

# Unsupervised Techniques for Feature Extraction in Seismic Noise Data

Dylan Snover  
dsnover@ucsd.edu

Derek Lam  
ddl004@ucsd.edu

Brian Whiteaker  
bwhiteak@ucsd.edu

Anish Chivukula  
anchivuk@ucsd.edu

***Abstract* - Seismic data measurements are often corrupted by noise due to the sensors' environment and position, and the frequency bands of interest. In order to separate the seismic features from noise, several techniques have been tested out. This paper utilizes non-negative matrix factorization and convolutional autoencoders to extract important features. Both of these machine learning methods proved to be robust in determining the latent space of the input data. The extracted latent space was then clustered with k-means and hierarchical agglomerative clustering to categorize the dataset.**

## I. INTRODUCTION

Seismic data can be classified into three distinct categories: Earthquake signals, random noise(RN), and non-random noise (NRN). While earthquakes generally only account for <1% of all daily seismic recordings, non-random noise can account for up to ~74% of the data [12]. The goal of our project was to use deep learning techniques to differentiate the non-random noise and cluster the NRN into separable classes. For this purpose, we used spectrogram images, which shows both time and frequency domain features, as the starting point of our analysis. The spectrogram used in this paper contained only amplitude information, as the phase information was deemed unnecessary for our analysis.

Non-random noise varies greatly between datasets depending on the environment where the data is recorded. Many of the NRN in our data is spread out over a wide frequency band and is not distinct to the observer. Because there is no absolute way of labeling NRN during preprocessing, we did not have the luxury of working with a labeled training dataset. For this reason we chose to pursue two machine learning techniques, Convolutional Autoencoders and Non-Negative Matrix Factorization, to extract important features from the NRN. Both of these techniques involved reducing the dimensionality of the input spectrograms, and then using the features in the low

dimensional latent space to reconstruct the spectrograms during training to benchmark each algorithm. The features were then used to differentiate the NRN into separable classes using various clustering algorithms. In Section II, we will discuss similar projects in seismology, as well related works dealing with feature extraction and clustering.

## II. RELATED WORK

In recent work by Mousavi et al, a deep learning approach was applied to the task of distinguishing teleseismic event signals from local seismic event signals, and identifying the first motion polarity of locally recorded P-wave arrivals [2]. Their algorithm made use of a blend of deep neural network and clustering algorithm to achieve this goal. The signal environments in which the data was collected had a signal-to-noise ratio from 0 to greater than 4.

Mousavi et al, used deep learning to offer a solution to the problem of differentiating teleseismic wave arrivals from local ones [2]. These two distinct seismic signals have distinguishing characteristics in both the time and the frequency domain. The problem we looked to solve was rooted in extracting distinct features of the noise that would allow us to differentiate NRN into separable classes. While our dataset contained some noise that was easily distinguishable from other types, a large majority of the noise signals span a wide frequency band, and are difficult to distinguish as unique.

Johnson et al, have done extensive research on utilizing deep learning techniques in the field of dense nodal array seismology. One of these techniques involves using convolutional neural networks to differentiate between p-waves, random noise, and non-random noise [12, 14]. This work, and how it is directly related to the research presented in this paper, will be described in more detail in Section III.

### III. DATASET AND FEATURES

In 2014, a spatially dense geophone array, consisting of 1108 “Fairfield Nodal 5 Hz” geophones arranged in a 600x600m grid, was deployed along the San Jacinto Fault Zone [13]. The goal of the survey was to record small, local, seismic events that could not be picked up by larger seismic networks. The array recorded ground motion for over four weeks, sampling at 500 samples per second. The resulting dataset consisted of ~1.6Tb of waveform data [13].

C. Johnson et al. divided the original dataset into 4s wavelets and used these to generate spectrograms for each wavelet using the Short Time Fourier Transform [13, 15]. Labels were generated for samples of p-waves, random noise and non-random noise [13, 15]. He then used this labeled training data to train a 4 layer Convolutional Neural Network that split the data set into sets of P-waves, Random Noise and Non-Random Noise [13, 15]. Our group worked exclusively with the dataset that contained the non-random noise spectrograms.

The Non-Random Noise dataset consisted of 520,941 spectrograms, each with a height of 251 pixels and a width of 41 pixels, totaling 10,291 features. Each feature represents the amplitude of a certain frequency at a given point in time. Of the 520,941 samples in the dataset, 50,000 were set aside to use as test data, while the remaining 470,941 samples were split into a training set (80%) and a validation set (20%). An example of a spectrogram from our dataset is shown here to the right.

