Audio Instrument Isolation via De-Noising Auto-Encoder

Eric Boulter
B.S. Candidate in Astrophysics at the George Washington University
ericboulter@gwu.edu

Research Advisor: **Alan Nash**
Astrophysics Faculty Advisor: **Sylvain Guiriec**
Music Faculty Advisor: **Dr. Douglas Boyce**

Abstract

This research project is centered on developing software that can isolate individual instrument signals called ”stems” from a commercial level multi- instrument audio file (.wav, .mp3) through machine learning techniques. For example, using the song Superstition by Stevie Wonder in this software, one could choose to pull out only the drum signal, thus isolating the drum from the entire song in order to place it in its own audio file. This process involves heavy data manipulation techniques from signal processing, as well as an Auto-Encoder machine learning neural architecture to effectively isolate individual instrument stems. Traditionally, Auto-Encoders are used for data compression. When the input and output are identical, the Auto-Encoder learns to ”encode” the data into a reduced form and to decode the reduced form back into its original dimensions. In this project, the Auto-Encoder architecture is given a full multi-instrument track as the input and an isolated instrument track as the output. This configuration trains the Auto-Encoder to treat all other instruments as ”noise” in order to isolate a single instrument signal. Currently, the training data used in this project is focused on isolating drum stems from Hip Hop/R&B/Pop Rock songs produced within the last thirty years. The project scope will be expanded to many more genres and instruments once the current area is perfected. This research will have a significant impact on the music community in regard to ”sampling” techniques, and it successfully demonstrates the De-Noising capabilities of Auto-Encoder neural architectures that can be applied to the field of signal processing.

1 Introduction

There has been a lot of work previously done on audio samples for various machine learning purposes. Many researchers and commercial entities have used machine learning algorithms for speech recognition, noise filtering, and audio compression. Various machine learning project have used Auto-Encoders to de-noise images, speech, and other types of signals with minor noise. This project intends to use a similar architecture to isolate specific instruments from multi-instrument audio files, thus treating non targeted instruments as ”noise”. In doing so, the basic structure of audio data must be understood.

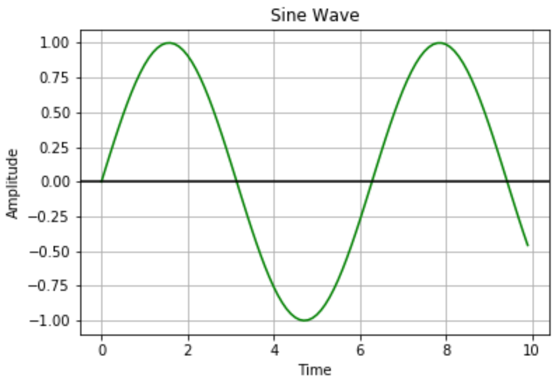
Digital audio data is a digital waveform representation of analog waveforms. Audio files have a sampling rate, or the number of samples taken per second of an analog waveform. An integer represents the amplitude of that analog waveform at each sample taken, creating a digital representation of the waveform. The sampling rate can be thought of as pixel resolution for images. To demonstrate this process, Figure[1] depicts a sine wave which is acting as the analog waveform. Digital waveforms representing the analog waveform with two separate sampling rates are plotted in in Figure[2a] and Figure[2b]. As the sampling rate is increased, then the digital waveform becomes a more accurate representation of the analog waveform.

For modern audio files, the most widely accepted sampling rate is 44.1kHz. The reason for this lies in a fundamental property called the Nyquist frequency. The Nyquist Frequency is the highest frequency a digital waveform can capture from an analog, which is equivalent to:

$$f_N = \frac{\gamma}{2} \tag{1}$$

where f_N is the Nyquist Frequency and γ is the sampling rate. This is because the digital conversion process must be able to capture two samples per sine wave to accurately represent the waveform. The reason audio files have a sampling rate

Figure 1: Analog Waveform



of 44.1kHz is because the range of human hearing goes from approximately 20Hz-20,000Hz. Using the Nyquist frequency formula, the minimum sampling rate needed to capture the edge of human hearing at 20,000Hz would be γ = 40,000Hz. The additional 4,100Hz is for additional detail in the higher frequency ranges.

