A GNN-based Interactive Application for Earthquake Early Warning

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 ABSTRACT

The use of an earthquake early warning (EEW) system has clear benefits in emergency responses. Such a system that ensures the safety of nearby and affected populations relies on fast and accurate predictions. A good EEW system should also be easily accessible and the content it displays should be easy to understand. In this project we explored the possibility of applying a graph neural network (GNN) based method in an EEW system. We started with training new GNN models in several high seismic risk areas in North America. We then conducted experiments in those areas to mimic real-time situation and see how well the GNN models perform. The quality of predictions were quantified by comparing them with the official earthquake catalog. An interactive dashboard that we built enabled us to investigate our results in fine detail. With this good visualization tool, we were able to discover new patterns that were not mentioned in previous studies. Although more studies need to be done in order to improve the performance of GNN models, we believe our project demonstrates the GNN model approach is beneficial for timely detection, therefore has potential for EEW application.

2 INTRODUCTION AND MOTIVATION

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.

3 OBJECTIVES

This project aimed to investigate the ability of the GNN model proposed in [16] with a special emphasis of the potential usage for EEW. The study areas this project focused on were high seismic risks areas including Southern California, Oklahoma, Pacific Northwest, and Vancouver Island. For each of the areas, we trained an area-specific GNN model and then probed them by various datasets. We purposely made the datasets incomplete in order to mimic real-world situations. In order to evaluate the accuracy of the predictions, we compared the predictions made by GNN models 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 visualized them using a dashboard style interactive web app. We hope this experiment can help with building the next generation of EEW systems, and the data structures and the visualizations we build can help improving the common data processing pipeline in the seismology community.

4 APPROACH

The project is constructed by 4 key building blocks. The tools we use in each of the building blocks are summarized in table 1. Details for each of the building blocks are given in the following subsections.

4.1 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) the 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 (fig. 1). 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 5.4. 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.

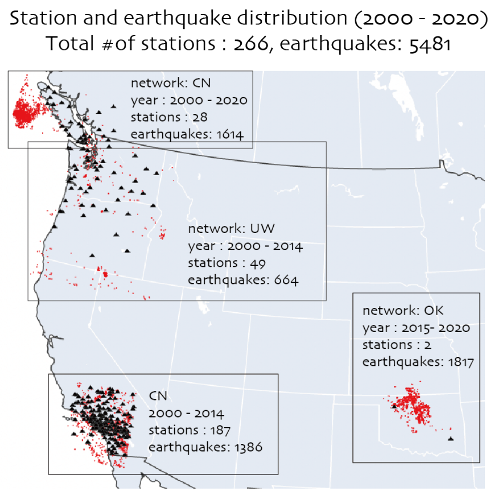


Figure 1: Earthquake and station distribution in each of the 4 study areas. Red dots are earthquake source locations and black triangles are seismic stations.

Table 2 summarizes the time range, the number of events, and the number of stations we downloaded for each region. Figure 1 shows the event and station distribution. The total number of stations and earthquakes used are 266 and 5481, respectively. The size of the whole data set is approximately 5 GB.

4.2 GNN model and model training

We used the GNN model developed in [16] as a bedrock to perform our experiments. The abstract

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

Region (network code)	Time period	Number of stations	Number of events
California (CI)	2000/01/01-2014/12/31	187	1386
Pacific NW (UW)	2000/01/01-2014/12/31	49	664
Oklahoma (OK)	2015/01/01-2020/12/30	2	1817
Vancouver Island (CN)	2000/01/01-2020/01/01	28	1614

Table 2: Datasets we downloaded for this project

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 it 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 and 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, number of epochs) that were observed to improve results during the evaluation stage. Figure 2 shows the training and validation loss for each of the study areas. Note we did not retrain a model for Southern California (network CI) as the model was already optimized in the original paper.

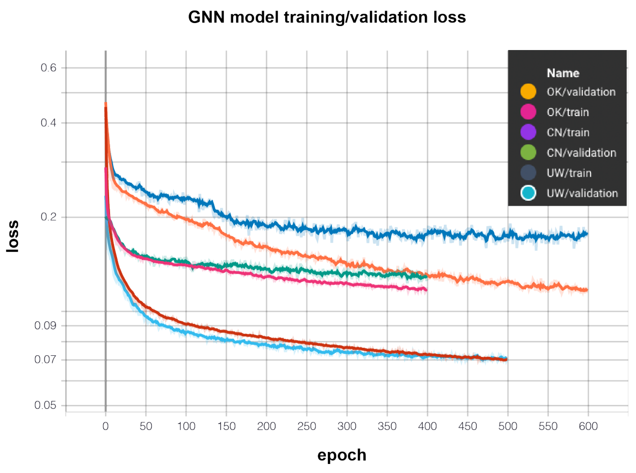


Figure 2: Training and validation loss for three of the study areas.