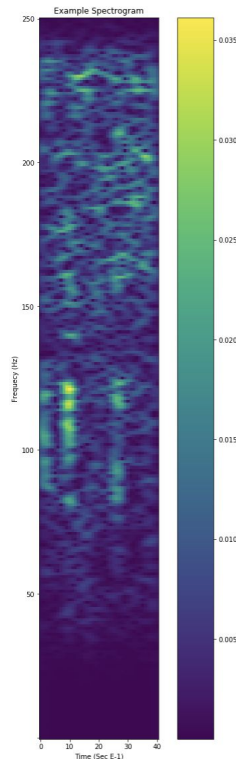


Figure 1. Example spectrogram

### III. METHODS

Clustering algorithms often suffer from the curse of dimensionality, in which high dimensional spaces are often subject to containing many useless or sparse features [3]. Dimensionality is especially an increasing concern in the task of clustering, which often uses a distance metric to determine the cluster that a data point should be assigned to. In high

dimensional spaces, samples taken from a dataset will typically be very similar Euclidean distances from one another, whereas low dimensional spaces are more likely to contain samples that can be more easily clustered into distinct groups. For this reason, the first part of this project involves feature extraction and dimensionality reduction on our high dimensional spectrograms to find a latent, or low dimensional, space in which they can be easily clustered.

In order to cluster the non-random noise spectrograms, dimensionality reduction techniques were first performed on the datasets by utilizing convolutional autoencoders (CAE), and non-negative matrix factorization (NMF). After extracting a latent space using both of these methods, several unsupervised algorithms, k-means and hierarchical agglomerative clustering were performed on the reduced dataset.

#### A. Non-Negative Matrix Factorization

Given a matrix  $\mathbf{X} \in \mathbb{R}^{m \times n}$  of the shape where  $m$  is the number of features and  $n$  is the number of samples, and where all the elements of  $\mathbf{X}$  are non-negative, NMF factorizes this matrix into two matrices  $\mathbf{W} \in \mathbb{R}^{m \times p}$  and  $\mathbf{H} \in \mathbb{R}^{p \times n}$  such that:

$$\mathbf{X} = \mathbf{W}\mathbf{H}$$

Where all the elements of  $\mathbf{W}$  and  $\mathbf{H}$  are also non-negative [4]. We consider  $\mathbf{W}$  to be the basis matrix, and  $\mathbf{H}$  to be considered our reduced latent space of dimension  $p$ . This factorization allows the original dataset to be reconstructed using a linear combination of the basis column vectors in  $\mathbf{W}$  using the extracted feature space from  $\mathbf{H}$ . Here, the dimension  $p$  is significantly lower than  $m$  and  $n$ , and satisfies the constraint  $p < \min(m, n)$ . By tuning the hyperparameter  $p$ , we can choose the desired dimension of our latent space. Given unseen data samples, such as those outside of the training set, the latent space for these examples can be computed using  $\hat{\mathbf{H}} = \mathbf{W}^\dagger \hat{\mathbf{X}}$ , where  $\hat{\mathbf{H}}$  is the latent space for the new examples in  $\hat{\mathbf{X}}$ , and  $\mathbf{W}^\dagger$  is the pseudoinverse of  $\mathbf{W}$ . We implement NMF using sklearn’s decomposition API, initializing  $\mathbf{W}$  and  $\mathbf{H}$  using non-negative double singular value decomposition [5]. The API determines the factorization by minimizing the loss function  $\|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F^2$  with a coordinate descent solver.

#### B. Convolutional Autoencoder

Convolutional autoencoders are a class of convolutional neural networks, which are deep learning techniques that are often used for classification of image

datasets. CNN's are generally take in some image  $\mathbf{X} \in \mathbb{R}^{m \times n}$  as an input, and have the output be a vector  $\mathbf{y} \in \mathbb{R}^d$ , where  $d$  is the number of classes in a classification problem. Unlike a regular feedforward neural network, CNN's are able to take advantage of spatial information contained in images - through the use of convolutional filters, which will be explained later. The CAE is made out of three main parts: an encoder, a latent space representation, and a decoder. Instead of classifying images like a standard CNN, the goal of the CAE is to encode some latent space representation of the input, and then to reconstruct the original input at the output by decoding this representation. To accomplish this task, the target of the CAE is set to be the same as the input such that the output is  $\mathbb{R}^{m \times n}$ . The CAE for this work was implemented using PyTorch.

For a convolutional autoencoder, the dimensions of the output volume of each convolutional and pooling layer need careful calculation.

$$W_A = \frac{X_{in} - X_{filter} + 2 * Pad}{Stride} + 1$$

$$H_A = \frac{Y_{in} - Y_{filter} + 2 * Pad}{Stride} + 1$$

The number of filters in each convolutional layer determine the output volume depth. As the convolution operation is often visualized as sliding a filter across an input, the stride is the number of pixels that are skipped as the filter moves across the input volume. Each convolutional filter has the effect of reducing the dimensions of the output volume width and height. In order to compensate for this dimensionality reduction, padding is often times employed in deep convolutional neural networks. Pooling layers are used for dimensionality reduction when appropriate.