Another important feature is the bit depth of audio files. The bit depth is the number of integers that can be used to represent the amplitude of the wave digitally. This can be considered the "dynamic range" of the audio file. The greater the bit depth, the greater difference in decibels (dB) between the loudest and quietest points in the audio. Bit depth can be represented by:

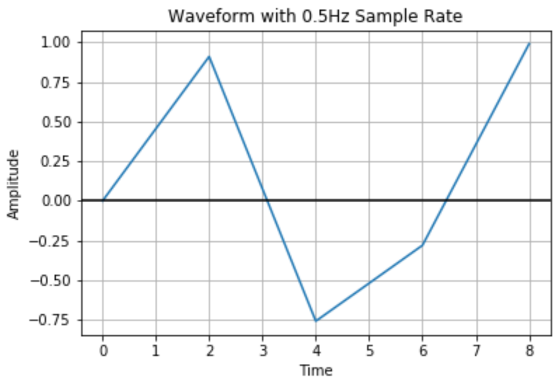
$$n = 2^{\beta} \tag{2}$$

where n is the number of integers from $(-n, n)$ available to express the waveform and β being the bit depth. Generally β is 16 or 24 for modern audio files.

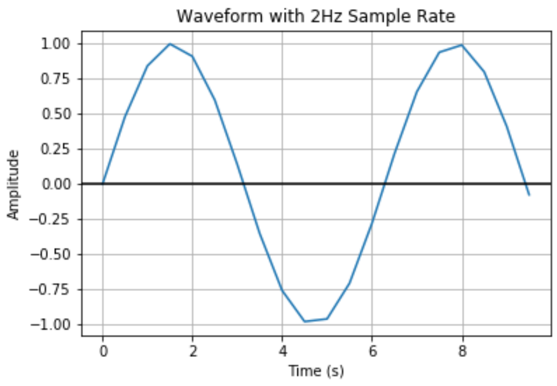
Once the digital waveform is constructed, the data is packed into an audio file (.wav, .mp3) as an integer array. This project extracts the integer arrays from the audio file to be processed for the machine learning networks.

2 Data Processing

The initial task of the project was to be able to identify drum audio samples from their audio data using a Convolutional Neural Network (CNN). The reason for this was to test the validity of the data set. Audio recognition with a CNN has already been done successfully by many research groups, so this process was to ensure my data



(a) 0.5Hz Sampling Rate



(b) 2Hz Sampling Rate

Figure 2: Sampling Rates for Digital Waveforms

set was strong enough for machine learning algorithms to process. A CNN is a machine learning network used for pattern recognition within data samples through a process called convolutions. In this process, the network has a window of determined size that parses through the entire sample and attempts to find "local" features within a data sample. This can be seen in Figure[3], where the red box represents the window size on the initial input array on the left. These features are then collected into a new layer called a Convolutional Layer, where the process can then be repeated onto that layer. This can be done as many times as needed until the network can identify enough features to classify the data properly.

For the CNN to work properly, the audio data needed to be samples of uniform dimensions from similar sources. For this project, I compiled around thirty different songs from the Hip-Hop/Rock/R&B with open source individual instrument stems from *CambridgeMT*.These audio files were picked to be the data files because of their rich and consistent percussive content, allowing for more drum samples per song than other

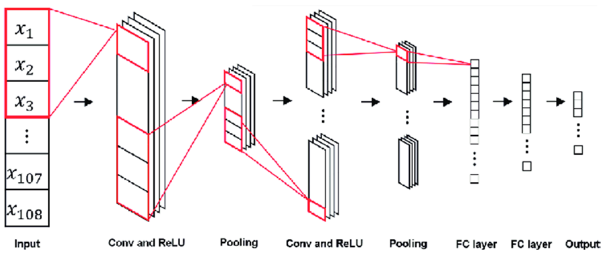


Figure 3: 1-Dimensional CNN Architecture

genres. These files were converted into integer arrays of audio data for further data processing.

