# Utilizing Machine Learning to Improve Searches for LIGO Sources

Sarah McCarthy[1]
Mentors: Dr. Thomas Massinger and Dr. Jessica McIver[2]

September 20, 2018

[1]Grinnell College,
1115 8th Ave. Grinnell, IA 50112
[2] California Institute of Technology,
1200 E. California Blvd. Pasadena, CA 91125

## 1. Abstract

LIGO is one of the most sensitive instruments ever built, and with great sensitivity comes a significant amount of noise. Currently, the LIGO Scientific Collaboration is searching for ways to more efficiently differentiate between signal and noise, on both hardware and software. To date, the PyCBC pipeline has aided in the analysis of several gravitational wave detections. We utilized outputs of PyCBC software to compile larger amounts of data, including signal and noise, into SNR density plots, and we modified them so that they could be easily interpreted by an image classifier. We expected the signal density plots to be much more well-localized than for noise, based on observations of known instances of signal and noise. After selecting the parameters that demonstrated features in the density plots, we created a convolutional neural network to search for these patterns. We trained and tested the neural network over increasingly large and varied data sets.

## 2. 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 $\chi^2$ 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 potential events, each with a set of parameters, such as total mass, effective spin, template duration, end time, 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 uses 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 built 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. In this parameter space, true signals have a much more well-localized appearance, which serves as a powerful distinguishing feature for our machine learning image classifier.

## 3. Methods

The aim of my summer research project was 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

First, I designed and built a simple convolutional neural network (CNN) algorithm that can intake images of parameter distribution of PyCBC triggers and output some likelihood that the trigger belongs to the 'signal' class or the 'noise' class. Next, I developed a training set to train the CNN to make accurate classifications. I injected 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 used 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 trained the CNN with known examples of both the signal and glitch classes, I tested it with a test data set. To ensure this data was independent from the training set I used a different period of Advanced LIGO data, and I followed the same method as above to inject signal examples and identify glitch examples. I evaluated the performance of the CNN by producing a confusion matrix for the test data which calculates the fraction of mis-classified PyCBC triggers for each class. I then tuned the CNN based on these results. Then, I was able to expand my training set and data set to an extended set of O2 data and then to include all images I had generated - real signals over real noise, injected signals over Gaussian noise, and injected signals over real noise - and produce and analyze my results. All simulations were run from my personal laptop using Caltech's LDAS computing cluster for computational power.

I followed 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

## 4. 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 $\chi^2$ v. template duration and end time v. template duration exhibited the most distinct behavior. The juxtaposition between poorly selected parameters and those I selected can be seen in Figures 1 and 2 below.

After 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 the data included in the plot. To meaningfully display the information, I modified

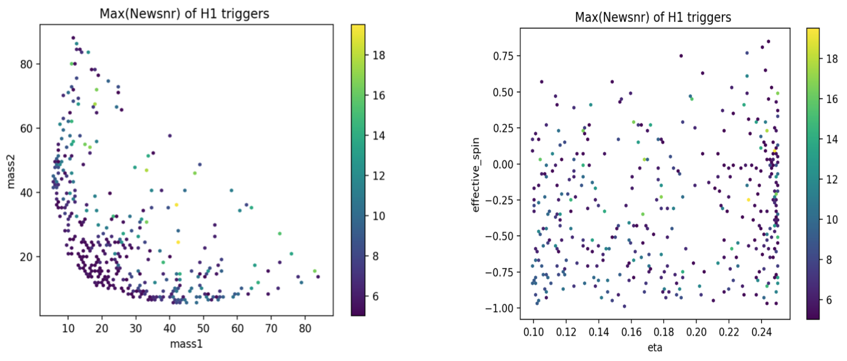


Figure 1: Samples of noise SNR density plots that are not well-localized.

