Identification of transients in nearby Galaxies

Aishwarya Penmetcha, Samuel Geiser
azp5566,smj6341

Preface

The first member of Team 4 is Aishwarya Penmetcha. She is a senior majoring in Statistical Modeling Data Sciences and minoring in Planetary Science and Astronomy. Aishwarya is experienced in Python and R programming languages, data visualization, data analysis, and web design. Her current interests are: integrating statistical modeling methods with her research in Game theory. She is also interested in applying machine learning models to classify transients or supernovae that occur in neighboring galaxies.

The second team member of team 4 is Samuel Geiser. He is a 5th year senior majoring in Data Science and minoring in philosophy. Sam has experience building, optimizing, and accessing database systems, data analysis, model building, data integration, visualization, and software development in Java, Python, R, and SQL. He is currently interested in analyzing data for his lab and gathering data on LGBTQ issues and martial arts.

Introduction

At the present time, there has not yet been developed a working machine learning or AI model in the Astrophysics Astronomical Sciences community that can classify transient events such as supernovae, kilonovae, Active Galactic Nuclei, etc., that occur in space. However, there is an abundance of data to work with to answer humanity's burning questions about star formation, and the evolution of stars and planets in galaxies. The purpose of this project is to create a binary classification model that will be able to identify whether or not there is a Supernova that has occurred in nearby galaxies, using image data from the Transiting Exoplanet Survey Satellite, also known as TESS. We found that out of a preliminary data set containing 4946 images about 46 of those images were identified as containing a transient event. We plotted these 46 images and found that 5 of the plots had a strange noise to the light curves, and Dr. Villar confirmed that it was noise due to a nearby star and therefore did not count as a transient event.

Key Words: **Neural Net, Pytorch, Supernovae, Binary Classifier, Causal Pixel Model, Transient Event**

Related Work

The data source that we will be using comes from the images captured by the TESS satellite, however, we are unable to capture the entire nature of the science from the raw images. Since we are interested in the change in luminosity or brightness of the stars, we will be using another python program written by Columbia University's Soichiro Hattori (Hattori et al.). The program is called Tess-CPM which stands for **"Causal pixel Model", this program utilizes statistical methods like principal component analysis to produce light curves** that have a gaussian distribution (soichiro-hattori). In other words, **it performs dimension reduction while also removing noise from the raw image data**. These light curve plots are what we will use to train our model.

Our chosen approach was inspired by the growing popularity of machine learning and AI. Specifically deep learning models for image classification problems. There are a few machine learning models that are able to solve image classification problems. Artificial neural networks like a Multi-Layer Perceptron (MLP) or Convolutional Neural Networks (CNN) are just two examples of such models.

One such toy problem using MNIST data set is a great example of a classification model based on image data to identify and classify different images of numbers (Graff ,2014). MLPs are fully connected layers and use regularization techniques like early stopping, and the learning rate can be adjusted to prevent overfitting. A CNN, also known as a Space invariant artificial neural network (SIANN). It can also be used in image classification. It uses a different regularization method than an MLP. They take into account the hierarchical pattern in data and it also has lower complexity than compared to an MLP. There has also been a recent development of a CNN that is used to search for superflares from pixel-level data from TESS (Tu et al. #). Although we are not using CNNs in our study we may consider replicating the model using a CNN to test the difference in performance between an MLP and a CNN model in the future.

We will be using the library PyTorch to train our first model, there are many toy problems available on GitHub and various other sources that we will consult while building our model (*PyTorch Documentation — PyTorch 1.12 Documentation*).
We will also look at image classification examples.

Methods

We first trained several models using less complicated models including a Logistic Regression, K-Nearest Neighbor, Support Vector Machine, and a Random Forest model to give us a better idea of a baseline. These models are all from the sklearn library and are trained and

tuned on the same data as the main model we made next. Using this information, along with the results from the two trained simple NN models, to decide which model is the most appropriate model to use. To make this assessment we tuned hyperparameters on several models to find their peak accuracy. During this process, we also determined if another metric would better suit this process than accuracy and started relying more on loss in the validation set for tuning hyperparameters. Once we had a trained model that we were satisfied with, we used an existing database of solar system locations in order to gather some real data we used to do a final test of the model that would be more accurate to the potential use case.