In addition to the convolutional layers, various other aspects of the network architecture must be considered such as choice of non-linearities. A common choice in wide use as of this writing is the Rectified Linear Unit [7]. This unit is defined as,

$$f(x) = \max(0, x).$$

A newer version, called an ELU or Exponential Linear Unit, was found to be particularly useful on this data. The ELU is defined as,

$$f(\alpha, x) = \begin{cases} \alpha(e^x - 1) & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$$

Here  $\alpha$  is an adjustable parameter [7].

For efficient learning we tested the extremely popular Adam stochastic gradient descent optimizer. Adam is derived from adaptive moment estimator. This is considered by many to be the best optimizer in use, yet we found in our work that RMSprop provided better results. Root mean square propagation makes use of a running average on the weight gradients to control the step size taken for a given weights direction. If the past gradients of a weight have been consistent, so that the average has not fluctuated in large amounts, the step will be larger. If the gradients have fluctuated in size, such as the situation of an ill-conditioned weight space, then the step is smaller. The key is the division by the moving average,

$$v(w, t) = \gamma v(w, t - 1) + (1 - \gamma)(\nabla Q_i(w))^2.$$

The weight update is,

$$w = w - \frac{\eta}{\sqrt{v(w, t)}} \nabla Q_i(w)$$

The value  $\gamma$  controls the impact of past gradient values and  $Q$  is the cost function. For our CAE model,  $Q$  was the mean squared error function.

RMSprop was suggested by Hinton as a mini-batch adaptation of Rprop, which required full-batch learning. RMSprop provides some interesting qualities such as strong performance on data coming from non-stationary distributions and due to its choice of step sizing it tends to escape from saddle points quickly [8].

Deeper neural networks often suffer from initialization and vanishing gradient problems when training. Activation functions can saturate and stifle the ability of the network to learn. One technique used to alleviate these concerns is batch normalization. Batch normalization will compute the mean and variance of each mini-batch, and normalize the output of a layer.

$$x'_k = \frac{x_k - E[x_k]}{\sqrt{Var[x_k]}}$$

### C. K-Means Clustering

Given  $n$  data samples, k-means assumes that  $k$  clusters or groups exists among these samples. The algorithm attempts to find the centroid of each cluster, computed as the mean of all the samples in a given cluster. Through an iterative process, each sample  $\mathbf{x}$  is assigned to a cluster  $i^*$  using a rule that minimizes Euclidean distance:

$$i^*(x) = \underset{1 \leq i \leq k}{arg \min} \|\mathbf{x} - \mathbf{u}_i\|_2^2$$

After assignment, the centroid of each cluster is recomputed

using the sample mean  $\mathbf{u}_i = \frac{1}{n} \sum_{j=1}^n \mathbf{x}_j$  where  $n$  is the number of samples in cluster  $i$  and  $\mathbf{x}_j \in \mathbf{S}_i$ , the set of samples in cluster  $i$ . The process is repeated until little to no change occurs in the assignment of samples from one iteration to the next. Using sklearn’s API, each centroid is initialized using the k-means++ algorithm, which aims to select initial means that are not similar to each other [9].

K-means++ randomly picks a point  $\mathbf{x} \in \mathbf{S}$ , the set of all available data samples, and assigns that point as the first centroid such that  $\mathbf{u}_1 = \mathbf{x}$ . From there, the other centroids are initialized by choosing other data samples  $\mathbf{x}$  with a weighted probability ( $D^2$  weighting):

$$\frac{D(\mathbf{x})}{\sum_{\mathbf{x} \in \mathbf{S}} D(\mathbf{x})^2}$$

Where  $D(\mathbf{x})$  is defined as the shortest distance from a data sample  $\mathbf{x}$  to one of the already chosen centroids.

#### D. Hierarchical Agglomerative Clustering

For a set of observations in a given space, agglomerative clustering approaches the problem of finding clusters without any assumption of structure or shape to the cluster. This is done by applying a linkage criteria and a metric. The linkage criteria will be maximized under some metric chosen accordingly. This process begins by treating each observation as its own cluster; by analogy, these can be thought of as leaves in a tree graph. Using the metric, the leaves are combined with their nearest neighbor which satisfies the linkage criteria. This merging is carried out recursively to create larger clusters of no particular shape.

For this work, various metrics were explored and applied to the encoded representations produced by the CAE and NMF. These metrics were  $L_2$ ,  $L_1$ , cosine and Mahalanobis distance. Both  $L_2$  and  $L_1$  proved useful and achieved subjectively “good” clusters when applied with Complete Linkage. The Complete Linkage for a distance,

$$D(X, Y)$$

between clusters  $X$  and  $Y$  is

$$D(X, Y) = \max_{x \in X, y \in Y} d(x, y)$$

Here  $x \in X$  and  $y \in Y$  with  $d(x, y)$  being the distance between these elements. Among all cluster pairs the smallest distance pair is combined to form a new cluster. The cluster distance is considered to be the largest pairwise distance between elements of clusters.