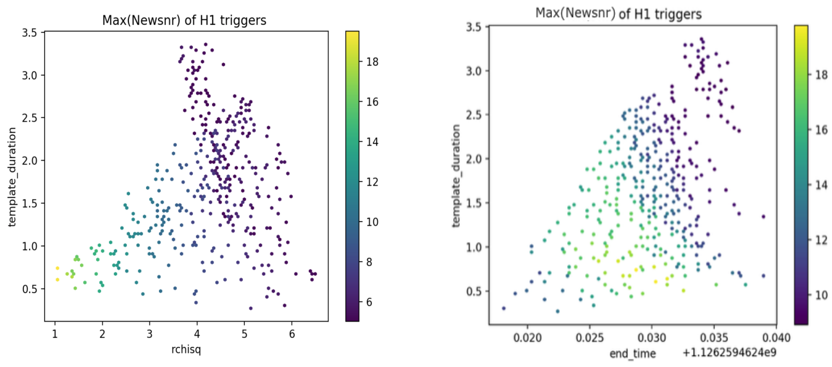


Figure 2: Samples of signal SNR density plots that are well-localized. These two sets of parameters were selected for use in the CNN.

the binning until the display was optimized. Examples of the impact of binning can be seen below in Figure 3.

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 $\chi^2$ plots, the highest new SNR points are concentrated around lower template duration and are narrowly peaked in reduced $\chi^2$. At higher reduced $\chi^2$ values, new SNR points fan out, allowing for more values of template duration at higher reduced $\chi^2$ 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

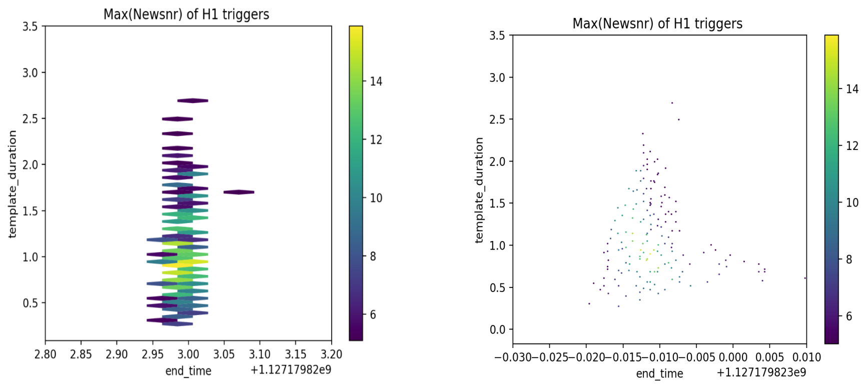


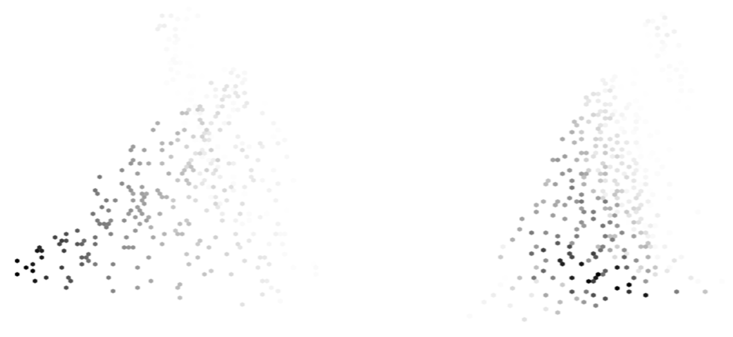
Figure 3: The left figure is an example of bad binning where the bins are too wide to effectively distinguish the shape of the distribution. On the right, the opposite is shown, where the bins are too small to communicate the distribution of the data, potentially allowing the CNN to over-fit.

as uniformly as for reduced $\chi^2$ and not in the same fan pattern; the end time spread resembles more of a triangular shape.

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. The final result was plots that looked like those in Figure 4.