We started by training a simple Multilayer Perceptron (MLP) model. An MLP is basically a feedforward artificial neural network, which generates outputs based on the inputs and assigned weights if any. Furthermore, an MLP consists of several hidden layers, for our model we chose to use 3 hidden layers. Each layer consists of a certain number of "neurons" also known as hyperparameters. This basic first attempt saw us using a given method for turning the satellite data into a set of points forming a linear light curve. We also normalized that data and padded the light curves to provide a uniform length.

For this first model, we estimated the number of layers and neurons per layer based on previous information. We chose this method as our starting point because the data transformation was recommended by domain expert Dr. Ashley Villar, who has also provided guidance in this project. Dr. Villar specializes in the subject of supernovae at Penn State University and is currently researching the behavior of different light curves that are produced by these events. She was able to provide us with a simulated training set which was used to train the model. Dr. Villar will also be aiding us by providing a much bigger and more realistic training set in the coming week along with a small sample testing set with real Supernovae light curves which will give us more insight into the model's performance. This model was the type that was used and explained in the main tutorial that we have been learning from (Brownlee).

The following method we implemented involved altering the data by balancing the initial data set we are training off of and tuning the parameters of our multilayer perceptron model. The current dataset we used in the first experiment to train off of was pretty heavily unbalanced, and since this is a binary classification problem that is especially problematic. So in order to fix that issue we decided to create a dataset where the majority classifier is undersampled enough to balance the data. Next, we experimented with the model by creating a dataset where the minority classifier is oversampled (we can generate excess artificial data) for a similarly balanced dataset.

Once we had modified the data we could tune the MLP model. In doing this we tuned two hyperparameters, batch size and number of epochs. We tested across the epoch counts of 10, 20, 30, and 40 and the batch sizes 64, 120, 200, 400, 600, and 1200. The best set of parameters was decided based on which model produced the lowest validation loss. The baseline models were also trained and tuned off of this data. The Logistic regression didn't need much tuning but we set the max iterations high enough to allow for it to run its course and used a balanced class

weight and liblinear solver. For the KNN model the value of k, or the number of nearest neighbors, was tuned over the range 1-21 using the modified data. With the SVM model we tuned both kernel type ('linear', 'poly', 'rbf', 'sigmoid') and penalty (1.0, 0.1, 0.001). Finally, we trained the Random Forest model on the data and tuned n_estimators, the number of trees in the forest (10, 100, 1000), and max_features, the number of features to consider when looking for the best split (‘sqrt’, ‘log2’).

To conclude whether or not a model trained on this generated data could recognize real transients, we had to gather as much real data as we could in order to create a much more realistic and challenging final test for our chosen model. In order to obtain this realistic testing set we used the open-source python program Tess-CPM also known as “Unpopular” (Hattori). This allowed us to obtain detrended or de-noised light curves of real Supernovae events from 2020 along with light curves of nearby galaxies without Supernovae events to create a somewhat realistic data set for testing.

The next steps we took in order to obtain our “realistic” test set were to incorporate data integration methods. We then created a python script that ran on Penn State’s supercomputer, to read data from the Young Supernova Experiment Survey (YSE) which consists of spectroscopic observations taken from ground-based telescopes located in Hawaii. This data set is stored as “.dat” files and the script reads important information such as coordinates or the location of the recorded Supernova event along with the time in Modified Julian Date (MJD). MJD is a continuous measure of time in days since midnight at the start of 17 November 1858, which is specifically used in the field of Astronomy. This information was integrated with host galaxy information that we obtained from the Glade+ galaxy catalog which has various other metadata on galaxies in the universe (Dalya).

Evaluation

Datasets and Models

We are using a data set that is simulated TESS-CPM data. This data is created by preprocessing real background data into light curves using the Causal Pixel Model, representing each pixel through multiple images taken at different times, and then pushing a gaussian distribution into these curves to replicate a supernova. We are then padding these light curves to make them the same size as well as normalizing them.

Then we trained a simple neural network from the PyTorch package on the simulated data to identify these fake supernovae in the simulated data. We want to match or exceed the accuracy of the model that was created in a previous project done by Aish.

Experiments: implementation, results, and analysis.

Our first and very basic implementation of the PyTorch multilayer perceptron model was performing very well on the validation set. It outperformed the prior work done by Aish that we are using as a benchmark. This prior work was training a similar model using Keras and TensorFlow and only achieved about 92% accuracy while our untuned model achieved close to 97%. However, unlike the previous work, this model was trained off of unbalanced data and so evaluating it just by accuracy may not be helpful. So instead we generated a confusion matrix and looked at how many of the positive cases were mistakenly labeled negative. This still showed high levels of performance from our model.