#### E. T-Distributed Stochastic Neighbor Embedding

Determining the effectiveness of clustering is difficult as an unsupervised learning problem. To help visualize the results of our clustering algorithms, we make use of t-distributed stochastic neighbor embedding (t-SNE), a method to visualize high-dimensional data on a 2D plot. Unlike principal component analysis, which lowers high dimensional spaces through a linear mapping, t-SNE is a form of manifold learning that can capture non-linear relationships and local structure in our data [11]. Given a set of examples of  $\mathbf{x}_i \in \mathbb{R}^p$  after dimensionality reduction from the CAE or NMF, t-SNE maps these points to a set of points  $\mathbf{y}_i \in \mathbb{R}^q$ , where most often  $q = 2$ . t-SNE attempts to minimize the following cost function:

$$C = KL(P||Q) = \sum_i \sum_j \log \frac{p_{ij}}{q_{ij}}$$

Where  $P$  represents the high dimensional space in  $\mathbb{R}^p$  and  $Q$  represents the lower dimensional space in  $\mathbb{R}^q$ .  $p_{ij}$  and  $q_{ij}$  represent joint probabilities that define pairwise similarities, i.e. the probabilities that one data sample would pick another data sample as its neighbor after centering a Gaussian distribution about the first sample.

### IV. EXPERIMENTS, RESULTS, AND DISCUSSION

To evaluate the effectiveness of our dimensionality reduction methods, we compute the reconstruction for each spectrogram in our training, testing, and validation sets and determine the Mean-Squared Error (MSE) between each original spectrogram and its corresponding reconstruction. The MSE metric is defined as:

$$MSE = \frac{1}{n} \sum_{i=1}^n (\mathbf{x}_i - \hat{\mathbf{x}}_i)^2$$

Where  $\mathbf{x}_i$  is the  $i^{th}$  sample in a dataset,  $\hat{\mathbf{x}}_i$  is its reconstruction, and  $n$  is the total number of samples. By minimizing the MSE and obtaining as best of a reconstruction as possible, we hope that the reduced latent space obtained will be representative of the overall dataset and thus be suitable for clustering algorithms.

#### A. Non-negative Matrix Factorization

Due to the large amount of data samples and the associated runtime complexity with non-negative matrix factorization, hyperparameter tuning was done on a smaller training set comprised of 15,000 spectrograms. The following results were obtained on the reduced training set of 15,000 spectrograms, with a latent space dimension of 250:

Reconstruction Error (MSE) on Reduced Training Set		
Initialization Type	Solver	Train Error
Random	Coordinate Descent	2.824e-05
NNDSVD	Coordinate Descent	2.818e-05
NNDSVDA	Multiplicative Update	3.162e-05

Given the results on the reduced training set, we opted to use nonnegative double singular value decomposition (NNDSVD) as the initialization for  $\mathbf{W}$  and  $\mathbf{H}$ , and the coordinate descent solver for NMF [6]. In order to train our NMF model on the complete training set, as well as validate our data and test our data on the validation and training sets respectively, the model had to be trained on a server containing enough computational power. We were able to utilize the UCSD NoiseLab Vellella Computer Cluster to train the NMF model. Using these model parameters, we achieved the following reconstruction errors on a split training, validation, and test set:

Model	Train Error	Validation Error	Test Error
NMF	2.86e-5	2.87e-5	2.85e-05

We found that NMF was extremely robust in terms of reconstruction error, even on data that the model was not trained on. We hypothesize that the basis vectors the algorithm stores in the matrix  $\mathbf{W}$  are very representative of the overall dataset, allowing it to accurately reconstruct any spectrogram. Below, we include an example of an original spectrogram from the test set, and its reconstruction:

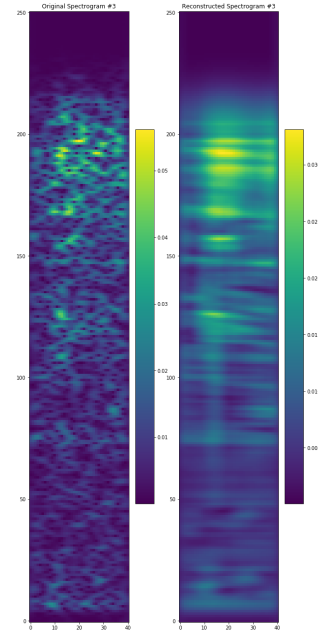


Figure 2. NMF reconstruction (right), Target (left)

We attempted k-means clustering, as well as hierarchical agglomerative clustering using both the cosine metric, with complete linkage, and  $L_2$  distance metric with ward linkage, on the latent space obtained from NMF. To estimate the amount of clusters that exist in the NRN dataset, we first ran k-means on the latent space and then plotted the sum of distances from each data sample to its nearest centroid.

