GROUP 23 - EMBEDDED CLUSTERING OF SEISMIC DATA

John O'Boyle, Cameron Lewis, and Heather Huntley

University of California San Diego, La Jolla, CA 92093-0238

ABSTRACT

A recent publication submitted to JGR: Solid Earth Unsupervised Deep Clustering of Seismic Data (Jenkins et., al 2021) (DCSD hereafter), used a deep learning approach to analyze seismic activity on the Antarctica Ross Ice Shelf [1]. The approach used an auto-encoder to learn the latent feature space of the seismic data. Using this latent feature space, the paper applied K-means to the lower-dimensional data. The point of learning the latent feature space is used to lower the dimensions of the data in order to more efficiently apply other operations.

We are interested in modifying the network aiming to improve the models performance. In particular, we are interested replacing the auto-encoder with a U-Net (Renneberger et., al 2016). Using the latent space as the bottom of the "U" before the first upsampling layer to cluster upon, we will compare clustering performance between a U-Net and an auto-encoder. We are also interested in replacing the K-means clustering method with a Gaussian Mixture Model. Our goal is to create an architecture that runs and trains faster than the original architecture.

1. INTRODUCTION

Seismic classification is a relevant problem because there are large databases of unlabeled seismic data. Seismic data is known to be difficult to classify because it has a large number of variables. The paper (Jenkins et., al 2020) classifies seismic data types using an autoencoder, and K-means clustering method. By reducing the dimension of the seismic data using the autoencoder, they were able to produce effective clustering on the latent feature space using K-means. Our aim is to build off the model used in the (Jenkins et., al 2020) paper and modify the network architecture to see how various feature encoding methods perform in clustering seismic data while using the baseline model as a metric. By implementing a U-NET, and GMM clustering method we want to achieve a architecture that runs and trains faster.

We were able to access the Antarctica Ross Ice Shelf [1] seismic data that has been previously processed by the original paper (Jenkins et., al 2020). Having access to prepossessed data will allow us to allocate our efforts toward directly experimenting with models. The input to our modified architecture is spectrogram images from the Antarctic seismic data. We then used T-SNE to visualize the cluster output of the latent feature space from the U-NET architecture and the GMM clustering method.

2. DATASET AND FEATURES

The goal of this project is to go through the massive amount of unlabelled seismic data gathered from seismographs across the Ross Ice shelf and provide context to the data. Seismograph data was recorded from 2014-2017, amassing to a very large amount of data filled with unimportant events with a small minority of the data containing events of interest. This dataset and all preprocessing was accomplished by "Unsupervised Deep Clustering of Seismic Data" by William Jenkins. This project was able to directly use the spectrograms used in that paper.

The seismic data was then scanned with seismological auto-detection algorithms to find the important sections of information. These important sections are recorded as time-series seismic waves. The Jenkins paper then converted this time-series data into the frequency domain via a Fourier transform to capture the frequency information in a spectrogram. This step is very important because it converts time-series data into a single image. By turning the time-series data into an image, this project can take advantage of the extensive research done to prove the effectiveness of CNNs on image data.

The encoded spectrogram images consist of 8700 features (1x87x100). We have a total of 427798 spectrograms that we split into a training set and testing set for the classification task defined below.

3. RELATED WORK

3.1. Clustering Methods

Hierarchical clustering has been used as an alternative to K-means to identify groups in a data set. In one paper [8], Hierarchical clustering is used for earthquake magnitude prediction. By comparing the performance of prediction models with and without clustering, they found that hierarchical clustering led to an improved accuracy in prediction models that clustered based on non-spatial attributes. Unlike the K-means clustering method that was used in the (Jenkins et., al

2020) paper, Hierarchical clustering is not as sensitive to initial seeding or outliers. However, we chose not to pursue this method due to the large computational cost. Gaussian Mixture modeling has also been used as an effective unsupervised clustering method for seismic data. It has been used as a more probabilistic and flexible alternative to K-means clustering. This [4] paper used a deep scattering network for feature extraction, then a Gaussian mixture model for clustering. This paper was significant because it is also an experiment where unsupervised deep learning with clustering is applied to the learned feature space of multi-channel seismic signals. The success the Gaussian Mixture model had in this paper led us to replace the K-means clustering method used in the (Jenkins et., al 2020) paper.