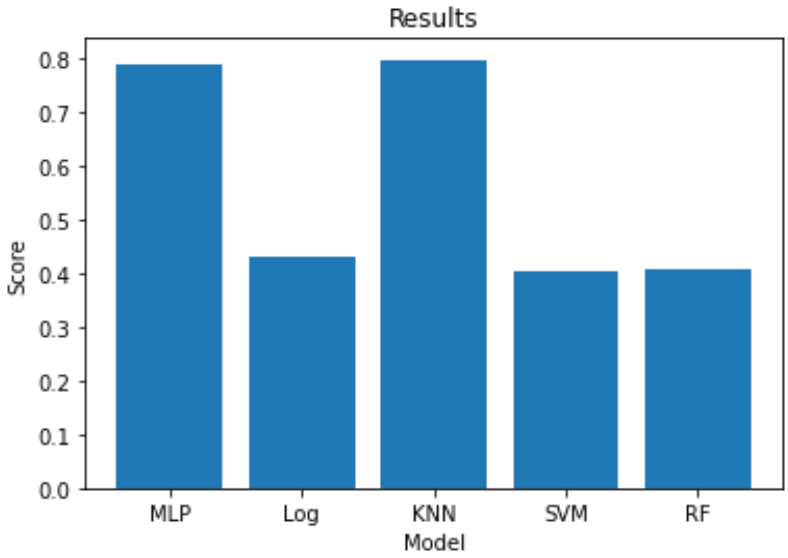
Furthermore, we also trained the model on a more balanced data set and saw that it still performed well and took less time to train albeit noting the hyperparameters such as epochs and learning rate were tuned in Aish's TensorFlow model and naturally took longer to train. Whereas the PyTorch model has epochs set to 10 while no other parameters are tuned at present and therefore made note of that difference between the models' performance.

Apart from this, we have also trained the model on a much bigger simulated data set as provided by Dr. Ashley Villar. The new data set consists of approximately 24,000 simulated light curves, (while the earlier version only contained about 12,000), and saw that in general, it was performing well on the validation set and test set. Aside from that we also trained the model on a balanced or "under-sampled" version of the new data set that was sampled from the new data set. And we saw similar results with accuracy scores of 97% and 96.1%for the unbalanced and balanced data sets respectively.

Upon testing the model on our first test data set we found that the size of the input was much larger. Dr. Villar was able to generate a new training set, which was used to retrain the model, and we obtained an accuracy level of 94.75% with a log loss of 0.175. This was significantly better than the previous models and beat the TensorFlow model as well. We then ran the test set and found that out of 4946 images, the model predicted that 46 of these were showing a strong indication of a transient event. We also set a probability cutoff of 0.4 on distribution, meaning that the predicted set had probabilities higher than 0.4. This was chosen by examining the distribution in a histogram. Upon further investigating these 46 images 5 of them were discarded due to the fact that there was a nearby star that was creating noise in the plot - confirmed by Dr. Villar. Bringing the total number of transient events that were detected to 41.

Once we had obtained real test data from the Young Supernova Experiment Survey, we revisited the basic models that we trained as well as doing further tuning and testing of the MLP model. After some proper tuning the Logistic Regression model was able to reach an accuracy of 97.15% on the validation data but when calculating predictions on the real test data, was only able to reach an accuracy of 43.11%. The SVM model after tuning was able to achieve an accuracy of 98.65% on the validation data but again this dropped substantially when predicting on the real data, with a best accuracy at only 40.12%. The Random Forest model when tuned was able to reach a very high validation accuracy of 99.72%, however this didn't make much of a

difference since its real data accuracy only hit 40.72%. The fourth model, the KNN model, was very surprising after the others and with tuning showed a validation accuracy of 99.82% but more surprisingly showed a high accuracy on the real data, reaching 80% accuracy. This KNN leaves some tough competition for our MLP model to go up against. The untuned base MLP model performed poorly with a validation accuracy of 98.45% but a real test accuracy of only 34.71%. The tuned MLP only had its batch size and epoch count parameters tuned but was still able to show promising results. The model showed a validation accuracy of 99.78% and a real test accuracy of 79%, putting it right up there with the fully tuned KNN model.



