part of this project. While some features like period of variation may provide good insight into which class an object might belong to, this is a

calculation which takes a handful of seconds to calculate for a single object. Considering millions of objects need to be classified, having a feature which is calculated on the order of seconds is not practical. Because of this idea, this project stuck to features which can be extracted in nearly instant time for individual objects. In the remainder of this paper, the methods used to carry out this project are shown, results presented and analysed, and finally discussed and concluded.

II. Methods

Developing a large scale, complex neural network is not an easy task. It requires methodical production, testing and tweaking. The neural network used in this project started from the very basics, beginning with a neural network of just one hidden layer with a few dozen features. This simple neural network evolved into a more sophisticated and complicated neural network which over time saw increased performance. The first iteration of the neural network began with just a few dozen features, and in its final form makes use of 229 features. These features attempt to capture the properties of the astronomical objects present in the data set which are productive in terms of distinguishing them, talked about in the Introduction section.

Features which were chosen to be tested came from a combination of sources, including assumptions from background astronomy knowledge, ideas presented in the ”starter kit” VCRO provided [1], papers which have looked at similar problems [6,7], and also my fellow Kaggle competitors [2-5]. Features were chosen to be rejected or continued to be included based on their merits, and also how they behaved in the model both in terms of time-efficiency and prediction performance. The direct output of the constructed model is a csv file containing the probability that the object belongs to each of the 14 classes found in the training set. However, one class, class 99, was in the test set but not found in the training set. This was an aspect of deep learning predictions which was interesting to tackle, as the model needed a solution to predict the probability of a class which it did not know about from training. There are several ways to handle this class 99 prediction. This paper's solution involves considering how confident the model is about it's non-class 99 predictions, where the less confident it is, the heavier weight of probability class 99 is given. During development, the model and workflow was evaluated using the same methods the Kaggle competition used, that is the log-loss. Neural network methods and data features which were efficient and improved log-loss performance were incorporated in the model, while others were not.

A. Data

This project, and Kaggle competition which it followed, made use of an abundance of data. In total, over 37 GB of data was provided, much of it being test-set data. Each object, whether train or test data, had its information stored in two separate data sets. One data set provided the raw time-series flux observations, where each row was a single observation of a single object. In these rows, the flux observed was given, estimated error in the flux, a 0/1 metric of whether the observation saturated the detector, in what wavelength range of light or passband the observation took place in, and at what time, given in mjd the observation was taken in. Each object had between 100 and 200 individual observations present, taken in 3 separate time ranges/survey-passes in roughly equal proportion of passbands. A sample of this data set is shown in Figure 1. The second data set contained the corresponding metadata for the objects, containing static information about the object, such as its coordinates in the sky and distance from Earth. A sample of this data set is shown in Figure 2. Each object had an object id, which allowed for matching of the data between datasets.

Both the time-series data and metadata provided meaningful information when considering the class of object, although by nature of the problem, most features came as a result of engineering the time-series data. The time-series data is best visualized in terms of light curves which are shown in Figures 3, 4, and 5. In these figures, you can visually get a sense of what the time-series data looks like, and what differences might be present in object classes. Each light curve shows a different class of object, and a very different story. Figure 3 shows the object having steady flux in two of the three observation groups, while in the middle one the source caries wildly in all three passbands. Figure 4 shows an object which varies in all wavelengths in all three observation groups. Figure 5 shows an object where in all observation groups, the source varies is steady in three passbands, but fluctuates in the other three passbands. Also note the difference in flux ranges the different sources operate on. Figure 3's source's flux spans about 2,000, Figure 4 spans about 200, and Figure 5 spans over 30,000. While only three different classes are shown, these ideas apply to all classes of object. Because they are of different type, they have different innate properties, which can be observed and ultimately allow for distinguishing of type. Also note negative flux values. This is because flux values are calculated using differential imaging, so if there ie less flux compared to the reference image, a negative value is measured.

object_id	mjd	passband	flux	flux_err	detected_boo
int64	float64	int64	float64	float64	int64
10	56210.172	1	5.659	3.7	1
10	56210.188	2	21.32	3.245	1
10	56210.203	3	13.28	3.221	1
10	56210.234	4	9.579	3.85	1

Fig. 1. Sample of training time series data.

