Sub-Classification of Blip Glitches Using Q-Transforms and Convolutional Neural Networks

Melissa Kohl

August 2, 2018

Abstract

In the Advanced LIGO observation runs, detection of gravitational waves is directly dependent on the sensitivity of the detectors. The strain data from both detectors contain continuous and transient noise that restricts the sensitivity of each detector. Transient noise, called "glitches," not only affects the general sensitivity of the detectors, but also mimics and obscures real gravitational waves in the calibrated strain data channel. The machine learning software package used to classify these glitches and identify their sources, GravitySpy[1], is successful when the spectrogram of the glitch has a very distinct and unique shape. However, the spectrogram of one of the most common types of glitches, called a "blip," has an underwhelming shape with no distinct characteristics. Additionally, auxiliary channels other than the calibrated strain can pick up glitches and supplement GravitySpy's search for sources, but blip glitches are very infrequently witnessed by other channels, making it difficult for GravitySpy to identify a source (or possibly multiple sources). The focus of this paper is to examine blip glitches in a format other than a spectrogram, such as a Q-transform, to determine if there are sub-classifications of blips that might have identifiable sources, and then use Convolutional Neural Networks to sub-classify these blips. Manual searches of Q-transforms of a variety of blip glitches, nearly indistinguishable in a spectrogram, reveal six distinct possible subclasses. Additionally, the implementation of simple neural networks has provided preliminary evidence of fundamental differences between these hypothesized subclasses, confirming the assumption that there are multiple types of blips.

1 Introduction to Glitch Classification

The glitches in the strain data from the science and observation runs of the Advanced LIGO detectors decrease the sensitivity of the detectors and obscure gravitational waves [1]. Although some glitches can be identified and eliminated from the strain data during later analysis, astronomical events that give off additional radiation, such as neutron star mergers, need to be recognized immediately so that astronomers can observe the event. Additionally, the shapes of some glitches, such as "blip" glitches, mimic that of a gravitational wave from a binary black hole merger so well that one of the only ways to distinguish a blip from a gravitational wave is by comparing the data between detectors. As a result, it is imperative to find the sources of the glitches and eliminate them before future observing runs, directly increasing the sensitivity and effectiveness of the detectors while observing [2].

The current method for classifying glitches and identifying their sources is a machine learning software package called GravitySpy [1]. Unlike previous machine learning techniques used on LIGO data that only looked at the waveforms of the glitches [2], GravitySpy's neural network takes in spectrograms from four different time frames to create a multi-layer network that utilizes image classification techniques [3]. Since different types of glitches have different durations, the multiple views not only provide complementary data across time frames, but also allow for identification of a broader group of glitches of different durations [3]. GravitySpy also relies on citizen science to classify glitches, utilizing a large volunteer network to aid in adding glitches to the neural network training set[2]. As a result, GravitySpy is great at classifying glitches into known classifications [1], but the classifications themselves may be too broad. The output function in GravitySpy's neural network is *softmax* [3], which essentially just classifies the input glitch into the classification with

[1] GravitySpy documentation
[2] GravitySpy Zooniverse

the highest correlation, regardless of how high that correlation is. The combination of the multiple-view input and the *softmax* output function allow for glitches caused by different sources to be classified into the same group if the shapes of the glitches are similar. In section 3.1, I introduce an example of two different spectrogram shapes (which most likely have different sources) that are grouped into the same classification.

Once glitches are classified, GravitySpy is also used to find similar glitches in the hundreds of thousands of auxiliary channels keeping track of the instruments and environments of each detector in an attempt to locate the source of each glitch classification [1]. Currently, this method is insufficient for finding the source of blip glitches.

2 Initial Approach and Goals

The most frequently-occurring glitches, including blip glitches, are likely conglomerates of glitches from different sources that happen to create the same general shape in a spectrogram. Unfortunately, the shape of a blip glitch in a spectrogram is uninspiring, with no weird spikes or unique shapes to provide a hint towards its source, and little to even distinguish one from another. As mentioned in the previous, blip glitches are also troublesome due to their resemblance to binary black holes. To sub-classify these blip glitches for future elimination, we first need a different analysis technique that might reveal hidden characteristics.

One common way to visualize a glitch or other transient noise waveform in the strain data is a Q-transform, which is a time-to-frequency domain transform related to the Fourier transform. The quality factor Q is related to the number of cycles processed at a central frequency, and in comparison with a Fourier transform, the Q-transform has a better frequency resolution over a logarithmic frequency scale, and a better resolution for short duration signals. Since the glitches and gravitational waves in the Advanced LIGO strain data have a very short duration and span auditory frequencies, which are on a logarithmic scale, the Q-transform is a more desirable time-to-frequency domain transform for our purposes than the Fourier transform. Additionally, while a Q-transform still produces a spectrogram, similar to the spectrogram images used by GravitySpy, the parameters set by GravitySpy are designed to help visualize the blip over larger time frames, and my goal is to use the Q-transform to zoom in on the blip glitches to see possible characteristics over a small time frame.