4.3 Experiment design

Because graph neural networks are agnostic to the underlying graph cardinality, different levels of regressive ability are displayed for node-level variable input. To perform a deeper analysis on the efficacy of the trained GNN models in the absence of full data (when only a subset of station data is made available), we conducted a series of experiments in which data from specific stations was discarded prior to model performance evaluation. Specifically, after the models are trained, we created 10 different station 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 section 5.1, table 3.

In addition, we produced and evaluated several hypotheses regarding an increase in model performance due to various architectural changes. Because the focus was to examine the efficacy of graph neural networks in this practical context, no architectural changes were made to the node feature vector construction and graph attribute prediction components of the model. As stated prior, the waveform feature extraction step was performed using a standard CNN architecture; as a result, we identified potential for architectural improvements to be made in this component, drawing from related work in time series analysis and image processing DNN models.

The first series of feature extraction models tested involved the implementation and usage of residual layers [18], famously used in ResNet architectures to produce arbitrarily deep networks for image classification and object detection tasks without encountering the performance issues associated with arbitrary depth. The high-level theoretical backing is that the skip-connections included in residual layers allow for the model to have the ability to ignore certain layers during training, settling on an ideal depth. A variety of depths and channel sizes were tested (partially drawing from [19]); however, none of the new architectures displayed the desired increases in performance. Notably, evaluation performance tended to decrease as the number of parameters increased - this leads us to believe that the original model is critically parameterized for the attribute regression problem.

The second set of feature extraction models tested utilized the concept of recurrent neural networks (RNNs) to produce a feature vector from a time series of arbitrary length. Specifically, an LSTM was trained to produce the node feature vectors from each station time series. The intuition behind utilizing a pure LSTM implementation for this step was the ability to produce these node feature vectors from time series of arbitrary length - the lack of the fixed-length restriction would allow for more flexibility during real-time EEW application. However, this pure implementation was not sufficient for outperforming the original model. Going further, a CNN-LSTM hybrid model was tested (CNN produces feature vector time series, which is then passed into LSTM) - this model produced marginal improvements on the California dataset (20.10 ± 17.09 km Location Error, 2.78 ± 2.44 km Depth Error, 0.12 ± 0.15 Magnitude Error (Richter)), but was otherwise unnoteworthy.

Sparse grid search was used to perform all hyperparameter tuning. While no improvements were found during this research phase, there is clear potential for successful improvements in this area.

4.4 Data structure and database

We used the NOSQL (Not Only SQL) database - MongoDB for data storage. NoSQL databases typically have very flexible schemes. 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[20, 21] and we think it is suitable for our application. We have built 3 collections. The "Experiments" collection is to store the experiment results about each earthquake event, which includes information like the latitude, longitude, depth, magnitude, and corresponding error for each prediction. The second is a 'station lookup' collection, which contains the latitude and longitude for 266 stations. The last is a 'ground truth lookup' collection, which stores every earthquake events' information, and we can use this collection to access the particular earthquake's details. The total document size of the database is 24.73MB. We integrated the MongoDB database with Flask-PyMongo to build our web application. PyMongo connects to a MongoDB server running on port 27017 on localhost, which allows us to use the connection to access the data we need.

Earthquake id	Experiment id	Stations	Event latitude/ latitude error/ latitude standard	Event longitude/ longitude error/ Longitude standard deviation	Event magnitude/ magnitude error/ Magnitude standard deviation
String	String	Array	Double	Double	Double

Earthquake id	Stations	Event latitude	Event longitude	Event magnitude	Event time	Network
String	Array	Double	Double	Double	Date	String

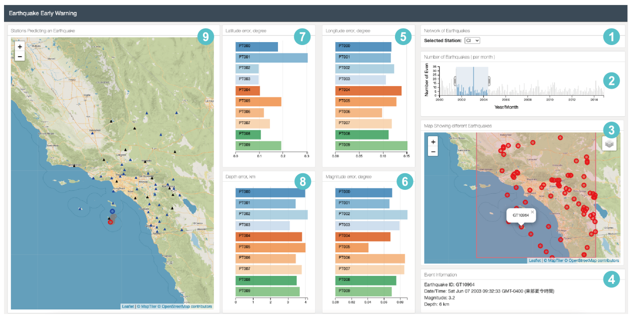
Station id	Station name	Station latitude	Station longitude	Network
Int	String	Double	Double	String

Figure 3: Database schema and datatype for each column

4.5 Interactive graph visualization

We built a dashboard-style interactive visualization framework with 9 interactive elements to display the experiment results. The main libraries we use here are Flask, DC.js, D3.js, Leaflet, and MongoDB. The design concept and technical perspectives were based on these two tutorials [22, 23]. Flask is a python-based web framework that equips many libraries for easy web development [24]. Leaflet is a tool with special focus on maps [25]. It has been used in visualizing geographical data [26], and therefore helped us in building interactive maps by integrating well with the other JavaScript libraries. DC.js is a JavaScript charting library with native cross-filter support, helping us in building interactive maps and integrating well with Leaflet.(https://dc-js.github.io/dc.js/). Figure 4 illustrates the layout of the dashboard. Details of each of the 9 elements are described below:

- **Element 1**: A drop-down list of networks that the user can filter the results on the map with.
- **Element 2**: A time range slider that the user can manipulate to see active events in the selected time range in element 3.
- **Element 3**: A map highlighting the main regions on which we will plot the ground truth data. The map is interactive, allowing the user to filter out the data and visualize it in elements 5-9 of the dashboard. Each region is defined by a bounding box of its min/max latitudes and longitudes.
- **Element 4**: A text box that shows the ground truth event ID, its date and time, magnitude, and depth.
- **Element 5 - 8**: Bar charts that show us the longitude, magnitude, latitude, and depth errors obtained for each of the ten predictions.
- **Element 9**: A second location map, displaying the locations of ten predictions associated with one ground truth event that were filtered from the first map (element 3). It will further show the stations associated with each of the predictions.



Earthquake ID: GT10964
Date/Time: Sat Jun 07 2003 09:32:33 GMT-0400 (東部夏令時間)
Magnitude: 3.2
Depth: 6 km

Figure 4: Interactive dashboard for experiment result visualization. The index of each elements are marked by green bubbles.

5 RESULTS AND DISCUSSION

Throughout our experiments, we observed that the average location errors are largest in Pacific Northwest, which we suspect is due to the training set being too small. Another possibility is that the larger study area could cause the normalized values of station/event coordinates to becoming too small and therefore increase the difficulty for training. We also observed that the magnitude errors of all regions were not too large. The reasoning for this observation could be the already-tight range for magnitudes in our data, between 3 and 6 on the Richter scale, for all events regardless of region. Additionally, the location errors in Oklahoma and Vancouver Island fall into similar range, with Oklahoma being smaller. This is surprising because there are only two stations available in Oklahoma but 28 stations in Vancouver Island. This may be due to the relative station-event distribution where most of the events in Vancouver Island located outside of the network but the earthquakes in Oklahoma are practically located within the network. However, the model trained in Oklahoma still performed much worse than the California model, location prediction-wise. This can be expected, as recording based on fewer stations are less diversified and can lead to some more uncertainty to its origin. The average errors for different areas is summarized in table 3.

After using the dashboard to investigate our results in detail, we observed that the stations closest to the ground truth locations play a key role in predicting correct earthquake locations. In the previous GNN study [16], only the number of stations used were mentioned and interpreted to be the main controlling factor. Figure 4 shows an example of such case, where the experiment 9 (location shown in blue circle on the left map) is missing the closest station to the ground truth (red circle on the left map). Even though this experiment used more stations than other ones, the predicted location is still the most deviated from the ground truth location.

Region (network code)	Location Error (km)	Depth Error (km)	Magnitude Error (Richter)
California (CI)	25.24 +/- 22.61	3.38 +/- 2.93	0.15 +/- 0.16
Pacific Northwest (UW)	68.66 +/- 99.97	3.51 +/- 6.03	0.14 +/- 0.15
Oklahoma (OK)	40.88 +/- 33.12	1.57 +/- 1.47	0.22 +/- 0.19
Vancouver Island (CN)	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)

6 INNOVATIONS

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. Due to the ability for GNN models to intrinsically handle the non-Euclidean structured station data, we projected that its application would lead to performance gains and feasible real-time usage.

We also examined several architectural changes to the feature extraction step of the original GNN model. Advances in this component would play a key role in improving overall regressive ability, and while our initial testing phase did not produce the intended performance gains, we expect further research to lead to innovations in this area.

We used a 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, used for storage and static plots, are predominant.

7 CONCLUSION

In this project, we explored the possibility of using a GNN based method for earthquake early warning systems, applied in high-seismic risk areas such as Southern California, Pacific NW, Vancouver Island, and Oklahoma. After organizing experiment results using MongoDB, we populate that data into an interactive dashboard that visualizes the deviations between the predicted attributes and the ground truth. Should a dashboard like this be deployed in real-life, utilizing the GNN model would be suitable, though in its current state, its predictions uncertainties are high and model improvement is needed.

8 MEMBER CONTRIBUTIONS

All members have participated in several group discussions, helped identified various tools to use, and written all the reports as well as construct the poster. Miss Chuang, Miss Jain, and Mr. Jeno have designed and conducted various experiments utilizing the GNN models. Mr. Jeno has worked to improve the baseline model by tweaking hyper-parameters and trying out new architectures. Miss Lin has constructed three databases using MongoDB. She also wrote code for the front-end and back-end data queries in the visualization app. Mr. Lee and Mr. Parekh surveyed the tutorials and tools to use for visualization. All members have contributed to building the dashboard.

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] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[19] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

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

[21] . 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.

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

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

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

[25] 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.

[26] 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.