While having this much data provided plenty of opportunity for feature extraction, it also was challenging to work with. I imagine this was by design with the competition organizers, as this is a core issue of the VCRO and seemingly a major purpose of this competition. It was required that processing of data be efficient, as if it was otherwise, the time taken to run

object_id	ra	decl	gall	galb	ddf_bool	hostgal_specz	hostgal_photoz	hostgal_photoz_err
int64	float64	float64	float64	float64	int64	float64	float64	float64
615	349.046051	-61.943836	320.79653	-51.753706	1	0.0	0.0	0.0
713	53.085938	-27.784405	223.525509	-54.460748	1	1.8181	1.6153	0.55
730	33.574219	-6.579593	170.455585	-61.548219	1	0.232	0.2262	0.0157
745	0.189873	-45.586655	328.254458	-68.969298	1	0.3037	0.2813	1.1523

Fig. 2. Sample of training metadata.

the model on test data faces ballooning. I expect that some of the issues pertaining to the large datasize would be mitigated by VCRO's superior technology and capability of utilizing it.

Data processing in this project was at a high level handles by the python package pandas. The use of aggregation functions were used heavily, where a function such as mean or standard deviation were able to be applied to compiled groups of all observations belonging to a given object. After aggregations, pandas also allowed for features stored in different columns to be used in operations, facilitating the calculation of raw differences and ratios between features. Pandas also made it possible to look at observations grouped by passband, apply aggregations, and finally operations between the aggregated values. After features were extracted from the time-series data, it was merged with the metadata data set, creating a final data set which contained all engineered features and all features present in the metadata.

The data used as part of this project was very much not balanced. This was the case both in terms of the proportion of data in the test vs training sets and proportion of classes within those sets. The training set had information regarding 7848 objects, roughly 60 MB of data total. The total testing data on the other hand had information regarding 3.5 million objects, roughly 30 GB of data in total. This is a very stark imbalance of data, which I think the model struggled with as the variation of objects seen in the training data was much different than in the test data.

With regards to the imbalance within datasets, the proportion of individuals belonging to each class was in many respects not balanced. In the training set, class 90 made up nearly 30 percent of the dataset, classes 52,63 and 16 each made up around 10 percent of the training set, but class 53 was responsible for just 0.38 percent of objects. The proportions of classes was also quite different in the test versus the training sets. Proportions did not match for any classes, and were different by factors of two up to factors of 100 as is the case with class 53.2

Similar to the raw data size, I expect the imbalance was also by design, as the data presented was in full control of the competition creators, since after all the data was all simulated. The data includes phenomena which are consistent with astronomical observations, such as variation in flux and noise due to atmospheric conditions, inconsistent time sampling, and other realistic astrophysical effects [1].

B. Features

Astronomical objects have complicated and often difficult to capture properties. However, this project attempted to extract

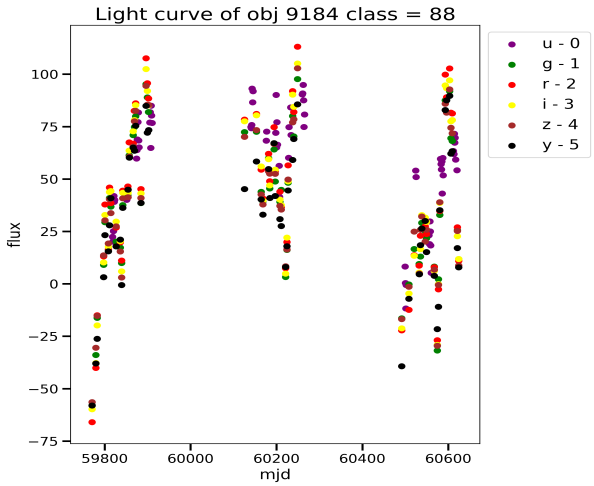

Fig. 3. Sample lightcurve of a class 88 object. Time (mjd) is on X axis, raw flux in on Y axis, color coded by passband of observation

