Utilizing Machine Learning to Search for LIGO Sources: Interim Report 2

Sarah McCarthy[1]

August 6, 2018

[1]Grinnell College,
1115 8th Ave. Grinnell, IA 50112

1. Introduction

In 1916, Albert Einstein predicted the existence of gravitational waves, ripples created by accelerating masses that would propagate through spacetime [1,2]. However, he believed they would be too small for human detection. On September 14, 2015 at 09:50:45 UTC, nearly one century after Einstein's prediction, both detectors of the Laser Interferometric Gravitational-wave Observatory (LIGO) simultaneously observed a transient gravitational-wave signal [3]. This was the first direct detection of gravitational waves, as well as the first detection of a binary black hole merger. The two LIGO observatories are in Livingston, LA and Hanford, WA. Each instrument is a dual-recycled Michelson interferometer with 4 km arms [4]. LIGO's discoveries were made possible by a factor of 10 sensitivity improvement in the frequency regime around 100 Hz.

One of the challenges for LIGO is differentiating between signal and noise. Transient noise arises as short $O(seconds)$ glitches in the data that can mimic true transient astrophysical gravitational wave signals including binary black hole mergers. One powerful method to identify signals with waveforms that are well predicted by Einstein's general theory of relativity, including neutron star and black hole binaries, is matched filtering, which calculates the cross-correlation between modeled templates and the noisy gravitational wave detector data. The PyCBC pipeline, which identified several LIGO discoveries, employs matched filtering to calculate the signal-to-noise ratio (SNR) between all modeled templates considered in a gravitational wave search. To help make the pipeline more robust to noise, PyCBC uses a χ^2 test to downweight the SNR of events where the data does not match the modeled template well. Any times where the re-weighted SNR is above threshold are saved as "triggers" [5]. Any given trigger will have multiple associated modeled templates with non-zero re-weighted SNR, each with individual re-weighted SNR tends to be densely clustered in total mass and effective spin.

One of the outputs of this process is a set of parameters, such as total mass, effective spin, and maximum re-weighted SNR [6]. These parameters

can be plotted for a given stretch of data. For gravitational wave signals, the plots typically have a compact area of high maximum re-weighted SNR, with a relatively low re-weighted SNR over other values of total mass and effective spin. If the event is noise, typically maximum re-weighted SNR is not well localized in these template parameters.

Our approach to improving the PyCBC pipeline performance by limiting the impact of transient noise is to use a convolutional neural network and image classification. A neural network is a biology-inspired computer program in which a computer learns a specified task from a series of provided 'training' examples. Neural networks have successfully been used to classify images of LIGO data in the past [7]. We will build an image classifier that takes as input a plot representing the distribution of templates associated with a trigger time in total mass and effective spin. We expect that the much more well-localized appearance of true signals in this parameter space will serve as a powerful distinguishing feature for our machine learning image classifier.

2. Objectives

The aim of my summer research project is twofold:
1) Create a convolutional neural network that will differentiate signal and noise in LIGO data
2) Test this algorithm's performance on increasingly large data sets

3. Approach

I will first design and build a simple convolutional neural network (CNN) algorithm that can intake images of the total mass and effective spin distribution of PyCBC triggers and output some likelihood that the trigger belongs to the 'signal' class or the 'noise' class. Next I will need to develop a training set to train the CNN to make accurate classifications. I will inject a series of simulated gravitational wave signals into data from Advanced LIGO's second observing run and flag each of these as part of the 'signal' class. I will use background events from the search itself as noise so that we focus on noise that is challenging for a CBC search to analyze.

After I have trained the CNN with known examples of both the signal and glitch classes, I will test it with a test data set. To ensure this data is independent from the training set I will use a different period of Advanced LIGO data, and I will follow the same method as above to inject signal examples and identify glitch examples. I will evaluate the performance of the CNN by producing a confusion matrix for the test data which will calculate the fraction of mis-classified PyCBC triggers for each class. I anticipate tuning the CNN based on these results.

At this point, I will be ready to expand my training set and data set to an extended set of O2 data. Then, I will produce and analyze my results. All simulations will be run from my personal laptop using Caltech's LDAS computing cluster for computational power.