3.2. Dimensionality Reduction

In the past, dimensionality reduction has been used to improve the performance of clustering methods in seismic predictions because there are a large number of variables. One experiment [3] applied Principal Component Analysis for dimensionality reduction on data from active seismic zones in Chile to predict earthquakes. They found that when combined with other classification networks, dimensionality reduction resulted in an noticeably improved performance in terms of average accuracy.

3.3. Deep Embedded Clustering

Deep Embedded Clustering (DEC) is a clustering method that uses a deep learning model to embed data X in a dimension-reduced feature representation Z, and clusters data based off that encoded representation. A unsupervised method DEC method introduced by (Xie et., al 2016), presents the framework our model is based upon. The method consists of two components, **cite**[1] initialization via a training and auto encoder, and optimizing or clustering, where the encoded represent ion is iteratively trained by minimizing the Kullback-Leibler (KL) divergence between the target distribution and computed distribution **cite**[2].
The model proposed by (Xie et., al 2016), is implemented by (Jenkins et., al 2020) to cluster seismic data collected from the Antarctic Ross ice shelf from 2014-2017.

4. METHODS

4.1. Data Processing

In this project we were able to use an H5 dataset file provided by the Jenkins paper. This file type is designed to store large datasets in a zipped format and lazy loaded into memory as the model is trained. We utilized the dataset, samplers, and dataloaders in the PyTorch library to handle random batch processing of the data in our model. We adapted the dataset class used in the Jenkins paper to handle indexing of the H5 file. We wrote our own code to split the dataset into a train, validation, and testing sets using the SubsetRandomSampler objects in order to randomly sample a set percentage of the dataset.

4.2. Autoencoder

Auto Encoders are a common machine learning model used to reconstruct input data. A typical autoencoder consists of two components, a contracting path (encoder) and a expanding path (decoder). The Encoder take the input data with a high number of feature, and passes it through a series of convolutional layers. These convolutional layers reduce the number of features representing the input data consecutively until a desired latent space or encoded representation is reached. The idea is to train the network to learn the a low dimensional feature representation of the input data distribution . The latent image representation is passed in to decoder, where the decoder performs an operation similar to that of the encoder in reverse. The decoder performs a series of up-sampling or transpose convolutional layers to recover the original input data shape. A loss function represents the difference between the original input image and and the reconstructed image and the network is trained to minimize this loss function.

The encoder we use consists of 5 convolutional layers each followed by a ReLU activation function. The activation function is use to introduce non-linearity to the network, which in-turn allows the network to learn non-linear patterns in the data distribution. After the 5th convolutional layer, a linear and flattening layer is used to shape the latent space into a 9 feature representation of the input image

The decoder follows a similar structure the that of the encoder. The decoder takes in the flattened image representation, passes it through a linear layer with a ReLU activation function and reshapes the image to be passed into the expansive path.The expansive path consists of 5 transpose convolutional layers each followed be a ReLU activation except the last, as a means of reconstruction the input image.

4.3. U-Net

Proposed by (Renneberger et., al 2016) for image segmentation, a U-Net works in a similar manner to autoencoder, consisting of and contracting and an expansive path to encode and reconstruct an image[5]. However, unlike an autoencoder, a U-Net contains skip connections which pass image information directly from each contacting layer to the respective expansive layer.

