A GNN-based Interactive Application for Earthquake Early Warning : Progress Report

Lindsay Y. Chuang*
EAS, Georgia Tech

Meghana Jain
COC, Georgia Tech

George Jeno
COC,Georgia Tech

Preston T. Lee
COC,Georgia Tech

Hsin-Yi Lin
ISYE,Georgia Tech

Paarth J. Parekh
BIOL,Georgia Tech

1 INTRODUCTION

Earthquake early warning (EEW) systems are crucial applications for high seismic risk areas, such as the U.S. west coast, Japan, Mexico, South Italy, and Taiwan [1–5]. An EEW system is designed to detect earthquakes and determine their source characteristics (e.g. location and size) by leveraging the very first few seconds of seismic signals (time-series of ground motions). Once an EEW system detects an earthquake, a timely alert can be sent out to users to take necessary actions and reduce damages. The speed and robustness of an EEW system plays a crucial role in minimizing preventable damage, evacuating citizens, and producing a more efficient emergency response [6, 7] - thus, attempts to improve either quality are well-founded in the context of real-world application.

All EEW systems rely on efficient and robust algorithms which automatically process seismic data in real-time. Most of the core algorithms implemented to-date are simple and easy-to-use regression based predictors [8]. Alternative approaches have involved the use of Bayesian probabilistic modeling to predict seismic event parameter probability given input features, allowing for maximum likelihood parameters to be determined [9, 10]. However, there are concerns about the robustness of point estimate extraction from the generated probability distribution, an issue not observed in classification/regression models.

Recent development of deep learning provides alternative ways for fast automation on earthquake related tasks. Convolutional neural networks (CNNs) are one such method, though they can vary in their implementations and outputs. These networks use seismic signals in order to provide source characteristics. Some variations simply output the coordinates of an earthquake's hypo-center [11], and some extend predictions to more detailed characteristics, such as magnitude, location, depth, and origin time [12, 13]. One interesting application comes in the form of DeepQuake [14], a seismo-acoustic event classification CNN. It argues that seismo-acoustic waves can originate not only from earthquakes, but also from events such as volcanic activity, man-made blasts, and the like. With DeepQuake, these differences can be classified, allowing for filtering of earthquake events and others.

Although CNNs are now the most commonly used type of deep learning model in the seismology field, latest researches argue that graph based neural networks (GNNs) are more suitable for earthquake applications [15, 16]. This is because GNNs can better process non-Euclidean structures, which is an inherited data structure of most of the seismic networks worldwide (i.e. seismic networks distributed irregularly spatially). A recent study on GNN demonstrates that GNN can outperform CNN models on automated seismic source characterization task even when only part of the data-set were used [16]. This research opens up a door for potential usage of GNN on EEW system where decisions are made based on incomplete data, though this concept is yet to be proved.

2 OBJECTIVES

This project aims to investigate the ability of the GNN model proposed on [16] with a special emphasis of the potential usage for EEW. The study areas this project focus on are high seismic risks areas including Southern California, Oklahoma, Pacific Northwest, and Vancouver Island. For each of the areas, we train an area-specific GNN model and then probe them by various datasets. We purposely make the datasets incomplete in order to mimic real-world situation. In order to evaluate the accuracy of the predictions, we compare the predictions made by GNN modles with the official earthquake catalogs announced by the United States Geological Survey (USGS). In order to gain more insights from the experiments, we first organized the outputs of the experiments into a NOSQL database, and then visualize them using a dashboard style interactive web app. If the project is successful, this experiment can help with building the next generation of EEW systems. An EEW system with higher efficiency and accuracy can significantly help reduce earthquake hazards in high risk areas. If not successful, the experiment can still shed light on the limitations of GNN. The data structure and the visualizations we will build will still be valuable for other graph based applications. Detains of methods, experiment designs, and progress we made please see section 3.

3 APPROACH AND PROGRESS

The project is constructed by 4 key building blocks as previously described in the proposal. The tools we use in each of the building blocks are summarized in table 1. Progress we have made for each of the building blocks are described in the following subsections.

Building Blocks	Tools
(1) Data scraping	Obspy Client API [17]
(2) GNN	Tensorflow GNN model [16]
(3) Database	MongoDB Mongoengine
(4) Visualization	Leaflet | DC.js | Flask

Table 1: Library used for each building block

3.1 Data and data pre-processing

We used the Obspy Client API to download data from the IRIS datacenter. The inputs for the API are (1) a bounding box for the study area, (2) seismic network code (e.g. "CI" for the Southern California network), and (3) the time range of the data. Once the request is sent to the IRIS data center via the API, the data center will query their database and return the data based on the request. The study areas we requested data for are Southern California, Oklahoma, the Pacific North West, and Vancouver Island. After the downloads were complete, we compiled the earthquake catalogs and station catalogs into json and csv files and then loaded them into a MongoDB database with the schema described in section 3.3. Waveform data are first band-pass filtered between 1.0 to 8.0 Hz, truncated to 101 seconds starting from earthquake origin time, grouped together based on events, and then saved as numpy array files. Table 2 summarizes the time range, the number of events, and the number of stations we downloaded for each dataset.