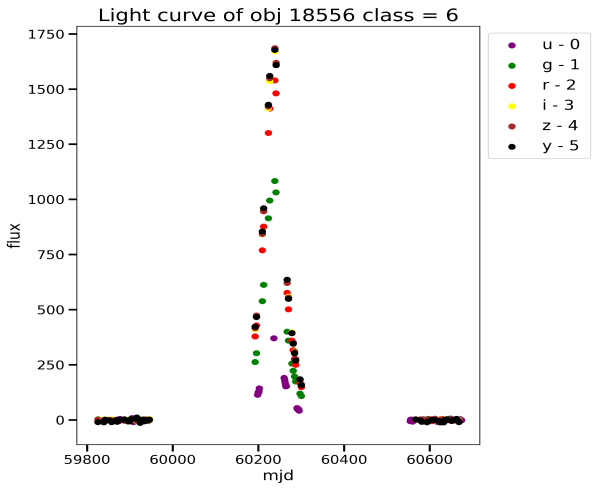

Fig. 4. Sample lightcurve of a class 6 object. Time (mjd) is on X axis, raw flux in on Y axis, color coded by passband of observation

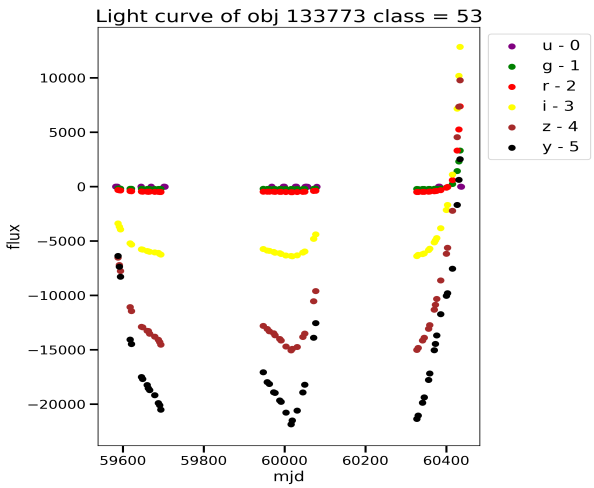

Fig. 5. Sample lightcurve of a class 53 object. Time (mjd) is on X axis, raw flux in on Y axis, color coded by passband of observation

meaningful features from the given data set and allow a deep learning model to learn about them. The given data set has time-series flux information as well as innate information about the object, called metadata. The features attempt to quantify astronomical properties related to an astronomical object's raw flux, rolling average in flux, estimated flux error and consider the passband observations are taken in, and at what time they are taken. Below these different groups of features are described in detail including specifically what they attempt to capture, and how they are engineered.

1) Raw Flux Features: At the core of the number of features used in the model are features related to the raw flux and related values for each object. In each row of the time series data, there are several values given including the flux, error in flux which, will be talked about here. These values provide insight into the absolute brightness detected from the object, but not specifically related to time or passband, making them simpler. For each object, with regards to the flux and estimated flux error a wide range of statistical functions were applied. These include basic calculations such as the mean, max, median, standard deviation, 10/25/50/75/90th percentiles etc. They also include more sophisticated calculations such as the skew, kurtosis, percent amplitude, max/min slope, etc. These more sophisticated calculations attempt to quantify more nuanced properties of the data. Skew and kurtosis attempt to capture the tailing of the data, slopes attempt to capture the biggest/smallest raw jumps in flux between observations, and percent amplitude more the proportion of the largest/smallest jumps.Various ratios and differences between these calculated values were also taken, such as the ratio of max to mean. In addition, a handful of simple calculations were applied to the rolling average of 10 previous observations in hopes that this would be a more consistent and less prone to fluctuation measurement. Roughly 50 features related to raw flux values were used.

2) MJD Features: While not as major, the direct MJD features were still used and provided insightful information about the objects. What time an object is observed at can not provide much information without context, but this competition has plenty of context. Considering VCRO follows patterns and may observe different classes in different ways with respect to time, MJD features can be used to detect patterns which can provide clues for classification. Perhaps this is not as science-based, and a little bit of cheating, but for the purposes of classifications MJD features proved helpful. The max, min and standard deviation of MJD was calculated for each object and used in the model. While not included in the final model, due to the amount of time required to calculate, features capturing the period or frequency which an object varies would also have been useful to include in the model. However, these calculations are much more sophisticated and take several seconds to calculate for each object, and were not practical to include in the model considering how much data needed to be processed.