The contracting path consists of 4 sequential operations will term ”reductions” for simplicity. Each reduction consists of 2 sequential convolutions, the first reducing the feature space, the second maintaining the same shape or a ”depth-wise” convolution - both convolutions with a ReLU activation. That image representation is then passed directly to the

respective expansive layer. The convolutions are followed by a 2 x 2 max-pooling layer to reduce the feature space and a dropout layer. The next reduction follows the dropout layer from the previous reduction.

After the 4 reductions, there is a layer we will term the "mid-layer". The mid-layer consists of two sequential convolutions in the same manner seen in the reductions, but does not include a max-pooling or dropout layer. These convolutions are followed by a flattening and linear-ReLU layer which we use this our latent feature space. The network was constructed such that this latent space matches the shape to that of the encoder - 1152 x 9,providing a 9 dimensional feature space.

The expansive path consists of 4 sequential operations we will term "expansions". Each expansion consists of a transpose convolutional layer, a dropout layer, followed by a concatenation between the output and the corresponding skip connection. The concatenation is followed a by two convolutions layers with ReLU activation.

4.4. Clustering via Gaussian Mixture Models

The Gaussian Mixture Models (GMM) is an implementation of the Expectation Maximisation algorithm (EM). These algorithms start by randomly initialising a parameter and then using the resulting classification to estimate a better parameter. Then those new parameters assign new labels to the data. These two step alternate in until convergence.

The GMM algorithm uses N gaussians where N is the number of clusters selected. Each Gaussian has a full covariance matrix and mean such that it is defined across the full feature space. The parameters learned in GMM are the covariance matrix as well as the mean for each of the N gaussians. Bayes's rule is used to estimate the values of each of the parameters from information about the rest of the gaussians. The result is that each data point has a probability it is a member of the other classes, and the highest probability assigns the final class number.

An important note is that these algorithms suffer from a selection of the number of clusters. In our project we used the experimentally selected value of 8 clusters from the Jenkins paper. This value was tuned to the DEC clustering algorithm, so it is possible that a different number of clusters would do better on the latent space we created.

In our project we used the Scikit-learn GMM library to fit the parameters to our dataset and then predict the classes of our test set. In order to train the GMM we evaluated our model on the entire training set and stored all of the latent space feature vectors in memory. We had enough memory that we could train our GMM on the entire latent space feature set without using lazy loading.

The GMM model could only run on the CPU and therefore took longer to run. There are other libraries out there that have started moving these types of algorithms onto the GPU, but for now we focused on getting results. The [CUML](https://docs.rapids.ai/api/cuml/stable/) library is a repo for many similar algorithms such as K-means. We used this library for visualising the results of our clustering via TSNE graphs as well.

5. EXPERIMENTS / RESULTS / DISCUSSION

5.1. Auto encoder Training

In order to cluster the data, a latent space representation of the data must be achieved through a trained autoencoder. The autoencoder is trained to minimize the mean squared error (MSE) between the input (original images) and output (reconstructed images). The hyperparameters were determined experimentally as follows: learning rate=.001, batch size=512, epochs=10. The validation and training loss for the autoencoder can be seen in **Figure 1**, however, one should note this shows training for 100 epochs for visualization purposes only.

5.2. U-Net training

The U-Net is trained in a similar manner the that of the autoencoder, by minimizing MSE between the input and output spectrograms. However, because the U-Net architecture contains skip layers, the U-Net learns a feature representation of the data distribution much quicker in terms of iterations in comparison the autoencoder. Even when training with low learning rates ($<$1E-4) the unet validation loss is typically minimized within 1 epoch and grows afterward. From experimentation, hyperparameters were determined as follows: learning rate=.0001, batch size=512, epochs=1. Trained for 1 epoch, 1 iterations total, the validation and training loss for the U-Net can be seen in **Figure 2**. It is worth noting that the validation is computed after each epoch of training is completed and shows minimal variance throughout iterations.