Region	Time period	Number of stations	Number of events
California	2000/01/01-2014/12/31	187	1386
Pacific NW	2000/01/01-2014/12/31	49	664
Oklahoma	2015/01/01-2020/12/30	2	38
Vancouver Island	2000/01/01-2020/01/01	29	1614

Table 2: Datasets we downloaded for this project

3.2 GNN model and experiments

We use the GNN model developed in [16] as a bedrock to perform our experiments. The abstract problem solved by GNNs can be expressed as the regression of "graph attributes" given node-level local features. We apply this abstract framework to the EEW problem by considering the seismic network nodes to be the graph nodes, with the node-level features being the measured waveforms for a seismic event (this underlying graph differs for each seismic event). The graph attributes to be regressed are the event latitude, longitude, depth, and magnitude.

We then can break the baseline model down into three parts: seismic waveform feature extraction, node feature vector construction, and graph attribute prediction. Waveform feature extraction is performed prior to GNN application; it allows for the large amounts of data contained in the waveforms to be concisely represented as feature vectors, reducing future computational load and preparing for the GNN steps of the model. This initial feature extraction is performed through the use of several deep convolutional layers with one-dimensional filters, stacked to produce a CNN feature extractor. It takes in three bands of waveforms for each of the stations with valid measurements for the seismic event, and outputs a 64-dimensional feature vector representing the measured data of each station. Note that strictly adhering to one-dimensional convolutions prevents data sharing between nodes, preserving the node structure of the underlying graph.

In the node feature vector construction step, we append the respective station position (normalized latitude, longitude) to each feature vector from the previous step. We then pass these new vectors through a shallow one-dimensional CNN to produce 128-dimensional feature vectors for each node (again preserving node structure). Finally, we perform the key aggregation step by taking the row-wise maximum across all node feature vectors, resulting in an aggregated graph feature vector. This is fed into a shallow fully-connected neural network to produce a four-dimensional graph attribute vector, containing the predicted latitude, longitude, depth, and magnitude of the event.

We trained models for each of the areas separately. The model training was done utilizing varied hyperparameters (e.g. learning rate, number of stations) that were observed to improve results during the evaluation stage. After the models are trained, we created 10 different subsets for each of the events and evaluated the models on the subsets. The subsets are made by randomly discarding 10 % to 40 % of the stations. We evaluate each of the 10 subsets for 20 times, and take the average and standard deviation values as the representative predictions. Note that we maintain 15% random dropout rate (i.e. randomly turn off weights in the models) during evaluation, so the outputs for each of the 20 evaluations are slightly different and can be used to measure uncertainties. Once the predictions are made, we estimate the errors by taking the difference between predictions and the ground truths. The average errors for location, depth, and magnitudes for each of the study areas are shown in table 3.

We are also currently testing several new model architectures that we hypothesize to perform better than the baseline model explained prior, drawing from other work in deep learning. We hope to empirically show increased performance on out-of-sample event characteristic prediction, thereby demonstrating significant improvements to baseline prior work.

3.3 Data structure and database

We used the NOSQL (Not Only SQL) database - MongoDB for data storage. NoSQL databases typically have very flexible schemes. A flexible schema will allow us to easily make changes to the database as the requirements change. Given the high volume and the variety of our data, the database must be capable enough to cope with the flexibility during operation. MongoDB, as one instance of the NoSQL database, can accommodate a variety of information with different structures into one single geodatabase. Non-relational databases have been widely used in geospatial analysis[18, 19] and we think it is suitable for our application. So far, we have built 3 databases. One is to store the experiment results about each earthquake event, which includes information like the latitude, longitude, depth, and magnitude for each prediction. Another is a station lookup table, which contains the latitude and longitude for each station. The last is a ground truth lookup table, which stores ground truth earthquake IDs, we can use this dataframe to access the particular earthquake's predictions. Since the experiments are still running, we will add more experiment results to the database in the future. We used Mongo Compass to build the database. Mongo Compass is the GUI for MongoDB. We can run the queries in the terminal to insert, delete, modify or extract the data. We will use Python package Flask for building a web server that interacts with MongoDB and serving HTML pages.