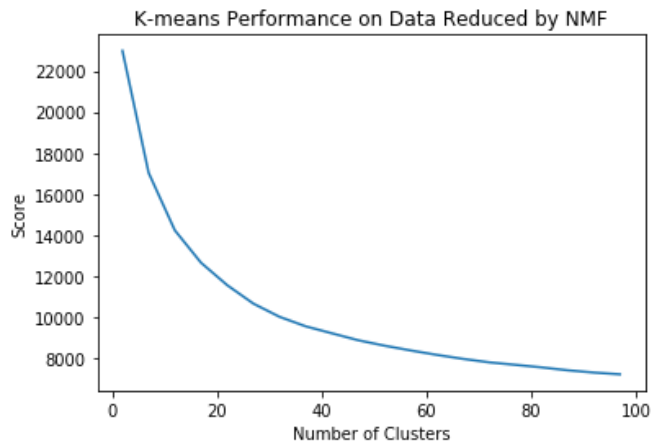


Figure 3. Number of clusters vs K-means score using latent space from NMF

Although there isn't a clear "elbow" in the k-means plot, we estimated that there were approximately 5-10 true clusters in the dataset. Below, we show a t-SNE visualization of the clusters obtained from k-means and HAC with the aforementioned metrics, using 8 clusters[11].

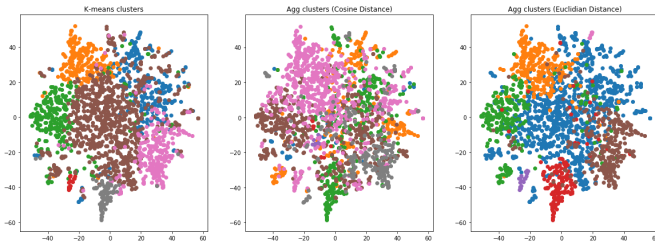


Figure 4. Clustering with K-means (left), HAC with cosine metric (middle), HAC with  $L_2$  metric (right)

We found that the two methods found very similar clusters within the data, indicated by the different colors in the plots. However, HAC with a cosine distance metric seemed to perform the worst, as the t-SNE visualization shows that many of the clusters overlapped with each other not leading to any clear separation within the data.

### B. Convolutional Autoencoder

Different types of NN architectures were explored in order to obtain better clustering and reconstruction performance. Specifically, CNN architectures with two, three, and six layers were explored to obtain optimal performance.

The six layer CAE was able to reduce the latent space to 24 features, with a test loss of  $5.34e-6$ . The reconstructed image from the latent space is shown on the left of Figure 1, while the target image is shown to the right. Observations from the entire test set indicate that the 6-layer CAE accurately learned the high intensity regions of the spectrogram, but had difficulty reconstructing the lower and middle intensity regions of the spectrogram. The six layer CAE was trained over 5 epochs with rms-prop as the optimizer, and batch normalization. In order to train the network with a learning rate of  $1e-2$  and ensure healthy gradient flow, the convolutional layers were followed by max pooling, and batch normalization layers.

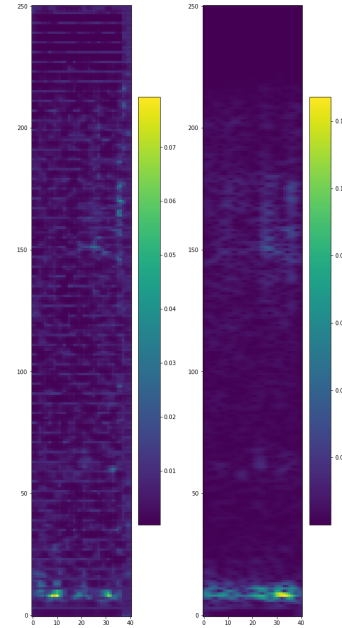


Figure 5. 6 Layer CAE Reconstruction (Left), Target (Right)

Reconstruction Error (MSE) on Train, Test, Validation			
Model	Train Error	Validation Error	Test Error
CAE-2	$6.30e-5$	$2.53e-4$	$9.16e-3$
CAE-3	$6.30e-5$	$2.53e-4$	$8.28e-3$
CAE-6	$5.34e-6$	-	$8.88e-3$
CAE-2*	$6.19e-5s$	-	$8.33e-3$

Moving beyond the architecture it was necessary to tune hyperparameters for best reconstruction results. The validation set was omitted in order to maximize the number of training examples available. Tests were performed using increasing numbers of epochs training the network up to 15 epochs. During this test the default implementation of Adam and RMSprop were used with the CAE-2 network

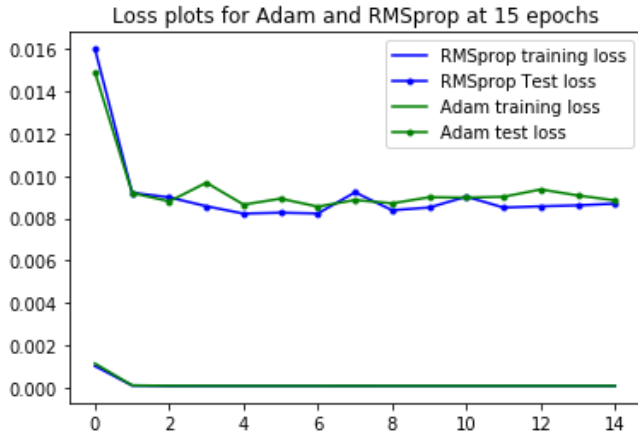


Figure 6. Performance test of Adam versus RMSprop optimizer for CAE-2 model.