5.3. Clustering

Using the methods discussed in **4.4** the spectrograms are clustered from the 9-dimensional latent features space for both the autoencoder and U-Net. Using t-SNE, a method proposed by (Maaten and Hinten, 2008), the 9 dimension clustering can be mapped into 2-dimensions for as a means of visualization [7]. The results can be seen in **Figure 3** and **Figure 4**, along with the corresponding cluster assignment distribution. To interpret these results we look for 3 main observations: (1) Localization of clusters, ie. class assignments are found a group or area, (2) separation of clusters, ie. minimal cluster overlap and more space between different clusters, (3) a logical distribution of class assignments.

From a visual comparison between the two results in **Figure 3** and **Figure 4**, there is no clear winner. The autoencoder appears to outperform the U-Net with in terms of cluster localization. While the U-Net outperforms the autoencoder when looking at the separation between clusters.

While clustering is inherently poorly defined, to goal is to present the data with an attribute that can provide some meaning. With our seismic data, one might expect clusters to represent different seismic events, such the size or type the seismic event, or location of the seismic event. Future work should be aimed toward interpreting clusters that gives clarity to better understand these seismic events.

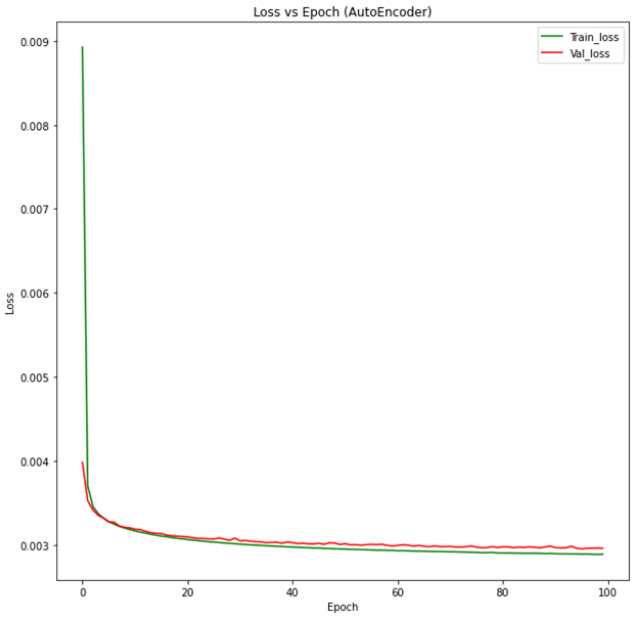

Fig. 1. Auto Encoder Training Loss, Train (Green), Validation (Red)

.

6. CONCLUSION

Through modifying the architecture in the original paper (Jenkins et., al 2021), we were able to compare the impact implementing a U-NET, and Gaussian Mixture Model. The results of this experiment are significant because we are able to see that making modifications to the clustering method and architecture of the original model will create changes in the speed, and accuracy of clustering a latent feature space. By working with the same seismic data set, and T-SNE visualizations as (Jenkins et., al 2021) we are able to compare the outcome of our clustering with that of the original paper. Our results show that making these modifications to the original architecture will have a clear impact on the labels, and separation between the clusters.

The future work of this experiment includes using the modified architecture for seismic labeling. The original model in (Jenkins et., al 2020) was able to label the input data. We implemented the architecture up to learning the latent feature space. While this was sufficient for the objective of our experiment, we would be interested in seeing