Region	Location Error (km)	Depth Error (km)	Magnitude Error (Richter)
California	25.24 +/- 22.61	3.38 +/- 2.93	0.15 +/- 0.16
Pacific Northwest	68.66 +/- 99.97	3.51 +/- 6.03	0.14 +/- 0.15
Oklahoma	40.88 +/- 33.12	1.57 +/- 1.47	0.22 +/- 0.19
Vancouver Island	54.77 +/- 51.55	1.59 +/- 4.95	0.22 +/- 0.22

Table 3: Experiment Results (highest average error observed to lowest average error observed)

3.4 Interactive graph visualization

We plan to develop a dashboard style interactive visualization framework with 6 interactive elements to display the experiment results. The libraries we will use are Flask, DC.js, Leaflet, and MongoDB. The design concept and the technical perspectives are based on these two tutorials [20, 21]. Flask is a python based web framework that equips many libraries for easy web development [22]. Leaflet is a tool with special focus on maps [23]. It has been used in visualizing geographical data [24], and therefore it could help us with building interactive maps with integration of other features. DC.js is a JavaScript charting library with native cross-filter support (https://dc-js.github.io/dc.js/). Figure 1 illustrates the layout of the dashboard. Details of each of the 6 elements are described below:

- **Element 01**: A map highlighting the main regions on which we will plot the ground truth data. The map will be interactive, helping the user to filter out the data, for visualising the next plots in element 2 and 6 of the dashboard. Each region is defined by a bounding box of its min/max latitudes and longitudes.
- **Element 02**: A list of networks that the user can filter the results on the map with.
- **Element 03**: A time range slider that the user can manipulate to see active stations and events in the selected time range.
- **Element 04**: A text box that shows the number of events visible on the map currently.
- **Element 05**: A second location map, displaying the locations of ten predictions associated with one ground truth event that were filtered from the first map. It will further show the stations associated with each of the predictions.
- **Element 06**: A collection of bar charts that show us the values obtained for the location, magnitude, and depth, for each predictions, along with the ground truth value for each (different colored bar) to allow for comparison, and error calculation for each of the ground truth event.

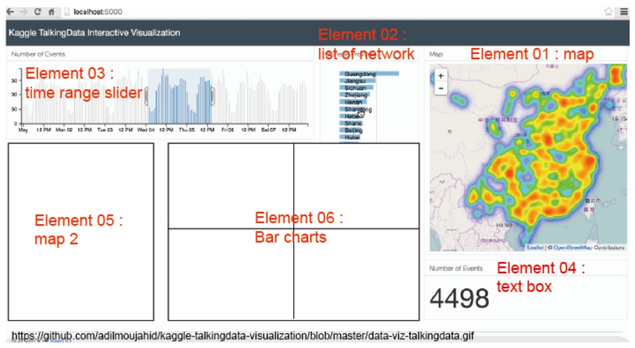

Figure 1: Example dashboard layout showing the layout of 6 elements. Figure modified after the page in [21].

4 LIST OF INNOVATIONS

(1) We have trained new GNN models for high seismic risk areas where this type of model did not exist to the best of our knowledge.
(2) We plan to further improve the original GNN model by adjusting the model architecture. It will make contributions to the field of seismology if the new architecture shows significant improvements on results.
(3) We use MongoDB database to store experiment data and incorporate it to a dashboard style interactive web application. This practice is not commonly seen in seismological research, where simple .csv files for storage and statics plots are predominant.

5 MEMBER CONTRIBUTION AND PLAN OF ACTIVITIES

All members have participated in several group discussions, helped identified various tools to use, and written this progress report. Miss Chuang, Miss Jain, and Mr. Jeno have designed and conducted various experiments. They are working to improve the baseline model by tweaking hyper-parameters and trying out new architectures. Miss Lin has constructed three databases using MongoDB. She will write codes to deploy the front-end and back-end data queries in the visualization app. Miss Chuang, Mr. Lee and Mr. Parekh have focused on identifying the tools, related tutorials and strategies for building the visualization app. All group members will contribute to building the visualization app by writing up codes for different elements in the app.

REFERENCES

[1] R. M. Given, Douglas D.and Allen, A. S. Baltay, P. Bodin, E. S. Cochran, K. Creager, R. M. de Groot, L. S. Gee, E. Hauksson, T. H. Heaton, M. Hellweg, J. R. Murray, V. I. Thomas, and D. Toomey, "Revised technical implementation plan for the ShakeAlert system—An earthquake early warning system for the West Coast of the United States," 2018.

[2] K. Doi, "Early Warning Systems for Natural Disaster Reduction," vol. 81, ch. 5.6, Springer, 2003.

[3] J. E. Aranda, A. Jimenez, G. Ibarrola, F. Alcantar, A. Aguilar, M. Inostroza, and S. Maldonado, "Mexico city seismic alert system," *Seismological Research Letters*, vol. 66, no. 6, pp. 42–53, 1995.