Once we find distinguishable subclasses of blip glitches in Q-transform images, we can use those images as input to different neural networks to find a neural network that can sort the blips into sub-classification. If we can identify clusters and groupings of using different neural networks, there is a possibility of identifying the specific characteristics that define the subclasses. Without sub-classification, GravitySpy cannot identify the sources of blip glitches, but with a new input set, the possibility of finding a source and reducing some of the noise caused by blip glitches is largely increased.

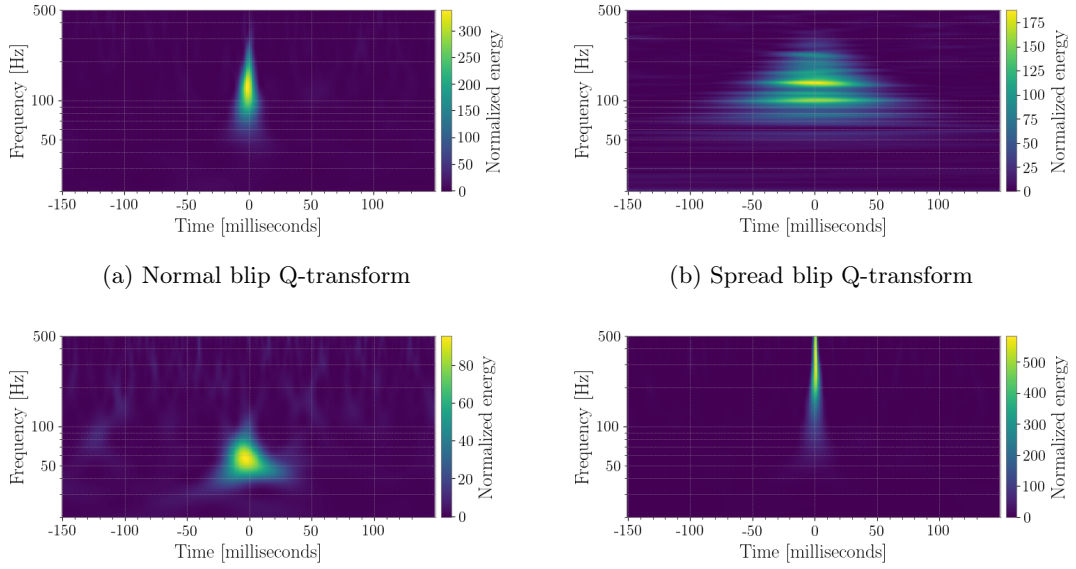
3 Investigation of Blip Glitches

3.1 Initial Look at O1 Blips from Livingston

To gain a better understanding of blip glitches and their characteristics, I started by performing simple Q-transforms on known blip glitches from the first observing run at the Livingston detector. I wrote a Python script, adapted from the GWpy[3] Q-transform documentation[4], to plot the Q-transform of about 30 random blip glitches, all cropped from the surrounding 30 seconds of strain data to a final image with a time domain of 0.30 seconds. While manually looking through the images, I found four distinct types of blip glitches, as seen in figure 1 on page 3. I call these four types normal, spread, dot, and stick for their appearances in the Q-transform. Since the shapes of these images are drastically different from each other, I made the assumption that at least one of these could be a subclass of blips.

After discovering these differing forms in the Q-transforms, and plotting about 100 more Q-transforms, I looked at the existing spectrograms from LigoDV-Web of a specific glitch from each

[3] GWpy overview
[4] Q-Transform code example



(a) Normal blip Q-transform

(b) Spread blip Q-transform

(c) Dot blip Q-transform

(d) Stick blip Q-transform

Figure 1: Four different types of blip glitches from Livingston O1 in Q-transforms, all with the same time duration of 0.30 seconds, cropped from 30 seconds of transformed strain data. The Q-value calculated by GWpy is the same for all four (5.65). Note that the energy scale is different for each blip.

of the four possible subclassifications to determine the legitimacy of my assumptions, the results of which are in figure 2 on page 4.

The most striking difference between the spectrograms and the Q-transforms is from the spread blip. The spectrograms from the normal blip and the spread blip, figures 2a and 2c, appear to be almost exactly the same, but while the normal blip's Q-transform (figure 2b) has the same shape as its spectrogram, hence the name "normal," the spread blip's Q-transform (figure 2c) "spreads" outward with distinct horizontal lines. These horizontal lines (which are essentially frequency bins) suggest that the duration of a spread blip is longer than the rest of the blips and therefore it needs more data than 30 seconds for a clear Q-transform.