The most significant information for identification in an instruments waveform is known to be the Attack, Decay, Sustain, and Release (ADSR) Envelope depicted in Figure[4]. The ADSR illustrates the acoustic differences in multiple instruments even when playing the exact same pitch (or frequency). For a drum sample, the ADSR envelope is approximately 200 milliseconds, as depicted in Figure[5a].In the 200ms sample, the attack, decay, sustain, and release of the drum hit can all be clearly identified. For other instruments such as a Synth in Figure[5b], the 200ms sample is not enough to contain their ADSR envelope. Therefore, 200ms was the choice of data sample size to isolate drum ADSR envelopes.

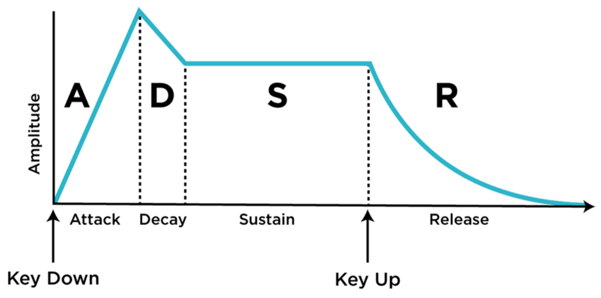
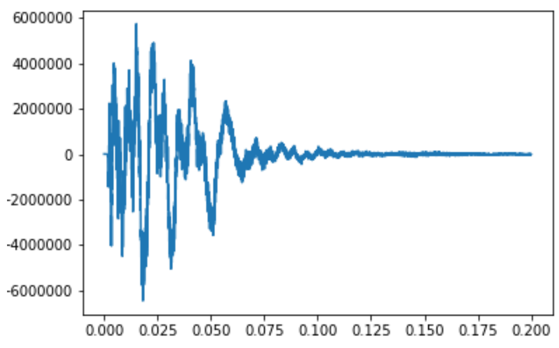


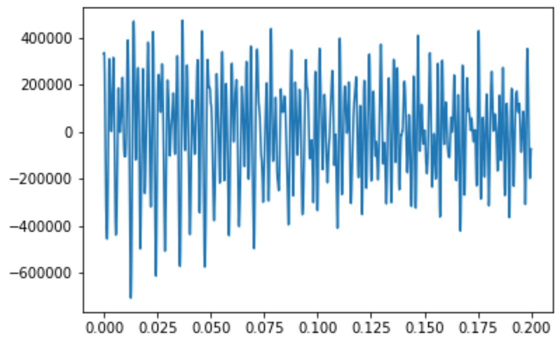
Figure 4: ADSR Envelope

As explained before, the sampling rate of audio files is $\gamma = 44.1$kHz, which translates to $44.1 * 10^3$ integers of data per second. For 200ms there are 8820 integer samples of audio data. These samples were converted into integer arrays and normalized from $(-1, 1)$ to eliminate differences between the bit depth of different samples. 8000 samples, half of them drum samples and the other half a mixture of guitar, keys/synth, and bass, were fed to the CNN.

This initial testing came out very poor with an accuracy of 51%, a score a human could score significantly higher on given the same data set. The



(a) Drum Hit at 200ms



(b) Synth at 200ms

Figure 5: ADSR Envelopes for Drum and Synth

problem with this data was that the CNN was over fitting the data, as there was way too much of it. Since the CNN looks at ”local” data within the samples, the CNN was fitting too many patterns that were not specific to the drum samples vs. not drum samples. To fix this issue, the data was then averaged in bins of 10 and sent again to the CNN for training. This testing came back with an accuracy of 83%, much higher than previous testing, but still not good enough.