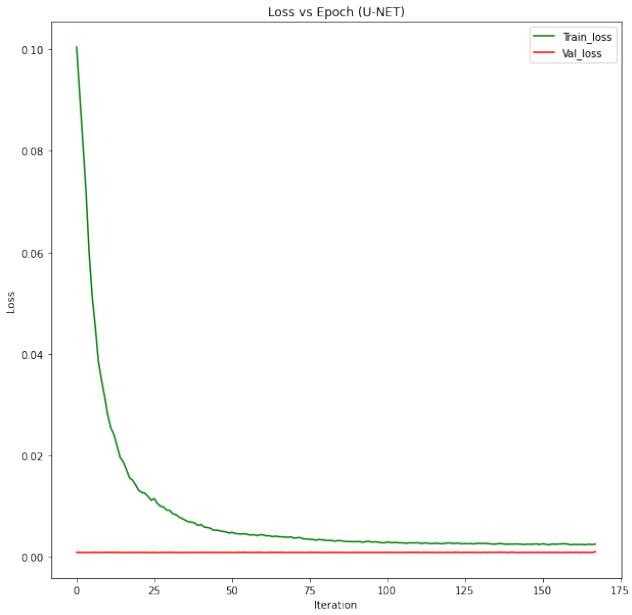

Fig. 2. U-Net Training Loss, Train (Green), Validation (Red)

.

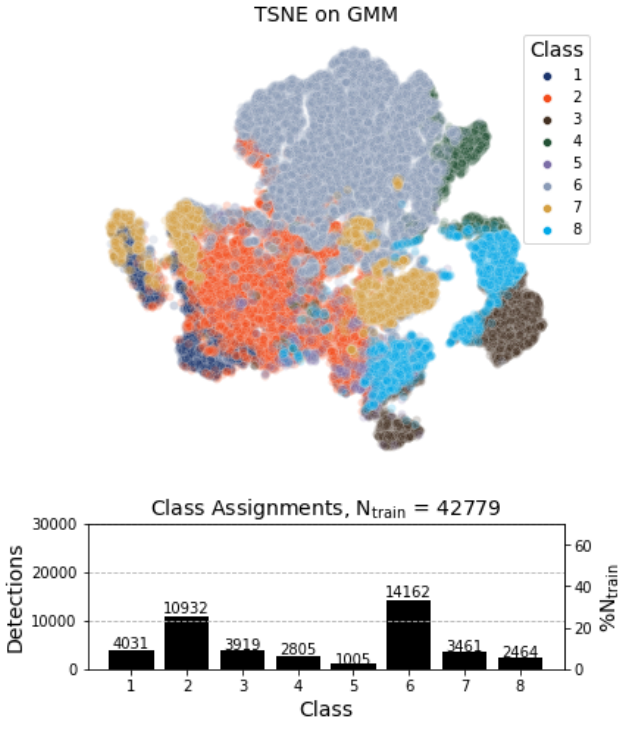

Fig. 3. TSNE Visualization of clustering with AE pre-training

.

the results that applying the U-net and GMM would have on labeling the data set. Code for this project can be found at https://github.com/CameronLewisUCSD/ECE228Project

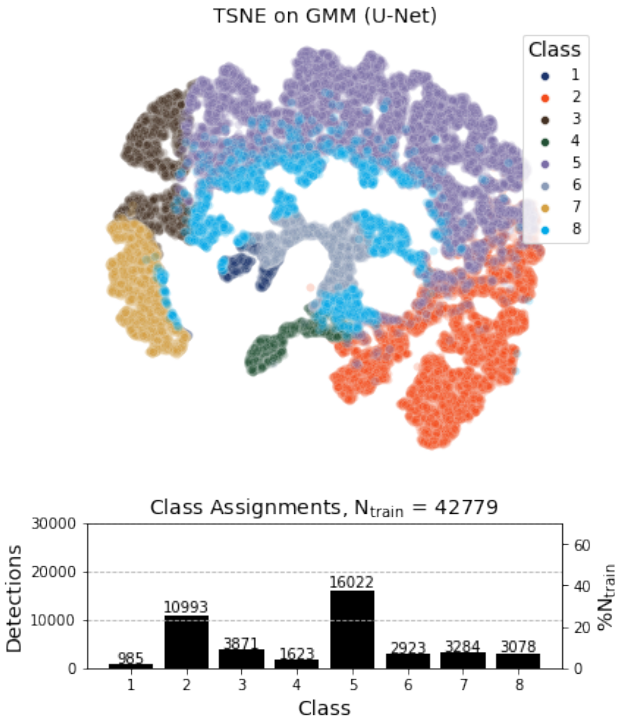

Fig. 4. TSNE Visualization of clustering with U-Net pre-training

.