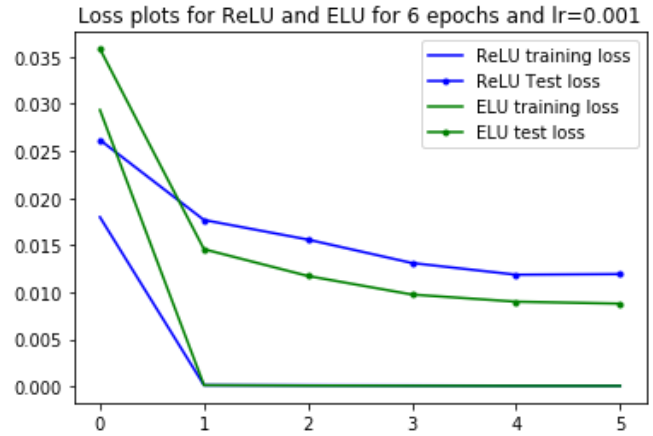


Figure 8. Tests of nonlinearities for CAE-2 model.

architecture. The nonlinearity was ELU for both versions. Overtraining was observed beginning after the fourth epoch. Overtrain was indicated by a subtle upward slope in the test set loss as the training set loss continued with perhaps a minor decrease in loss. This indicated the network was beginning to memorize the training set, or in another sense, to learn the noise of the training set.

The final tests ran revolved around the RMSprop optimizer [8]. Of the parameters available in the PyTorch Optim implementation of RMSprop the alpha and momentum parameters were tuned. For the final tuned model of the CAE-2 architecture alpha was set to 0.4 and momentum was 0.5. With these settings the provided cluster examples were found. These settings provided the most coherent cluster examples

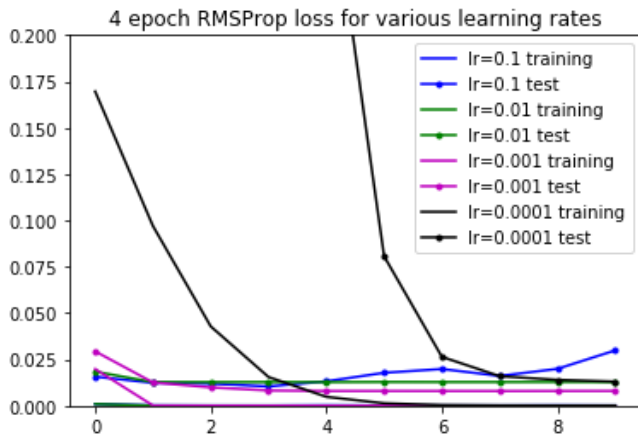


Figure 7. Tests of various learning rates for CAE-2 model.

Next, the learning rates were varied for 10 epoch runs. The learning rate values were  $lr = 0.1, 0.01, 0.001, 0.0001$  of which 0.001 performed best after 4 epochs. This led to the use of 0.001 in the following test of ReLU versus ELU performance. For this test ELU clearly outperformed ReLU over 6 epochs of training.

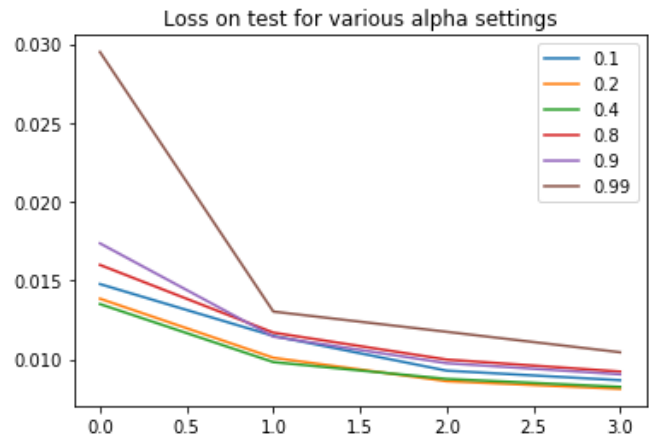


Figure 9. Tests of alpha values for CAE-2 model.