Bar Chart comparing the real test accuracy scores of the best versions of each model that we trained.

Discussion

Machine learning is an excellent tool, it is useful to this project, and along the way, we have learned about various other applications of ML. Some such applications of ML also use python libraries which are used to “fit” different types of Supernovae events, as well as other astronomical phenomena. Active Galactic Nuclei (AGN) or Kilonovae are also examples of phenomena that can be studied using principles that are used to classify Supernovae as well. We have also come across challenges with adapting to the PyTorch package. Some aspects of the tutorial we followed were found to be unnecessary for our project but we did it anyway as “good programming” practice. One such part of the pipeline we found was redundant was the data loader function, we later realized this was particular to the PyTorch package which forces any NumPy array to be turned into a Tensor. This wasn’t the case for the Tensorflow model because the neural network in that model doesn’t require this extra step. Albeit that, we note that the performance of both models is comparable.

Redesign and Possible Future Improvements

The initial model for this project consisted of a basic Multi-Layer Perceptron (MLP). An MLP is a feedforward artificial neural network, which generates outputs based on the inputs and assigned weights if any. Furthermore, an MLP consists of several hidden layers, for the initial model we chose to use 3 hidden layers. Each layer consists of a certain number of "neurons" also known as hyperparameters. This basic first attempt saw us using a given method for turning the satellite data into a set of points forming a linear light curve. The number of neurons was calculated roughly to match the neurons for the MNIST PyTorch problem, as seen in the tutorial we were following which was published by Dr. Bownlee. We also normalized that data and padded the light curves to provide a uniform length (Brownlee).

In the future, this model can be improved by tuning on more parameters like the number of layers and nodes per layer to better calculate predictions for our final binary output. Furthermore, tuning some other parameters in this model such as the learning rate of the optimizer and trying some other more complex NN models such as more complex models offered by PyTorch might yield significant results. These other models will serve to better show the effectiveness of the final model that we do determine to be the most effective. Finally, collecting more real data to test with, by combining several databases, will greatly improve our final test set evaluation to better show how well-generated data can be used to train a model that can identify classes in real data.

Observations and Future plans to improve solutions

When we started modifying the data that we are training our model on, we made a few observations that we hope will help us improve our model in further iterations. We altered the original, generated data that was unbalanced to create a smaller, balanced dataset by undersampling the majority class. This data unexpectedly seemed to produce a less accurate model than the one trained on the unbalanced data. We also have worked to generate new data for the minority class through oversampling style methods, and it seems like it may be able to produce improved results for both balanced and unbalanced sets if used. While making these observations we have also been researching methods of using automated machine learning with models from the PyTorch library. We will be using the FLAML auto ml and tuning library to tune our current multilayer perceptron model.

Another important observation we made was that upon generating real data using TESS-CPM we found that there was a drastic difference between our trained model's input size and the test data's input size. In order to overcome this issue, Dr. Villar kindly provided us with a new simulated data set that was closer to the size of the test set as the model would not run without this important change. We realize that this process of generating a new training set might be something we need to do when new batches of data are received from TESS. This is due to the fact that TESS has a lot of random noise in the data that it collects and we need to check for that each time a new batch is processed.

What works and what does not?

Manually tuning can be time-consuming and we may not necessarily reach the optimum hyperparameter tuning values by hand-tuning. Instead auto ML seems to help in overcoming that challenge while also saving time. In the upcoming weeks, our hope is to implement this final step to achieve optimum hyperparameter values in order to obtain a decent neural network, as we also expect it to perform better than our current model which was not tuned much.

Future plans

In the future, we hope to implement AutoML and compare the model's results to the initial model we created which was manual or hand-tuned. Along with that we also need to check the size of the test set as mentioned earlier in the "Observations and future plans to improve solutions" section. Dr. Villar will use the findings in this research project to make a) produce a piece of publishing describing the inner workings and results of the neural network and b) she will be sharing the findings of this project with the astronomical community as it will produce new findings of transient events which will provide a new perspective to the study of this particular subject.