7. CONTRIBUTIONS

Cameron Lewis focused on data preprocessing, constructing the clustering components for GMM, and data visualisation via TSNE graphs. Heather Huntley did extensive literature surveys, guided the direction of the project with research on feature reduction, and model training techniques as well as contributed towards construction of the model architecture. John O'Boyle constructed the autoencoder and U-net architectures and created the code to train the architectures.

8. REFERENCES

[1] G. Cortés, F. Martínez-Álvarez, A. Morales-Esteban, J. Reyes, and A. Troncoso, "Improving earthquake prediction with principal component analysis: Application to chile," *Lecture Notes in Computer Science*, p. In press, 06 2015.

[2] S. Flores, "Variational autoencoders are beautiful." [Online]. Available: https://www.compthree.com/blog/autoencoder/

[3] W. F. Jenkins, P. Gerstoft, M. Bianco, and P. D. Bromirski, "Unsupervised deep clustering of seismic data: Monitoring the ross ice shelf, antarctica," *Earth and Space Science Open Archive*, p. 40, 2021. [Online]. Available: https://doi.org/10.1002/essoar.10505894.1

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.

[5] C. Savaş, M. Yildiz, S. Eken, C. İkibaş, and A. Sayar, *Clustering Earthquake Data*, 01 2019, pp. 224–239.

[6] L. Seydoux, R. Balestriero, P. Poli, M. d. Hoop, M. Campillo, and R. Baraniuk, "Clustering earthquake signals and background noises in continuous seismic data with unsupervised deep learning," *Nature Communications*, vol. 11, no. 1, 2020.

[7] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008. [Online]. Available: http://jmlr.org/papers/v9/vandermaaten08a.html

[8] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," 2016.

9. RESPONSE TO CRITIQUES

- Q. Also, I am curious on your reference code and code of CNN architecture.
- A. Please review section 4.2 for more info.
- Q. However, the clusters are not that discrete compared to the DEC trained results for unknown reasons. It can be noticed that the Ntrain(=42779) in your results is smaller than that (=100000) in the DEC trained results. It would be interesting to know what the comparison of results would be if the Ntrain are the same for both cases.
- A. The TSNE plot in Fig 2 was shown before. This visualisation is of only 42k data points that compose the test set of our data. We trained on the full training set composed of 300k spectrograms. We believe the TSNE should look similar for the full dataset compared to the randomly sampled testing set.
- Q. Is there any possible solution or strategy in your mind to get better separated clusters than the DEC trained results?
- A. Please see section 4.3 for the U-net architecture with GMM we implemented to try and get more separated clusters compared to our AE with GMM. We beleive it worked well at spearating the clusters.
- Q. The training time for getting a more discrete clustering may be long. Would it be easier or more time-efficient to train a discrete clustering with less data points?

- A. Please see section 4.3 regarding the U-net. We implemented this model to see what would happen with a different architecture without using the full DEC. We found that the U-net architecture produced much more discrete clusters than the traditional auto-encoder with GMM, but it was much faster to train than a full DEC.
- Q. So I would like to suggest that you could explain in more detail about why you decided to choose the '8 clusters' in your deep embedding method?
- A. The Jenkins paper experimentally found 8 clusters to result in the "best" clustering, so we used the same hyper parameter.
- Q.What kind of samples of your dataset will be located in a specific cluster?
- A. We do not know much about seismology or what the underlying data actually represents. We would like to visualise a spectrogram from each cluster, but it doesn't fit in the report.
- Q. I didn't see some formulates or details on how you convert Time-series data to Fourier Transform to Spectrogram.
- A. Please view the section 4.1 for more information on data-preprocessing. This project used the data directly preprocessed into spectrograms by the Jenkins paper, so we did not do any of the FFT ourselves.