3) Passband Features: Taking into consideration the specific wavelength of light observations are taken in is essential in classifying astronomical objects. Simply looking at raw flux features without thought about passband ignores the properties common among astronomical objects in the same class. Thus, for this project, passband features were considered heavily and proved to be essential in the creating of the model. Using pandas, the observations for each object were grouped by passband, and their means, maxes, medians and standard deviations were taken. With these values taken, various ratios and raw differences were taken between passbands including between adjacent passbands such as 0 and 1, and non-adjacent passbands such as 1 and 5. Ratios in addition to raw differences were taken because since different astronomical objects operate on such different scales, such as evident by Figures 3-5, the ratios between values is perhaps more important than their raw differences. The goal of these features were to capture properties which may exist within classes of objects where they perhaps may vary in some passbands, but be consistent in others, such is the case in Figure 5. In Figure 5, the object is steady in passbands 0-2 and varies in passbands 3-5. In total, 167 features related to passband were calculated along these lines and used in the model.

4) MetaData Features: While in small quantity compared to the number of features engineered from the time-series data, the features contained as part of the metadata provided meaningful insight into the astronomical objects. The metadata rows contained information mostly regarding the object's position in the universe relative to Earth, such as the distance from Earth and coordinates in the sky. These properties are useful, as much can be inferred solely from an object's position. The distance value alone can help determine whether an object is within our galaxy or outside our galaxy. Because they are so relatively dim, very common objects such as main-sequence stars can only be observed directly if they are in our own galaxy. Thus, if an object is observed to be outside of our galaxy, it is safe to assume that this object is not a main-sequence star, but must be something more impressive and brighter such as a galaxy or supernova. Similary, with regards to coordinates in the sky, because of the structure of the galaxy being disc-shaped, stars within our galaxy have a very high likelihood to lie along the galactic plane. Thus, if an object is observed based on its coordinates to not lie in the galactic plane, then there is a high likelihood that the object is not located in the galaxy and is not a main-sequence star. Similarly, if it does lay in the galactic plane, you may be accurate in giving higher probability that it is an object within our galaxy and is a star-type object. In total, 9 metadata features were used in the model for each object.

C. Neural Network

Considering the high number of features present in the processed data set, a relatively sophisticated fully neural network model was required to learn about the features. In total in the fully connected neural network, created with tensorflow, used to carry out the project has over 550,000 trainable parameters. These are present in the model's five fully connected layers, where each one makes use of batch normalization and dropout.

The number of neurons in the first hidden layer is 512, following another layer of 512, then a layer with 256, a layer with 128, and finally the output layer with 14 neurons, one for each class. In each hidden layer, sigmoid activation was used and the dropout rate was 0.40. In the final output layer, a softmax activation was used which allowed the model to output the probability the object belonged to each class rather than just the most probabilistic class.

Before being fed to the model, the data is normalized using tensorflow's keras normalize function. This function makes use of L2-Normalization to transform the inputs to be between 0 and 1 to ensure smoother learning. Using dropout and batch normalization both help to ensure that each feature has its opportunity to be learned about and incorporated into the model. Batch normalization works by normalizing the output of hidden layer neurons. This functions to allow each layer to be less reliant on the previous layer and learn by itself. Dropout works by essentially turning off a percentage of neurons in each layer while it is learning so that other neurons can learn on their own and hopefully make more generalizations. This helps address the problem of over-fitting, which is easy to do when working with a neural network like this. Overfitting was something that required balancing throughout the project, which was mitigated in part by using dropout, batch normalization and hyper parameter selection.

Neural networks attempt to learn about the data by adjusting parameters and minimizing a loss function. In this model, the loss function used to train was similar to the loss function used as the competition performance metric [2], with the goal of maximizing competition outcome. Training was performed using a k-folds validation style method. This involved training separate models 5 times, each with different samples from 80/20 train/validation splits. Each neural network was trained using 150 epochs and a batch size of 20. The best of the 5 trained models was chosen based on an assessment of its validation and training loss. The chosen model from this method was used to make predictions on the test sets.