Lessons Learned

We learned a lot about how to properly implement the base mechanics of the PyTorch library and various specific parts related to Multi-Layer-Perceptrons. We have also learned how to use the FLAML auto machine learning with PyTorch neural net models in order to do hyperparameter tuning. Sam has also learned even more than prior about the importance of having expertise in the field which your data comes from as a data scientist. He has also learned a great deal about transients and the general formatting of astronomical data. Aish has learned more about the inner workings of a PyTorch-based MLP. She is also learning to navigate through the challenges of data wrangling and data integration which is essential to studies like these where there is an abundance of data but there is a general disconnect between different formats of data this is stored and collected by the different varieties of scientific instruments and telescopes.

Conclusion

In this work, we provide a fully functional machine-learning pipeline that is both: trained and tuned only on artificial transient data and is able to detect transient events in real data with promising accuracy. So far it has been able to detect 46 transient events from a data set containing 4,946 images from the TESS satellite and shows a 99.78% accuracy on validation data and 79% accuracy on real satellite data. This test data consists of the most recent data collected in August through September, by TESS. It is also of note that our baseline K-Nearest Neighbors model performed exceptionally with a validation accuracy of 99.82% and a real test accuracy of 80%. This shows that KNN, even if it is a much simpler model, might perform very

well for this problem. However, there is much room for improvement with tuning our MLP or even just using a more complex NN model whereas there is not much room to further improve the KNN model. A feedforward neural network such as the MLP we used in this project is indeed an effective way to detect and identify transients in nearby galaxies. With our results, we have also established a proof of concept. Especially with the identification of Supernovae SN2022ubb that was cross-referenced with the Zwicky Transient Facility (ZTF).

Contribution

Aishwarya Penmetcha's Contributions

I contributed to the project by working on developing the framework of the neural network along with Sam. I also worked on redesigning the initial MLP by adjusting the hyperparameters to account for the adjusted training set. Along with that I also worked on creating a program that will help in data integration which is crucial to creating one of the many test sets. Also worked on testing the model on a real data set and made the conclusion that for now 46 of the 4946 images had some indication of transient events in them. I also worked on editing the progress reports and participated in weekly meetings and one-on-one group meetings with Sam.

Samuel Geiser's Contributions

I contributed to the project by researching sources and tutorials for the initial development of a rough MLP model. I also worked to make the code for the initial model run correctly by modifying the tutorial code to better suit our data input. This included manual changes made to the layers and nodes of the model so that it could take in our transformed data and give a binary output. I also worked to tune the MLP model for this project as well as to train and tune several simpler models as baselines for the neural nets that we are focused on here. I also helped to produce the progress reports and participated in our weekly meetings on zoom with Aish and the meetings with our professors.

Github:
https://github.com/Aish-u/DS440-CAPSTONE

References

The Young Supernova Experiment – Time-Domain Astrophysics with the Pan-STARRS Telescopes, https://yse.ucsc.edu/. Accessed 21 October 2022.

Dalya, Gergely. "Glade." *GLADE+ galaxy catalog*, 10 July 2022, http://glade.elte.hu/index.html. Accessed 21 October 2022.

Brownlee, Jason. "PyTorch Tutorial: How to Develop Deep Learning Models with Python." *Machine Learning Mastery*, 2020, https://machinelearningmastery.com/pytorch-tutorial-develop-deep-learning-models/. Accessed 5 October 2022.

Hattori, Soichiro, et al. "The Unpopular Package: A Data-Driven Approach to Detrending TESS Full-Frame Image Light Curves." The Astronomical Journal, vol. 163, no. 6, June 2022, p. 284, https://doi.org/10.3847/1538-3881/ac625a.

PyTorch Documentation — PyTorch 1.12 Documentation. https://pytorch.org/docs/stable/index.html. Accessed 23 Sept. 2022.

soichiro-hattori. Soichiro-Hattori/Unpopular. 2019. 2022. GitHub, https://github.com/soichiro-hattori/unpopular.

Graff, Philip. "{SkyNet}: an efficient and robust neural network training tool for machine learning in astronomy." *Monthly Notices of the Royal Astronomical Society*, vol. 441, no. 2, 2014, 1741--1759. *Oxford Academy*, https://academic.oup.com/mnras/article/441/2/1741/1071156.

Tu, Zuo-Lin, et al. "Convolutional Neural Networks for Searching Superflares from Pixel-level Data of the Transiting Exoplanet Survey Satellite." *The Astrophysical Journal*, vol. 935,

no. https://dx.doi.org/10.3847/1538-4357/ac7f2c, 2022, p. 90. *arxiv*,

https://dx.doi.org/10.3847/1538-4357/ac7f2c.