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



(a) Final image for reduced $\chi^2$ (b) Fimal image for end time

Figure 4: Samples of modified images that are prepared for input into the CNN.

modified the code so that after masking the injections to the 100 loudest SNR, any peaks that had either above SNR 18 or template duration under 4 seconds were removed. This also eliminated the artificially high SNR band from the template duration vs. end time dataset.

Because the template duration vs. reduced $\chi^2$ plot exhibited differing behaviors from when viewed on different scales, I decided to consider three axis limits for reduced $\chi^2$ - 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, as can be seen in Figure X.

Figure 5: Combined view of reduced $\chi^2$ plots to consider multiple viewpoints. The three x-axis limits are 8, 30, and 140.

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 then produced plots that had injected signal over real noise recorded at Livingston and Hanford. This would provide a more realistic set of parameters for the CNN to recover. Now, I had a more comprehensive dataset to analyze.

In order for the CNN to analyze the data, I needed to prep it by providing matching arrays, since I was using a supervised CNN. This required going through each individual dataset and finding the times for which a signal was either recorded or injected. Then, I could combine all of these individual datasets and matching arrays into a large array with images and either a [0,1], noise, or [1,0], signal. One especially important component of the analysis was to generate a confusion matrix that would output how frequently the CNN miscategorizes signal as noise or noise as signal, as well as how frequently the CNN correctly categorizes signal and noise.

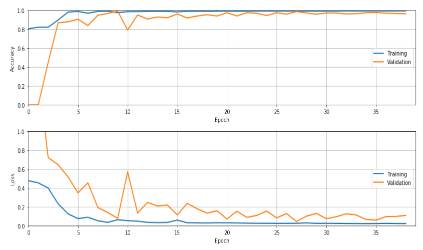
As one final modification, I altered hyperparatmeters, such as the activation function, optimizer, epochs, batch size, and learning rate until I had a combination that produced the highest accuracy.

## 5. Results

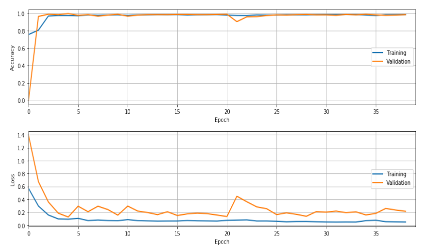
After generating a large enough sample set, I was able to run my CNNs over the randomly selected training and test sets. Sometimes, the accuracy would start around 60 percent and flatline, never showing any signs of true learning or improvement. We believe that the potential for this error arises from an inadequate training set, where there are not enough high SNR examples for the CNN to learn the distinct features of a signal. Therefore, the CNN loses its discriminating power and cannot learn effectively. However, most of the time, the results matched those demonstrated in Figure 6. The final accuracy was around 97 percent, on average.

## 6. Future Work

We believe that these results are promising of the CNN's potential to effectively distinguish signal and noise. There are modifications to the code that need to be implemented to improve the CNNs' accuracies. We plan to use more injection sets with higher SNR as training sets to eliminate the potential for the CNN to never learn. We also could generalize this code to test every combination of parameters to see if there are other parameters that would be as effective, if not more effective, when used in a CNN. We also could undo the grayscale to see if the CNN has any bias towards reading the absence of data as minima rather than maxima. In theory, the CNN should not be impacted by that modification, but it would be worthwhile to ensure that it is not. We also plan to increase the SNR floor to optimize the network performance and analyze how accuracy changes with the SNR cutoff.



(a) Accuracy and loss plots for reduced $\chi^2$ CNN



(b) Accuracy and loss plots for end time CNN

Figure 6: These are randomly selected accuracy and loss plots for reduced $\chi^2$ and end time CNNs. Both CNNs operate around a final accuracy of 97 percent, with the majority of errors stemming form a selection bias in the training set.

## 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)

The authors are grateful for the support of the US National Science Foundation's Research Experience for Undergraduates Program, award #1757303.