To improve the accuracy further the data needed a process that would capture the significant information in a lower dimension data set. A real Fourier Transform was done on the data, giving its amplitude as a function of frequency and compressing the data by a factor of 2 (only real parts of the Fourier Transform were taken) as seen in Figure [6].This data used as training data for the CNN resulted in a 60% accuracy, which again was due to over fitting. The Fourier Transform data was averaged with a binsize of 10 as seen in Figure[7]. When this was sent to the CNN an accuracy of 99.7% was achieved.

Thus drum samples compared to not drum files

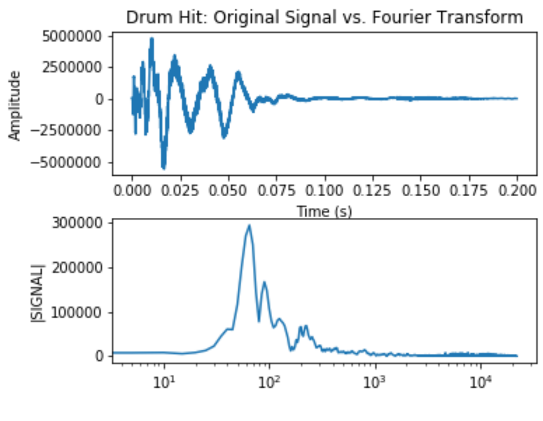


Figure 6: Original Drum Sample with Fourier Transform

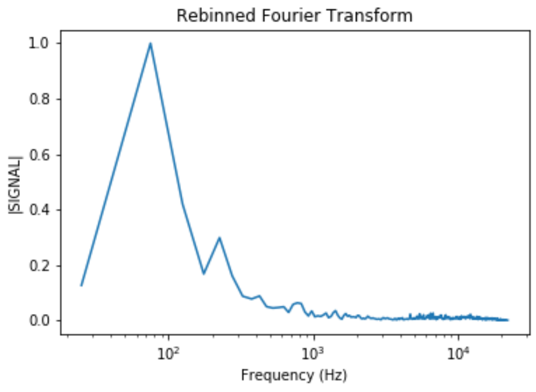


Figure 7: Binned FFT binsize=10

could be identified by the CNN at an accuracy of 99.7% with the binned Fourier Transform data. This ensured the validity of the data set for machine learning purposes and allowed the project to progress to the next step, the Auto-Encoder.

3 Auto-Encoder

A traditional Auto-Encoder is used for data compression. An Auto-Encoder takes in a data set of specific dimensions and learns how to compress that data into a smaller dimension representation of that data. This part of the process in the data ”encoder”. The second part of the process is teaching the network to reconstruct the original data from its encoded form, also known as the data ”decoder”. This process can be seen in Figure[8] below, where the Auto-Encoder is using the MNIST image data set. In this case, the reconstructed output data is compared to the input data and a loss value is given based on the error in the reconstructed output. As you can see in Figure[8], the reconstructed image of a two is slightly different than the original input, but still recognizeable. The system keeps training until the loss on the reconstructed output is minimized. This architecture has many uses including data dimensionality reduction and anomaly detection. However, re-configuring this architecture makes the Auto-Encoder capable to de-nosing.

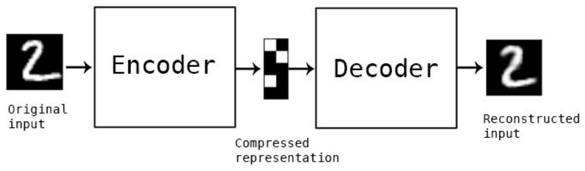


Figure 8: Auto-Encoder Architecture

A de-noisng Auto-Encoder architecture is one in which the input data is ”noisy”, and the output data is the ”de-noised” data. In this neural architecture, the Auto-Encoder learns to encode the noisy input and then decode it into its de-noised form. The reconstructed output is compared to the pure de-noised data and a loss value is given based on the error. The network keeps training to minimize this loss value and in turn create a successfully de-noised data set. For the audio data, this project looks at two separate methods for de-noising. The first of which is using the traditional Auto-Encoder architecture where the training data is reconstructing drum samples, and then using the model on mixed audio samples with drums for de-noising. The second of which is training the de-noising architecture with mixed samples as the input and isolated drum samples as the output. The two are explained in detail below.