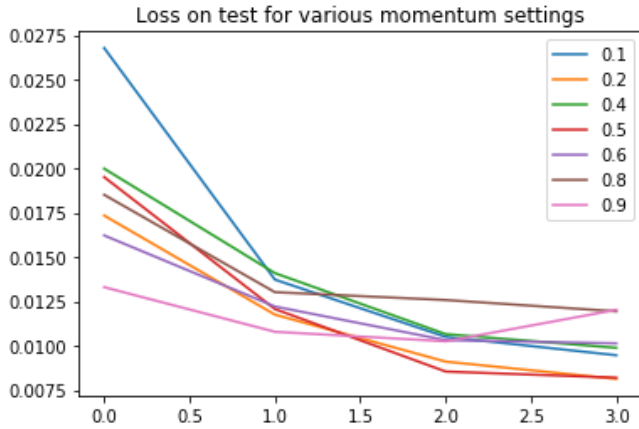


Figure 10. Tests of momentum values for CAE-2 model.

To form the clustering using HAC a dendrogram was constructed [10]. The heuristic employed was to evaluate the distances between merges within a cluster. For a dendrogram the vertical lines represent the distance; a merge between two lower clusters contains a horizontal bar connecting them. If the larger the vertical distance before a horizontal merge bar indicates that the two clusters being merged were far apart under the metric in use. Within clusters it is clearly preferable to have merges with short vertical distance.

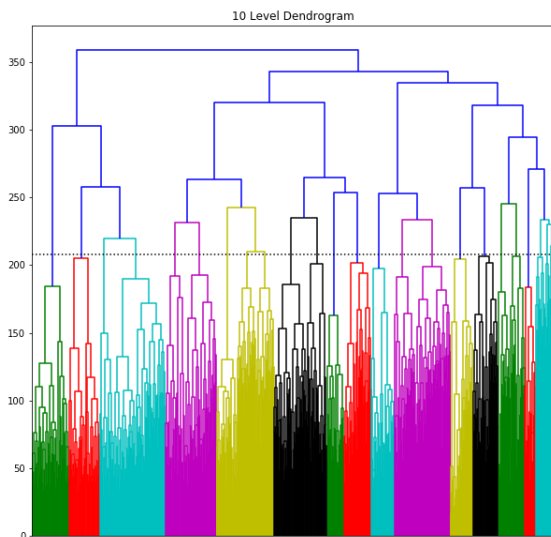


Figure 11. Dendrogram for HAC with the  $L_2$  metric. The horizontal line intersects 26 vertical lines giving 26 clusters.

An example of a cluster produced utilizing the CAE method is shown below.

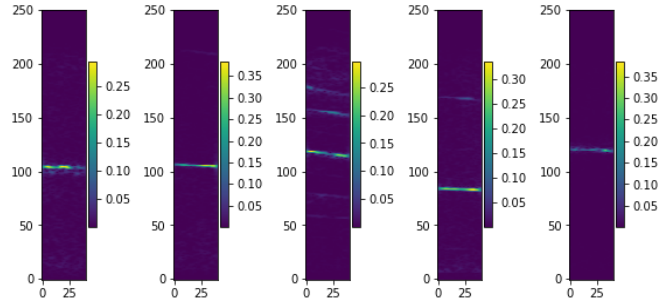


Figure 12. Example cluster using latent space from the CAE

### C. Analysis Of Cluster Separations

The NMF method of feature extraction led to clustering between 5-10 separate classes of spectrograms. The clusters seen in section A above, shows the case when the data was split into 8 classes. These classes are representative of different types of non-random noise, which we believe is related to their source. The region of the San Jacinto Fault Zone where this data was collected is relatively quiet. Over the course of the 4 weeks that the data was being recorded, the array likely picked up noise generated by: air-traffic, car-traffic, foot-traffic, wind, instrumental noise caused by the geophone itself, as well as other unknown sources.

Inspecting the clusters generated by the NMF model, we observed that the smaller clusters were very distinct and only included spectrograms that were similar in their features. Larger clusters were less distinct and often included spectrograms that had no observable similarities to the majority of the samples in the group. We believe that one cluster in particular is very accurate. This cluster, known to us only as "Label 2" consisted of 6,637 spectrograms that showed a very strong signal at ~50Hz across the 4 second time window. The following figure shows the spectrograms for three examples of this signal.

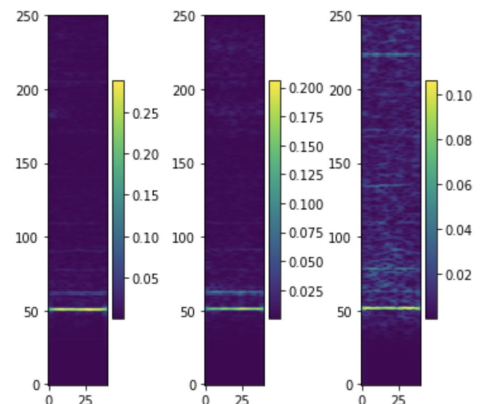


Figure 13. Example cluster using latent space from NMF

This signal is a topic of study for Geophysicists who use the Fairfield Nodal 5 Hz Geophones [1]. The strong signal at 50 Hz in these geophones is the result of instrumental noise. While this particular cluster is very accurate and only groups spectrograms that exhibit this 50 Hz signal, we believe that there are other examples of instrumental noise that are not included in this cluster. The reason for this is that variable wind speed and temperature fluctuations modulate the frequency of the signal, and this noise can be seen at resonant frequencies up to 150 Hz [1].