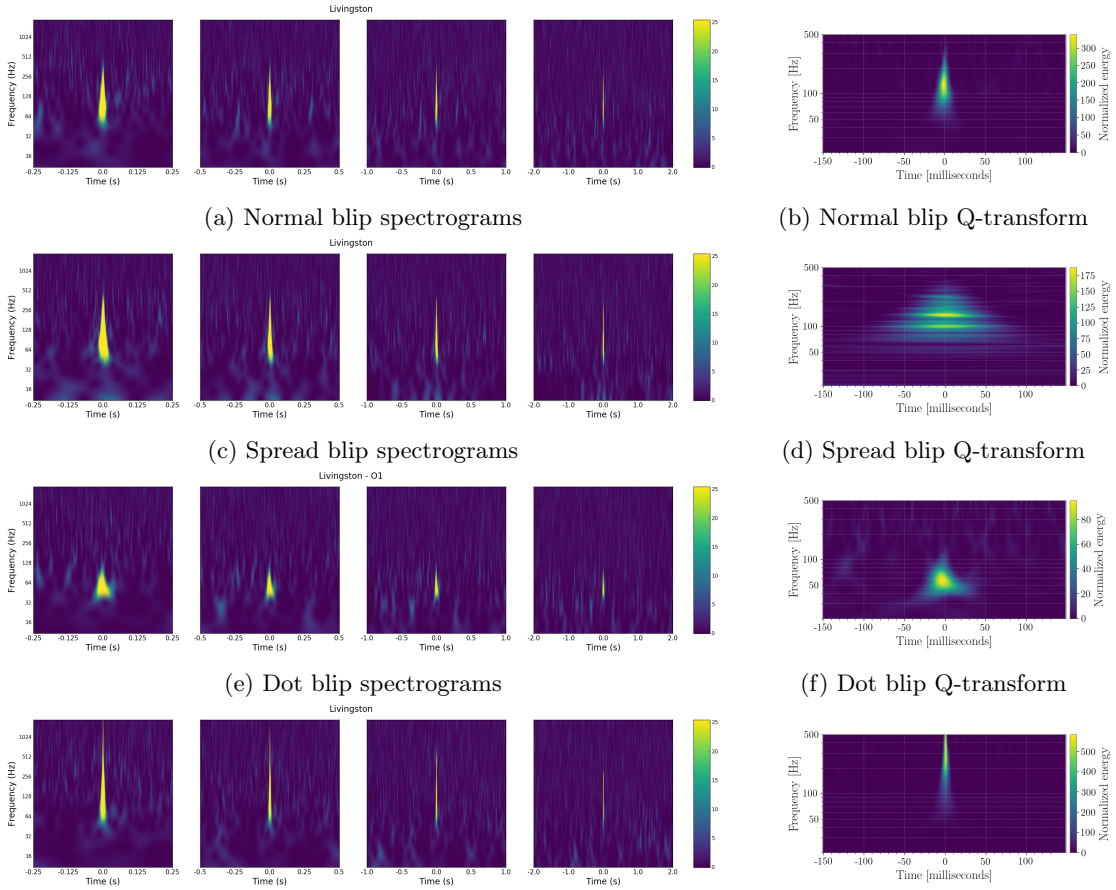
The dot blip also has a chance of being a legitimate sub-classification, as its Q-transform (figure 2f) has a distinct round shape at a relatively low frequency compared to the other blips. Unlike the spread blip, however, the spectrogram of the dot blip (figure 2e) is easily distinguishable from the others, with a much smaller frequency range and an almost triangle-like shape. This is an example of a possible distinguishable classification of glitch that GravitySpy groups in with blips, as mentioned in the section 1. So, since a dot blip can be identified on a spectrogram, it is possible that it could be classified apart from other blip glitches by GravitySpy, but it is worth investigating possible attributes of dot blips, such as lower frequency and longer duration compared to other blips.

The stick blip also seems to be different than a normal blip. Its spectrogram (figure 2g) is only slightly skinnier than that of a normal blip, with a slightly longer tail. However, the Q-transform (figure 2h) reveals that a stick blip is at a significantly higher frequency than a normal blip. Even if the stick blip is the same shape as a normal blip, the higher frequency sets it apart, especially compared to dot blips.

In addition, this comparison validates GravitySpy's ability to classify similar images. The four images found from Q-transforms do not appear to be mis-classified, they simply have characteristics that cannot be discerned from GravitySpy's specific type of spectrogram.