3.1 Traditional Auto-Encoder: Data Testing

The first Auto-Encoder setup is the traditional encoder and decoder with the same input and output data used for training. The model will then predict on data sets that are of mixed instrument samples of the same length to see if the Auto-Encoder only reconstructs the drum data within the mixed samples.

For the traditional Auto-Encoder setup, the first neural architecture tested was made up of only two Dense layers, one for the encoder and one for the decoder. The first Dense layer had the number of nodes equivalent to the encoded length, and the second layer had the number of nodes equivalent to the input data length. With this very simple architecture and a *Sigmoid* and *tanh* optimization functions this network alone was able to reach a loss value of 0.002 (with *Mean Squared Error* as the loss function). In testing this model, the level of dimension reduction was extremely important to the success of the network. Looking at Equation (3) below:

$$\mu = \frac{L_e}{L_i} \tag{3}$$

where L_e is the encoded length, L_i is the input data length, and μ is the reduction factor, plotted in Figure[9] is the loss value as a function of μ. As seen in the figure, the loss reached a minimum of 0.002 at $\mu \approx 0.5$, which translates to a data reduction from 8820 to 4410. This process demonstrated the optimal compression for the Auto-Encoder was μ=0.5.

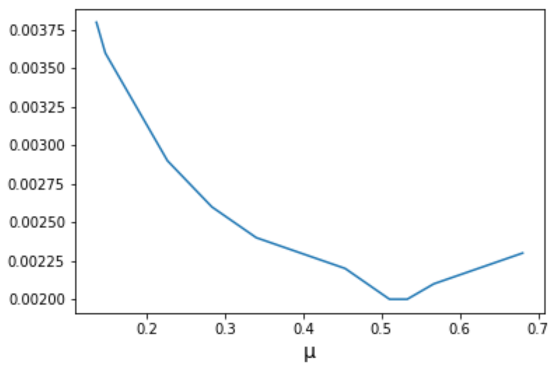


Figure 9: Loss as a function of μ

To actually look at what this data compression means, I have plotted in Figure[10] the compressed representation of the data samples the Auto-Encoder has created. As seen in the figure, the compressed input is not clearly identifiable to the human eye, but the Dense layers have encoded this input to be decoded back into the original waveform. Looking at the encoded representation gives a little insight on to the compression algorithms of the neural network. One interesting thing about the encoded data is that all the values are greater than zero with a range of $(0, 1)$, whereas the original waveform has a range of floats from $(0, 1)$.

The reconstruction of the wave forms are plotted in Figure[11]. As you can see they are not perfectly reconstructed even with a loss on the scale of $< 1e^{-4}$ but they are quite close. The first of two major error sources can be seen in the maximum amplitude, where the reconstructed wave-

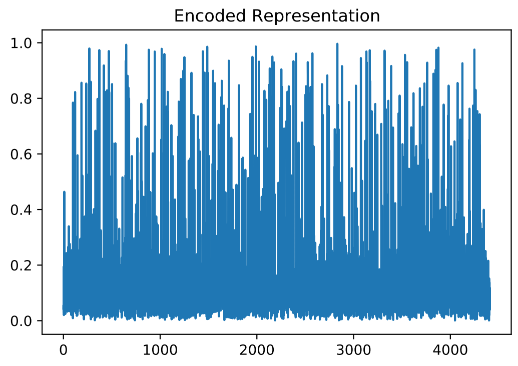


Figure 10: Encoded Waveform with μ=0.5

forms do not get back to the original maximum value of 1. The next greatest source of error are in the values $(-0.1, 0.1)$, demonstrating that the network has trouble discerning periods of silence and minimal acoustic energy.