## V. CONCLUSION AND FUTURE WORK

The problem of extracting features from non-random noise data was tackled with both linear (NMF) and nonlinear (CAE) techniques. The seismic data was presented as an image in a spectrogram. Some of the strengths of neural network architectures have been in image processing and classification. The work presented in this paper lends itself to a similar result, with the CAE performing better in both latent space reduction, and spectrogram reconstruction. An interesting aspect of the CAE is that the cluster groups contained spectrograms with similar features, but not necessarily at the same frequency. For instance, a spectrogram containing one narrow high intensity frequency can appear at various frequencies in this dataset, and all spectrograms with this characteristic were often found in the same cluster. This lends itself to the conclusion that the CAE, encoded features without regard to localization. However, in spite of this, the decoder was able to properly reconstruct the image from this non-localized representation.

While this project only spanned ~8 weeks, given more time we would have liked to optimize the hyperparameters that were used in the CAE and NMF model. Additional research can be done on both models that would have optimized the feature extraction and reconstruction process. Future work also includes improving the clustering results to better differentiate the NRN into separable classes, and identifying the noise sources for these non-random noise.

## VI. CONTRIBUTIONS.

Derek and Dylan primarily worked on feature extraction with non-negative matrix factorization, along with clustering the reduced latent space obtained from this method. Brian and Anish primarily worked on feature extraction with the convolutional autoencoder, along with clustering the reduced latent space obtained from this model.

This project would not have been possible without the help of Christopher Johnson, who provided us with the dataset.

## VII. REFERENCES

- [1] Christopher W. Johnson, Frank Vernon, Nori Nakata, Yehuda Ben-Zion; "Atmospheric Processes Modulating Noise in Fairfield Nodal 5 Hz Geophones." *Seismological Research Letters* 2019 [Online] doi: <https://doi.org/10.1785/0220180383> [Accessed June 14, 2019]
- [2] S. Mostafa et al, "Unsupervised Clustering of Seismic Signals Using Deep Convolutional Autoencoders," *IEEE Geoscience and Remote Sensing Letters*, pp. 1-5, 2019.
- [3] M. Steinbach, L. Ertöz, and V. Kumar. Challenges of clustering high dimensional data. In L. T. Wille, editor, *New Vistas in Statistical Physics – Applications in Econophysics, Bioinformatics, and Pattern Recognition*. Springer-Verlag, 2003.
- [4] I. Buciu, "Non-Negative Matrix Factorization a New Tool for Feature Extraction: Theory and Applications", *Int'l J. Computers Comm. and Control*, vol. 3, pp. 67-74, 2008.
- [5] F. Pedregosa et al, "Scikit-learn: Machine Learning in Python", *JMLR* 12, pp. 2825-2830, 2011.
- [6] C. Boutsidis, E. Gallopoulos, "SVD Based Initialization: A Head Start for Nonnegative Matrix Factorization", *Pattern Recognition*, vol. 41, no. 4, pp. 1350-1362, 2008.
- [7] "PyTorch," 2018. [Online]. Available: <https://pytorch.org/>.
- [8] T. Tielmen, "Homepage CSC231," 2014. [Online]. Available: [https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture\\_slides\\_lec6.pdf](https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf). [Accessed June 9, 2019].
- [9] D. Arthur, S. Vassilvitskii, "K-Means++: The advantages of careful seeding", *Proc. Symp. Discrete Algorithms*, pp. 1027-1035, 2007.
- [10] E. Jones et al, "SciPy: Open Source Scientific Tools for Python", 2001 [Online]. Available: <https://docs.scipy.org/doc/scipy/reference/cluster.hierarchy.html>. [Accessed June 1, 2019]

[11] L. van der Maaten, G. E. Hinton, Visualizing data using t-SNE. *J. Mach. Learn. Research* 9, 2579–2605 (2008).

[12] C. W. Johnson, ECE 228. Class Lecture, Topic: Machine Learning with CNN in Seismology, University of California San Diego, San Diego, CA, Apr. 29, 2019.

[13] Y. Ben-Zion, F. Vernon, Y. Ozakin, D. Zigone, Z.E. Ross, H. Meng, M. White, J. Reyes, D. Hollis, M. Barklage, “Basic data features and results from a spatially dense seismic array on the San Jacinto fault zone.” *Geophysical Journal International*, vol. 202, no. 1-11. [10.1093/gji/ggv142]

[14] H. Meng, Y. Ben-Zion, C.W. Johnson, “Detection of random noise and anatomy of continuous seismic waveforms in dense array data near Anza California.” *Geophysical Journal International* 2019 (in review).