3.2 Further Investigation into O2 and Hanford Blips

Before delving into possible characteristics that could separate the types of blips I had found, I wanted to clear up the spread blip's Q-transform and see whether the same variety of shapes from my findings in O1 at Livingston are present in O2 and/or at Hanford.



(a) Normal blip spectrograms

(b) Normal blip Q-transform

(c) Spread blip spectrograms

(d) Spread blip Q-transform

(e) Dot blip spectrograms

(f) Dot blip Q-transform

(g) Stick blip spectrograms

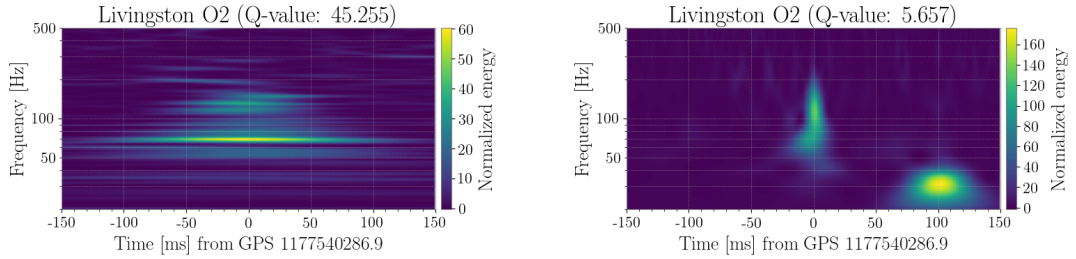
(h) Stick blip Q-transform

Figure 2: A comparison between spectrograms of four different blip glitches on the left and the corresponding Q-transform of the same blip glitches on the right. The spectrogram images come from LigoDV-Web, with time durations from left to right of 0.5 seconds, 1.0 seconds, 2.0 seconds, and 4.0 seconds. The Q-transform images were created from my Python script, cropped from 30 seconds of transformed data down to 0.30 seconds. The top row is a normal blip, the second row is a spread blip, the third a dot blip, and a stick blip on the bottom. Although the spectrograms of the different blips are distinguishable from one another, the Q-transforms reveal that these blip glitches may have fundamental differences.

I attempted to condense the spread blips by creating another two Q-transforms for each spread blip using the surrounding 40 seconds and 60 seconds of data, rather than my original 30 seconds, thinking that the image wasn't clear due to the signal having a longer duration compared to other blips. However, the exact same spread effect occurred for all of the known spread blips for both widened time domains. So, I went the other direction and used the surrounding 20 seconds, which could indicate a problem with the internal whitening that occurs to create the spectrogram of the transform, rather than a signal duration problem. Fortunately, this shortening of the domain cleared the images and we could see the actual shapes of the spread blips, one example of which is in figure 3 on page 5.

If not for the suspiciously large Q-value in figure 3a, it would appear that the original time domain parameter of the Q-transform was simply obscuring the image. Some of the spreads condensed down to dot blips, others normals and sticks. A couple spreads even contained two apparent signals within 100 milliseconds of each other, such as figure 3b, which I call a double-blip. Although changing the amount of data to get rid of spreading in my Q-transforms is good enough for the purposes of this project, the reason behind this spreading is unknown and could be a future point of investigation.

Now knowing that 20 seconds is a more consistent parameter for the Q-transform than 30 seconds, and also knowing that there was a large possibility for other types of blips beyond dots, normals, and sticks, I plotted and classified around 150 blip glitches from Hanford O2 and about