[4] A. Zollo, G. Iannaccone, V. Convertito, L. Elia, I. Iervolino, M. Lancieri, A. Lomax, C. Martino, C. Satriano, E. Weber, and P. Gasparini, *Earthquake Early Warning System in Southern Italy*, pp. 175–201. New York, NY: Springer New York, 2011.

[5] Y.-M. Wu, H. Mittal, T.-C. Huang, B. M. Yang, J.-C. Jan, and S. K. Chen, "Performance of a low-cost earthquake early warning system (p-alert) and shake map production during the 2018 mw 6.4 hualien, taiwan, earthquake," *Seismological Research Letters*, vol. 90, no. 1, pp. 19–29, 2019.

[6] P. Gasparini, G. Manfredi, and J. Zschau, "Earthquake early warning as a tool for improving society's resilience and crisis response," *Soil Dynamics and Earthquake Engineering*, vol. 31, no. 2, pp. 267–270, 2011. Prospects and applications of earthquake early warning, real-time risk management, rapid response and loss mitigation.

[7] P. UNISDR, "Global survey of early warning systems: An assessment of capacities, gaps and opportunities toward building a comprehensive global early warning system for all natural hazards," *Platf Promot Early Warn UNISDR—PPEW UN*, p. 2006, 2006.

[8] C. Satriano, Y.-M. Wu, A. Zollo, and H. Kanamori, "Earthquake early warning: Concepts, methods and physical grounds," *Soil Dynamics and Earthquake Engineering*, vol. 31, no. 2, pp. 106–118, 2011.

[9] S. Wu, M. Yamada, K. Tamaribuchi, and J. Beck, "Multi-events earthquake early warning algorithm using a bayesian approach," *Geophysical Journal International*, vol. 200, no. 2, pp. 791–808, 2015.

[10] G. Cua, M. Fischer, T. Heaton, and S. Wiemer, "Real-time performance of the virtual seismologist earthquake early warning algorithm in southern california," *Seismological Research Letters*, vol. 80, no. 5, pp. 740–747, 2009.

[11] M. Kriegerowski, G. M. Petersen, H. Vasyura-Bathke, and M. Ohrnberger, "A deep convolutional neural network for localization of clustered earthquakes based on multistation full waveforms," *Seismological Research Letters*, vol. 90, pp. 510–516, 2018.

[12] O. M. Saad, A. G. Hafez, and S. M. S., "Deep Learning Approach for Earthquake Parameters Classification in Earthquake Early Warning System," *IEEE Geoscience and Remote Sensing Letters*, pp. 1–5, 2018.

[13] A. Lomax, A. Michelini, and D. Jozinović, "An investigation of rapid earthquake characterization using single-station waveforms and a convolutional neural network," *Seismological Research Letters*, vol. 90, pp. 517–529, 2019.

[14] L. Trani, G. A. Pagani, J. P. Pereira Zanetti, C. Chapeland, and E. L. G., "Deepquake – an application of cnn for seismo-acoustic event classification in the netherlands," 2020.

[15] K. Yano, T. Shiina, S. Kurata, A. Kato, F. Komaki, S. Sakai, and N. Hirata, "Graph-partitioning based convolutional neural network for earthquake detection using a seismic array," 2020.

[16] M. P. van den Ende and J.-P. Ampuero, "Automated seismic source characterization using deep graph neural networks," *Geophysical Research Letters*, vol. 47, no. 17, p. e2020GL088690, 2020.

[17] M. Beyreuther, R. Barsch, L. Krischer, T. Megies, Y. Behr, and J. Wassermann, "ObsPy: A Python Toolbox for Seismology," *Seismological Research Letters*, vol. 81, pp. 530–533, 05 2010.

[18] W. Tampubolon, "Utilization of nosql database for disaster preparedness," 2014.

[19] . H. Q. Cao, K., "Real-time earthquake monitoring with spatio-temporal fields. isprs annals of the photogrammetry," *Annals of GIS*, vol. 24, pp. 125–138, 2018.

[20] A. Moujahid, "Interactive data visualization with d3.js, dc.js, python, and mongodb."

[21] A. Moujahid, "Interactive data visualization of geospatial data using d3.js, dc.js, leaflet.js and python."

[22] M. Grinberg, *Flask web development: developing web applications with python.* " O'Reilly Media, Inc.", 2018.

[23] D. Edler and M. Vetter, "The simplicity of modern audiovisual web cartography: an example with the open-source javascript library leaflet. js," *KN-Journal of Cartography and Geographic Information*, vol. 69, no. 1, pp. 51–62, 2019.

[24] A. Chourasia, K. Richards-Dinger, J. Dieterich, and Y. Cui, "Visual exploration and analysis of time series earthquake data," in *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact*, pp. 1–6, 2017.