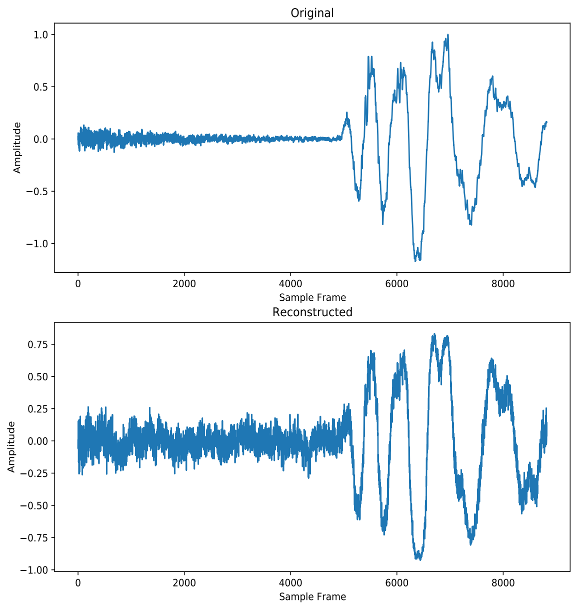


Figure 11: Original and Reconstructed Waveform

To improve upon this initial model, I changed the Auto-Encoder two a convolutional architecture to fix the errors of the Dense network. This network consisted of four convolutional layers, two for encoding and two for decoding. Each layer had 64 filters and a convolutional window size of 10 integers. In a traditional Auto-Encoder there is data reduction, however in this system the data was being encoded with dimension (8820, 64) to then be decoded back to the dimensions (8820, 1). This is because the Pooling layers meant for data reduction were losing too much local information in the waveforms and the network was not able to reconstruct them well. Plotted in Figure[12] are the reconstructed waveforms from the Convolutional Auto-Encoder which reported a validation loss of $\approx 1 * 10^{-4}$. As seen in the plots the wavefroms are practically identical with minimal error. The localised focus of the CNN was able to fix the errors in the Dense network and reconstruct the waveforms accurately.

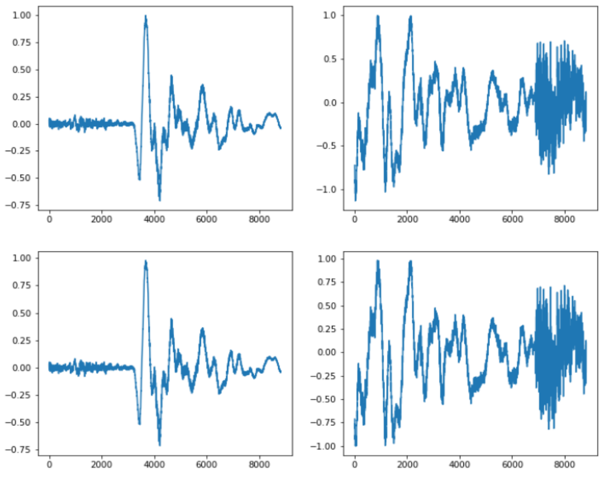


Figure 12: Reconstructed Waveform: CNN

4 De-Noising Auto-Encoder

Since the Traditional Auto-Encoder model was very successful, the first test at De-Noising was to run a mixed data set through the old model. This meant that the previous model would predict on mixed audio samples such as drum and bass together and return the former isolated. However, this test was not successful as the model recreated the entire waveform with no noise filtering. The idea was that a network trained on recreating drum waveforms may ignore the frequency content that did not resemble drums, however this was not the case.

Therefore, a new model and data structure were made to train a new De-Noising Auto-Encoder based on previous models made. This neural architecture would take in mixed audio samples of drums with all other instruments and learn to recreate only the drum track isolated. As seen in Figure[13], the input and output are frame the same 200ms frame of music, one isolated as a drum sample and the other as the fully mixed instrument audio. This data architecture teaches the network to treat all instrument signals besides drums as noise to reproduce the isolated drums.