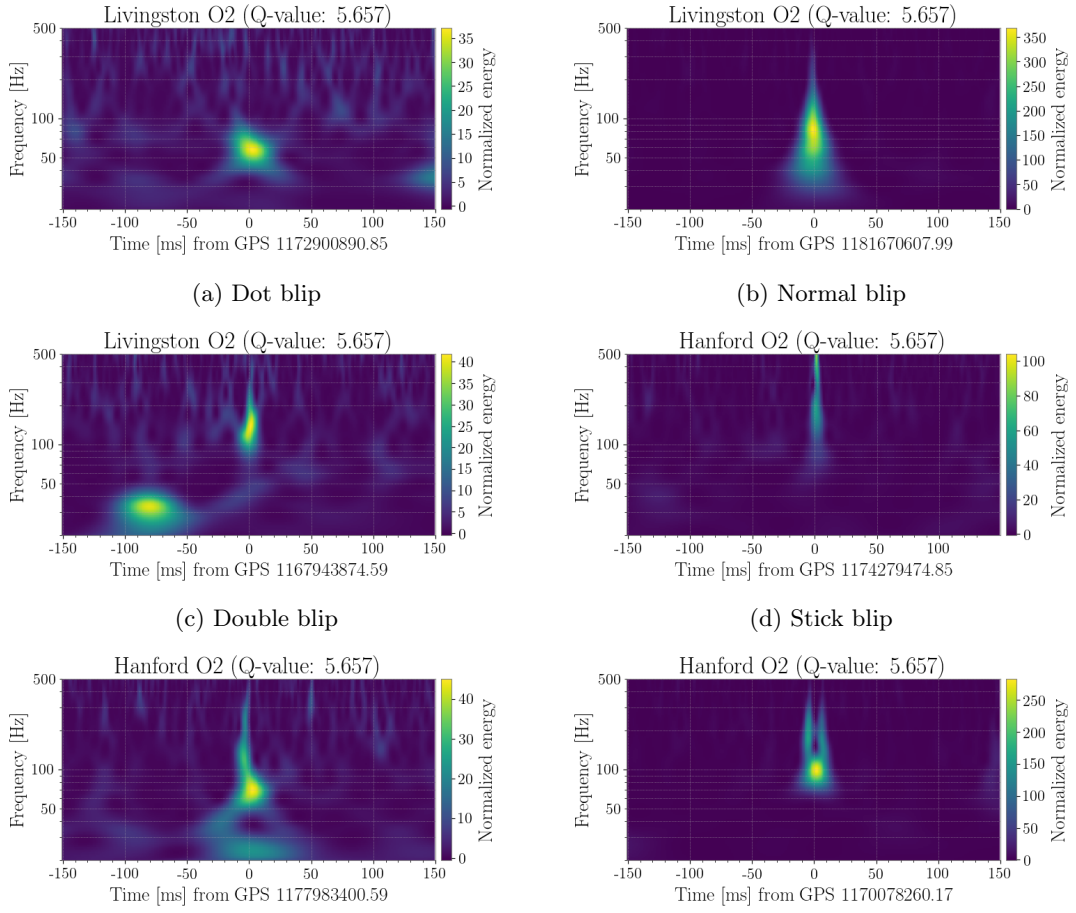


(a) Spread blip with 30 second domain

(b) Spread blip with 20 second domain

Figure 3: A blip glitch from Livingston O2 with two different amounts of surrounding data, both cropped back down to 0.3 seconds.

100 blip glitches from Livingston O2. The six major shapes of blips, all of which are at both detectors, are in figure 4 on page 5. The dot, stick, and normal are still there, but I also found double-blips, snitches (named for the Golden Snitch from *Harry Potter*), and hats (named for the Sorting Hat from *Harry Potter*).



(a) Dot blip

(b) Normal blip

(c) Double blip

(d) Stick blip

(e) Hat blip

(f) Snitch blip

Figure 4: Six possible subclasses of blips.

The variance in the background of each image is due to the relative energy of each blip. In this figure, the dot blip (figure 4a) has a relatively low energy compared to the normal blip (figure 4b), but I also found high energy dots and low energy normals in my image database. The double-blip, on the other hand, appears to always have relatively low energy, and it always has one low-frequency dot shape around 100 ms away from the mid-frequency, lower duration signal that causes the omicron trigger. It is unclear whether this is simply a coincidence, where two blips

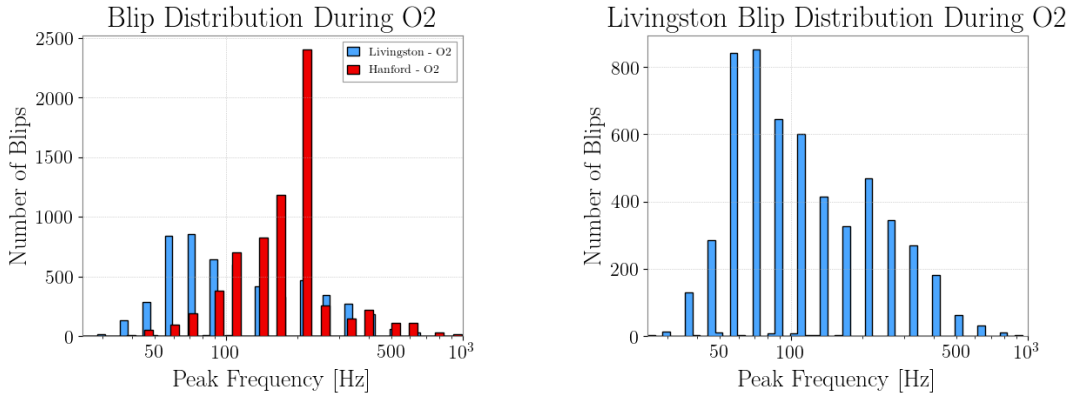
just so happen to be practically on top of each other, or if these are two connected signals in a cause-effect relationship, or whether the low-frequency dot is simply a side effect of the whitening of the Q-transform.