D. Making Predictions - Class 99

As mentioned previously, there was one class of object, class 99 which was present in the test data sets, but not in the training data sets. This posed an interesting problem as to how to predict a class which the model knows nothing about. In the solution for this project, the model made predictions under the assumption that there were only the 14 classes present in the training data, and then class 99's probability was calculated based on those predictions. Heavier weight was given to class 99 in cases where the model was less sure about it's predictions of the other 14 classes. Although many solutions were hypothesized and tested, the final calculation involved summing the two highest given probabilities, subtracting that from one, and dividing by 3, according to Equation 1.

$$class_{99}prob = \frac{1 - (maxprob + secondmaxprob)}{3} \quad (1)$$

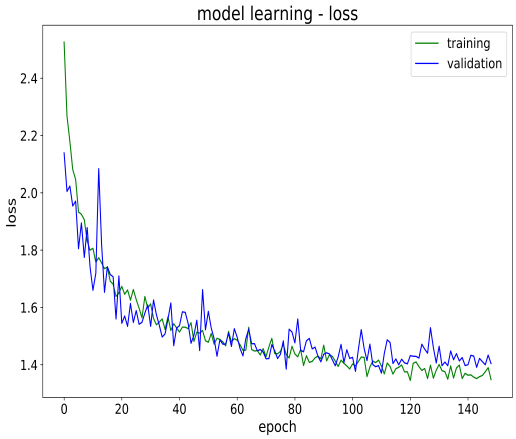

Fig. 6. Learning curve of utilized Deep Neural Network Model. Shows evolution of loss function given at each epoch.

E. Evalutation

Evaluation of the model was primarily performed using the same log loss function used as the competition metric. Luckily for this project, since the competition is formally over, test data containing the true labels was provided to the competitors, where previously the labels were not given. This allowed for the calculation of my competition score using a fellow competition member's script [3].

The log loss formula used is given in Figure 7. This formula takes into account the probability assigned to the actual class, as well as weights which the competition organizers assigned to each class.

$$\text{Log Loss} = -\left(\frac{\sum_{i=1}^{M} w_i \cdot \sum_{j=1}^{N_i} \frac{y_{ij}}{N_i} \cdot \ln p_{ij}}{\sum_{i=1}^{M} w_i} \right)$$

Fig. 7. Log loss function used as competition metric. N is number of objects, M is the number of classes, y is 1 if the true class of i is class j, 0 otherwise, p is the predicted probability that i belongs to j

Outside of the log-loss calculation, during development, the model was evaluated using training and validation performance using the training set, such as shown in Figure 6. While not a perfect one-to-one, in general, models which performed better on the training data performed better on the test data, so this was often more practical than running test data through the model.

III. Results

Results of the model were determined similar to the competition as described in the Evaluation section above. After being trained, the model was fed testing data, whose output is probabilities for each class, and was evaluated using the log-loss formula shown in Figure 7. Because of computational

constraints, a limited but still significant test set was used. This test set contains roughly 350,000 objects, still a very large sample size. Using this test set, the log-loss score was calculated to be 1.53 when taking into consideration class 99, and 1.35 when not considering class 99. To put this in context, this score is slightly above average among fellow competitors, but certainly far from competition winners. The average score in the competition was 1.75, with winning scores being around 0.7. Also, notice a gap between the model's score when considering class 99 and when not considering class 99, indicating difficulty classifying class 99 objects.

```
loss with 99: 1.5340482796934924
loss without 99: 1.3487286414236859
```

Fig. 8. Final competition scores, calculated with formula shown in Figure 7, with script from fellow Kaggle Competitor [3]

To give a better sense of prediction accuracy, a confusion matrix was constructed and can be seen in Figure 9 [5]. This demonstrates for each true label, what was the most likely predicted label by the model. It portrays interesting results, where the model has high (greater than 0.8) accuracy for a handful of classes such as 16, 53, and 88, but struggles with other classes. The constructed model struggled with classes 42,90, 52 and 92, with prediction accuracy below 0.3. Other classes were predicted at a rate mediocre at best, with accuracies around 0.50.