4. Project Schedule

I will follow the timeline below:

Weeks 1-2: Learn LIGO software and computing clusters, assemble basic code

Weeks 3-5: Tune algorithm using isolated test cases

Weeks 6-8: Run on extended data set

Weeks 9-10: Combine results, prepare final report and presentation

5. Current Work

Upon my initial arrival at Caltech, I began to familiarize myself with LIGO data through Gravity Spy. I practiced categorizing different types of glitches so that I could develop intuition categorizing noise and distinguishing it from signals. This work was important for the potential to need to troubleshoot further in the project when building the training and test data sets. I then designed the data feature that I intend to feed into the machine learning algorithm, studying how the SNR varied with certain parameters like end time, mass, and spin, to name a few. I more strictly defined the information to feed into the classifier so that I can reliably represent the data that I am putting into the classifier, and I modified the existing software to automate plots for an inputted time.

Once I familiarized myself with promising feature sets for a signal, I then tested the plots on noise. I used Python to find the times that had maximum SNR and were not artificial. I reran the same density plots, this time using noisy data, and searched for more interesting behavior. I decided that the plots of reduced χ^2 v. template duration and end time v. template duration exhibited the most distinct behavior. Once I had done a preliminary analysis of the plots, I modified the code so that it would display the data with more confined axis limits and only analyze that portion of the data. Part of this required that I modify the binning of the data so that it would best display meaningful information. Examples of the impact of binning can be seen below in Figure 1.

To prepare for feeding the images into a CNN, I automated the generation of plots to the top 100 SNR events for data sets from Livingston and Hanford. I also ran a similar test on injected signals that were superimposed over Gaussian noise. Expanding to larger and more diverse data sets allowed me to search for patterns with more precision and to finetune my ability to search for deviance from the more clearly defined signal patterns. For reduced χ^2 plots, the highest new SNR points are concentrated around lower template duration and are narrowly peaked in reduced χ^2. At higher reduced χ^2 values, new SNR points fan out, allowing for more values of template duration at higher reduced χ^2 values. However, at those points, the new SNR value is lower, indicating that the fit is worse. For the end time plots, a similar phenomenon occurs where the high SNR points are concentrated in a small area and then spread out, but not as uniformly as for reduced χ^2 and not in the same fan pattern; the end time spread resembles more of a triangular shape.

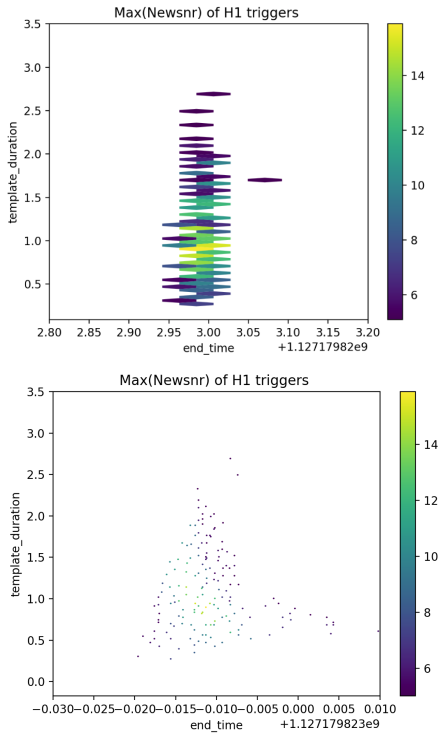

Figure 1: The top figure is an example of bad binning, where the bins are too wide to effectively distinguish the shape of the distribution. On the bottom, the binning has been adjusted so that a more distinct feature space can be seen.

After I made the data understandable for human analysis, I needed to modify it further so it could be read by a computer. One aspect of this was altering the plotting so that the absence of data would not be interpreted as more significant than it was. If I were to keep the code in its original format as dictated, the white background would be interpreted by a computer as an extrema rather than the absence of information. This posed a significant problem when there were small holes in the middle of large data clusters. To counter this issue, I decided to use an inverse gray scale for the color bar so that I could place the white end of the spectrum at the lower limit of the color bar. This would effectively increase the noise floor to the lower limit. To force the code to always produce the same output, I also eliminated the feature of the code that allowed for a logarithmically scaled color bar. This then produced identical scaling for all plots.

Another modification that needed to be made was cropping the plot so that the neural network would receive data that was all significant and understandable. This entailed setting a consistent values for dots per inch (DPI) and image size. I set the figure size to be 6in x 6in, with 200 DPI. With those two values

set constant, I could then calculate the pixels that would specify the region I should crop so that the neural network would only analyze the plot, eliminating features like the axes and color bar.