The hat blip, similar to the double-blip, is a complete mystery. The name comes from the wavy, squiggly nature of its shape, which is very different from the symmetry present in all the other types of blips (other than double-blips). The hat in figure 4e is particularly interesting and unique due to the hole underneath the brightest part of the signal, but all hats follow the general shape of a wavy top and a more spread-out, wavy bottom. One consistency among hat blips is their general frequency range, which is low- to mid-frequency.

The last new blip is the snitch, which has two vertical, upward trails on either side of a round ball. Similar to the original three blip types, the snitch occurs at both high and low frequency.

4 Determining Distinguishable Characteristics of Possible Sub-Classifications of Blip Glitches

The first step I took in finding specific attributes to sub-classify the blip glitches was to create a variety of histograms examining possible characteristics to find clusters of blips that might correspond to the six different types. I started with peak frequency and found a striking difference between the Livingston blips and the Hanford blips, as shown in figure 5a on page 6. Since the scale is dependent on the Hanford blips, figure 5b shows the Livingston blips by themselves.



(a) All O2 blips from Hanford and Livingston.

(b) Livingston blips

Figure 5

These histograms show a clear difference between the Livingston blips and the Hanford blips, which is somewhat surprising considering that all six blip shapes are found in both detectors. Most of the Hanford blips appear to have a peak frequency of 200 Hz, with a fairly steep dropoff on both sides of the 200 Hz spike. The Livingston blips, on the other hand, look to be more evenly spread out among the different frequencies, and there are considerably more low-frequency blips at Livingston. One possible explanation for this is that Livingston just happens to be more sensitive at low frequencies.

To get a closer look at the Livingston blips and see if there is anything significant going on within Hanford's 200 Hz spike, I made 2D histograms plotting peak frequency versus SNR (signal to noise ratio) for each detector. Figure 6 on page 7 shows these histograms. Here, the differences between Livingston and Hanford are even more clear. For example, while figure 5b does show a local maxima around 200 Hz, the 2D histogram of Livingston reveals that most of those blips have a low SNR, whereas a considerable amount of Hanford 200 Hz blips have a high SNR. Other points of interest include the 100 Hz peak at Livingston compared to Hanford, and low SNR blips at Livingston at 50 Hz compared to higher frequencies.

Both 2D histograms have high density bins, and we can use neural networks to investigate these possible clusters and see if they correspond to the six types of blips I identified in section 3.2.

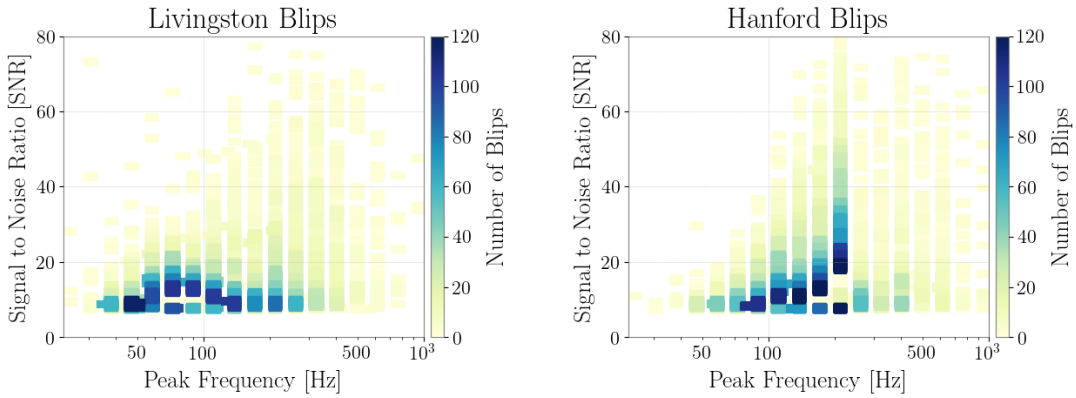


Figure 6: 2D histograms of the O2 blips from Livingston and Hanford.

5 Neural Network Implementation

At this point in the project, Neural Networks become desirable. We want to know if the six types of blips can be identified by a computer, not just one person looking at a couple hundred images. If a Neural Network can be used to find the differences between the types of blips, then we know that sub-classification is possible and can be implemented in GravitySpy to classify future blips and find their sources.

5.1 Building a Neural Network

There are countless ways to build and implement a Neural Network. The first step is deciding what type of learning we want: supervised learning, reinforcement learning, or unsupervised learning. Reinforcement learning can almost immediately be discarded. Since it depends on a function to reward the program, it is not very helpful for image classification, which cannot be easily translated into better states versus worse states. The ideal technique is unsupervised learning because we have a guess as to the number of classes and we don't have a large dataset of images that are already classified into those classes. However, unsupervised learning is extremely complex and only effective when there are clear subclasses to be found. Supervised learning, on the other hand, is a more accurate method because the neural network is trying to classify already-labeled data and therefore has a much better gauge of accuracy. Unfortunately, since we are trying to identify the classes themselves, this learning technique is also not ideal for our purposes, especially because I only have about 250 sub-classified blips that can serve as a labeled dataset. As a result, I implemented both supervised and unsupervised learning neural networks.