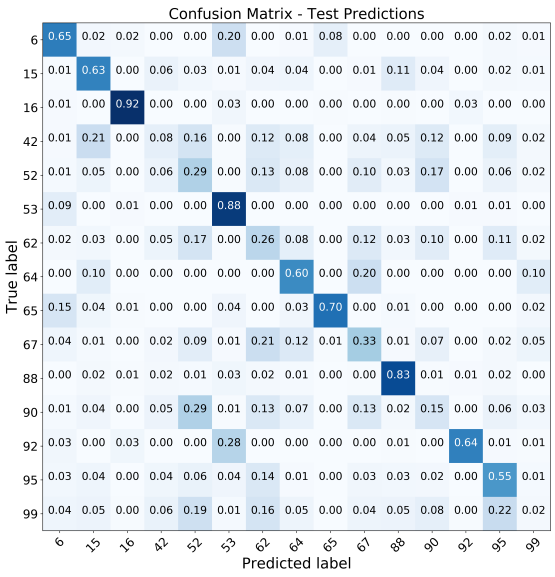

Fig. 9. Confusion matrix demonstrating predictions made by model for each class of object

IV. Conclusion and Discussion

Overall this project was successful in that a neural network was created which was able to make use of engineered features to give insight into the probability unknown astronomical objects belong to given classes. This project followed a Kaggle competition centered around a state of the art astronomical observatory under construction, the Vera C. Rubin Observatory. Competitors were given simulated time-series data and metadata related to astronomical objects and tasked with creating a model which can classify astronomical objects. Models were evaluated using a custom log-loss function. In this project a neural network was the solution to the competition. A neural network was created with several dense layers each making use of dropout and batch normalization. Features were engineered from the datasets whose goal were to capture the properties common among astronomical objects. These features involved statistical calculations applied to raw flux values, with consideration of time and wavelength of observations.

The constructed model was tested using a testing sample, which had mediocre results. From a competition standpoint, a score of 1.53 was the result, which is better than the average score of 1.75, but far away from winning scores of 0.7. The model excelled in classifying some classes of objects, but heavily struggled in classifying other classes of objects, as evidenced by the confusion matrix shown in Figure 9.

Several factors contribute to the less than ideal results. First, the problem of classifying astronomical objects is an incredibly difficult one. Astronomical objects are complex in nature and imperfect to observe. Combining this idea with complex machine learning and deep learning makes for a tough to crack problem. In addition to this, the competition was designed to have challenging problems which are applicable to the real life observatory. This includes an imbalance of training versus test data, and rare occurrences of some classes.

However, with much effort and skill, many competitors were able to successfully create models which performed well, and were effective in classifying objects. These competitors made use of several machine learning/deep learning concepts, created more complex and nuanced features, and ultimately created more sophisticated models than presented in this project. This goes to show while this competition was difficult, it is possible to be successful. Considering this was my first attempt at creating a neural network model and workflow on this scale, I am content with the results. However, there is plenty of room for improvement with the model, and I believe if I implemented some concepts my fellow competitors used results could be improved.

References

[1] Narayan, G., Apers, M., Dane, S., amp; Reade, W. (2018, August 31). The PLAsTiCC Astronomy "Starter Kit". In Kaggle. Retrieved 2020, from https://www.kaggle.com/michaelapers/the-plasticc-astronomy-starter-kit

[2] Reina, Y. (2018). Multi-weight logloss for keras. Retrieved 2020, from https://www.kaggle.com/c/PLAsTiCC-2018/discussion/69795411103

[3] Zlobin, S. (2019). Results on the unblinded test set. Retrieved 2020, from https://www.kaggle.com/c/PLAsTiCC-2018/discussion/78366614486

[4] PLAsTiCC Astronomical Classification. (2018). Retrieved December 12, 2020, from https://www.kaggle.com/c/PLAsTiCC-2018/overview

[5] Isro, S. (2018, November 02). Simple Neural Net for Time Series Classification. Retrieved 2020, from https://www.kaggle.com/meaninglesslives/simple-neural-net-for-time-series-classification

[6] Hinners, T. A., Tat, K., amp; Thorp, R. (2018). Machine Learning Techniques for Stellar Light Curve Classification. The Astronomical Journal, 156(1), 7. doi:10.3847/1538-3881/aac16d

[7] Richards, J. W., Starr, D. L., Butler, N. R., Bloom, J. S., Brewer, J. M., Crellin-Quick, A., . . . Rischard, M. (2011). On Machine-Learned Classification Of Variable Stars With Sparse And Noisy Time-Series Data. The Astrophysical Journal, 733(1), 10. doi:10.1088/0004-637x/733/1/10