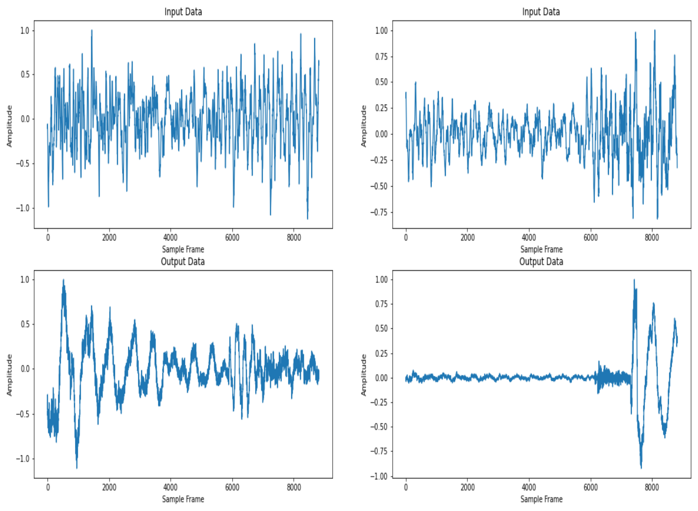


Figure 13: De-Noise Auto-Encoder Data Structure

The neural architecture for this system was the same two convolutional layers as the previous model now applied to de-noising. The results are plotted in Figure[14]. As seen in the figure, the de-noised signals achieved a loss value worse than that of the previous model. The De-Noising Auto-Encoder achieved a loss value of ≈0.03, still impressive but two orders of magnitude higher than the previous model. Further testing on varying neural architectures needs to be done to improve the model.

5 Future Research

Once this process is perfected, the same process will be extrapolated to expand the neural networks capabilities to more instruments. It is not known yet whether it will be multiple models or a one model de-noiser, but based on this research it is more likely the process would involve a model for each instrument. Also data architecture may have to be varied in order to train new models, most likely adjusting the data sample size to include each instruments ADSR envelope. Once these models are developed it is planned to convert these models into a full software package for users to apply on audio of their choosing.

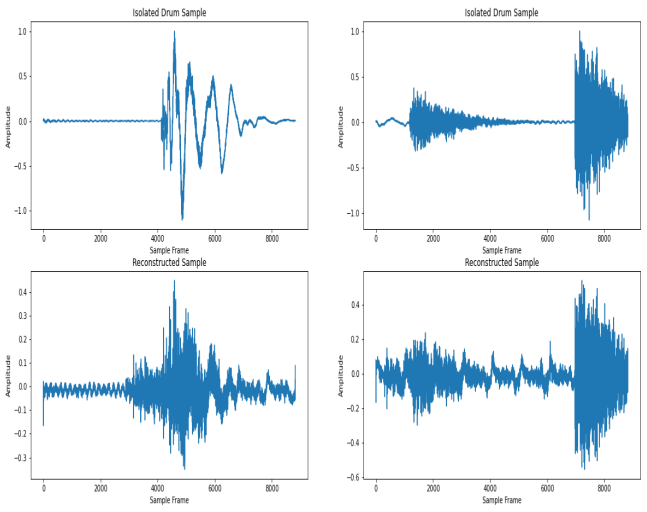


Figure 14: De-Noise Auto-Encoder Results

6 Conclusion

Overall the project has much more work in perfecting the de-noise algorithms, however thus far it has shown the capabilities of machine learning for de-noising signals. This is a very powerful property that can be utilized in many other areas of signal processing other than audio. In the music world, this would be an extremely powerful software for music composition through revolutionizing sampling techniques.

7 Acknowledgements

This research was done in conjunction with the principles and directives of the Innovation Lab of the GW Physics Department. Without the support of Physics Department and the Music Department this project would not have been possible. Special thanks to Alan Nash, Sylvain Guiriec, and Douglas Boyce on their continued support in this research process.