The next step in creating a neural network is deciding the type of layers to put in the network. Since I want my input set to consist of images, I used Convolutional Neural Networks (CNNs) in both the supervised and unsupervised networks. CNNs are better for images than a normal dense layer because they take all the surrounding points into consideration. Each image is stored as a multi-dimensional array, but normal dense layers can only interpret a single array, and can therefore only take into account pixels that are horizontally next to each other. CNNs, on the other hand, take in multi-dimensional arrays as input and use all the surrounding pixels to learn about the image.

The last thing to do before creating the neural networks themselves is to create the input dataset, i.e. the images. I created a database of all the blips from Hanford and Livingston, starting from O2, by plotting Q-transforms with a standardized time domain (20 seconds), cropped down to 0.22 seconds, similar the previous images I created. I left the energy (colorbar) to scale itself during the calculation, mostly because the disparity between blips with low energy and blips with high energy is so drastic that the shapes become difficult to distinguish with a standardized energy scale. An example of the images in the database is in figure 7. Since there are no axes or titles, the images are labeled with their GPS time and detector from their file names. This also allows us to access the images of specific blips, which is helpful for supervised learning.

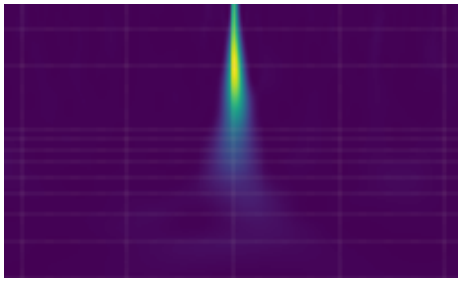


Figure 7: An example of the 120 x 200 images in the database used as input into neural networks.

5.2 Supervised Learning Neural Network

The goal of a supervised learning neural network is to see whether specific attributes of blips create clusters. We can label specific blips based on their omicron trigger values, and then evaluate the correlation based on the accuracy of the trained neural network. For example, I can label all the 200 Hz blips from Livingston with a "1," and label all the 200 Hz blips from Hanford with a "0," and see if the neural network can distinguish one "class" from another. If the Hanford 200 Hz blips are fundamentally different than the ones at Livingston, the neural network should have a high accuracy.

The example above is a case of binary classification, but we can create a neural network with as many classes as we want. Additionally, most neural networks have one input, but we can also create a two-input neural network that takes in both images and auxiliary data from the omicron trigger. As a result, I created four neural networks templates: one-input binary, two-input binary, one-input multi-class, and two-input multi-class. All four have almost exactly the same network of layers, but since they all have different uses, it is easy to implement different ideas. A condensed version of the layers in both of the two-input neural networks is in figure 8. The only difference between the two-input CNN and the one-input CNN is that the Flatten layer goes straight to the first Dense layer. To make this network multi-class instead of binary, the end output shape of the last layer is changed from 1 to the number of classes, and the loss function is changed from $binary_crossentropy$ to $categorical_crossentropy$.

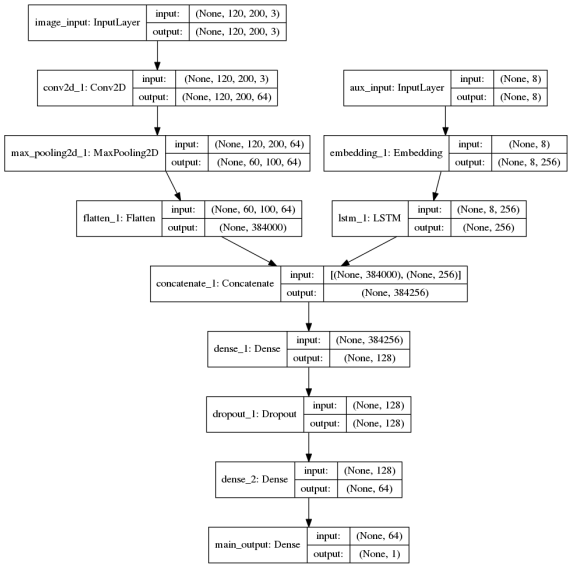


Figure 8: Condensed version of my two-input Convolutional NN. There are additional Conv2D and MaxPooling2D layers between the Input layer and the Flatten layer, and an additional Dense and Dropout layer before the main output layer.