Once I had specified the plot features, I saw interesting behavior in the outputted injection plots. There was a strong band of high SNR data points that seemed as if they were saturating the upper limit of the color bar, implying that they were an artifact of an artificial signal. To eliminate that problem, I modified the code so that after masking the injections to the 100 loudest SNR, any peaks that had either above SNR or template duration under 4 seconds 18 were removed. This also eliminated the artificially high SNR band from the template duration vs. end time dataset.

Because the template duration vs. reduced χ^2 plot exhibited differing behaviors from when viewed on different scales, I decided to consider three axis limits for reduced χ^2 - 8, 30, 140. To allow the neural network to consider how a true signal would behave in all three limits, I wrote additional code to combine all three views into one image.

With all of the aforementioned changes, I had effective code that could produce plots that would be easily understandable by a CNN. However, I needed to generalize the code so it was easily iterable. After doing that, I was then able to test my neural network on a small data sample from the two data sets and the injection data. The CNN ran and produced reasonable results, able to predict the outcome of the data to an average of 99-98 percent accuracy.

Then, I was able to change the neural network so that I could input only one dataset and set aside a random 90 percent of it for a training set and a random 10 percent for a testing set. However, with the current parameters, the CNN was running very slowly, so I altered the CNN to read in the data with a smaller pixel count. After that modification, the CNN seemed to be producing eerily consistent results, so I checked the plots with reduced pixel count. I found out that the method I was using to downsample was taking the maximum value, which was solely based on the gray scale count. Although white was the minimum value on my color bar, it represented the maximum value in the grayscale. To counteract that, I changed the downsampling to the mean value and that method of averaging produced a plot similar to the original but at a lower resolution.

In order to ensure that the data I was analyzing in the injection data set, I wanted to compare the highest SNR times with the times at which a simulated gravitational wave signal was injected. I included a two second time window from the intended injection time as another filtering parameter. With that extra level of selection, I ran the injections again to include the top 1000 SNR and got 15 outputted plots that met all criteria. Similarly, I ran my Livingston and Hanford data sets over the top 1000 SNR to include more samples. I have produced multiple injections of signals over real noise, and I am currently injecting them into the CNN.

6. Challenges

One of the difficulties that arose thus far was determining what made the

plots interesting enough to consider them as viable for training the CNN. I struggled to develop an understanding of what combinations of data would provide a meaningful plot, especially since a large portion of that entailed learning how to quickly yet effectively interpret code written by someone else. A similar problem that arose from that issue was doing unnecessary work. It took me a long time to find out that code that I had been writing was already part of the code I was using, just hidden under different names or deeper in source code.

Another difficulty was determining how to constrain the plots that would allow high levels of clarity for data sets with differing distributions. This required studying the behavior that signals had and finding variations across different noise data sets to look for different patterns. Changing the binning was a significant step towards remedying this issue, and utilizing the potential to run the CNN over different data constraints will help in the future.

A current difficulty for me is the uncertainty if the CNN will still be as effective on O2 data sets as they have been on O1 data sets, because I have only tested O1 data. This will hopefully be remedied in the next week, as I move into analysis of the CNN on O2 data.

Acknowledgements

I would like to thank Dr. TJ Massinger and Dr. Jess McIver for their guidance and support throughout the summer.

References

1. A. Einstein, Sitzungsber. K. Preuss. Akad. Wiss. 1, 688 (1916).
2. A. Einstein, Sitzungsber. K. Preuss. Akad. Wiss. 1, 154 (1918).
3. B.P. Abbott et al. (LIGO Scientific Collaboration and Virgo Collaboration). "Observation of Gravitational Waves from a Binary Black Hole Merger." *Phys. Rev. Lett. 116. 061102* (2016)
4. LIGO Scientific Collaboration. "Advanced Ligo." *arXiv:1411.4547* (2014)
5. B.P. Abbott et al. (LIGO Scientific Collaboration and Virgo Collaboration). "Characterization of transient noise in Advanced LIGO relevant to gravitational wave signal GW150914." *arXiv:1602.03844* (2016)
6. B P Abbott et al. (LIGO Scientific Collaboration and Virgo Collaboration). "Effects of data quality vetoes on a search for compact binary coalescences in Advanced LIGO's first observing run." *Class. Quantum Grav. 35 065010* (2018)
7. M Zevin et al. "Gravity Spy: integrating advanced LIGO detector characterization, machine learning, and citizen science." *Classical and Quantum Gravity, Volume 34, Number 6.*(2017)