The auxiliary input consists of about 6-12 omicron data values, including but not limited to q-value, confidence, SNR, bandwidth, duration, interferometer, and image status. The shape of the auxiliary input varies depending on what "classes" the neural network is looking at. Going

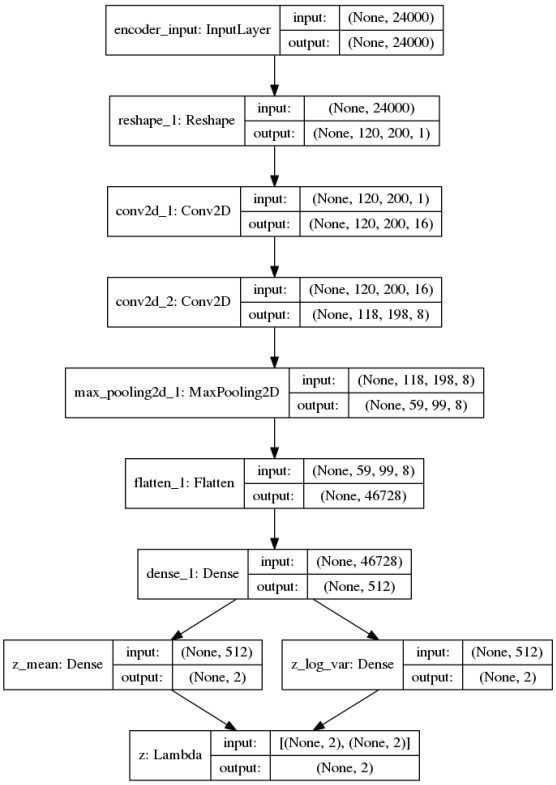
back to the earlier example about distinguishing Livingston 200 Hz blips from Hanford 200 Hz blips, that auxiliary input would not include the interferometer or the peak frequency of each blip.

Although adding auxiliary input seems helpful because it gives the neural networks more information, the performance difference between one-input neural networks and two-input neural networks is non-existent. Using images by themselves is exactly the same as using images along with auxiliary data. This is consistent with GravitySpy, which is directly dependent on images. The auxiliary data is not robust enough to aid in classification, both in GravitySpy's glitch classification and this project's sub-classification of blips. As a result, I mostly used the one-input neural networks. A summary of the different neural networks that I ran and their conclusions and implications is in section 6.

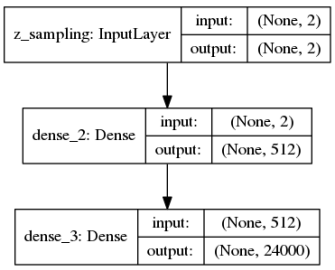
5.3 Unsupervised Neural Network

For my unsupervised neural network, I modified public source code[5] to make a Variational Autoencoder (VAE). This type of unsupervised neural network consists of an encoder and a decoder, where Bayesian statistic techniques are used between the encoder and decoder to better predict what the data should look like . Unlike supervised learning, which outputs a value corresponding to a class, autoencoders output new images. The new images then get put through the encoder part of the neural network, which also outputs meaningful statistical values. We can then create a scatter plot of the statistical values from all the images and see if there are visible clusters[4].

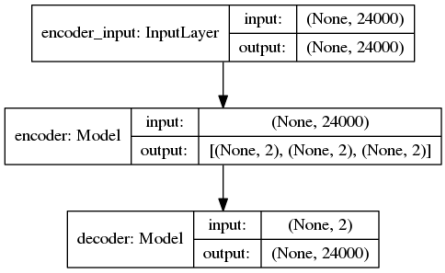
Since VAEs are so complex, it is difficult to implement convolutional layers within the autoencoder, especially when the input data is so large. However, I managed to sneak in a few layers into the encoder without overloading memory. The layers of the VAE neural network are in figure 9.



(a) Convolutional encoder layers



(b) Decoder layers



(c) Combined model summary

Figure 9: Layers of a Variational Autoencoder.

[5] GitHub source code

6 Neural Network Results

References

[1] M. Zevin *et al.*, "Gravity Spy: Integrating Advanced LIGO Detector Characterization, Machine Learning, and Citizen Science," *Class. Quant. Grav.*, vol. 34, no. 6, p. 064003, 2017.

[2] S. Mukherjee, R. Obaid, and B. Matkarimov, "Classification of Glitch Waveforms in Gravitational Wave Detector Characterization," *Journal of Physics: Conference Series*, vol. 243, 2010.

[3] S. Bahaadini, N. Rohani, S. Coughlin, M. Zevin, V. Kalogera, and A. K. Katsaggelos, "Deep multi-view models for glitch classification," *CoRR*, vol. abs/1705.00034, 2017.

[4] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013.