SEISMIC FAULT IDENTIFICATION USING GRAPH HIGH-FREQUENCY COMPONENTS AS INPUT TO GRAPH CONVOLUTIONAL NETWORK

Patitapaban Palo, Aurobinda Routray

Department of Electrical Engineering, IIT Kharagpur, India

ABSTRACT

Many activities such as drilling and exploration in the oil and gas industries rely on identifying seismic faults. Using graph high-frequency components as inputs to a graph convolutional network, we propose a method for detecting faults in seismic data. In Graph Signal Processing (GSP), digital signal processing (DSP) concepts are mapped to define the processing techniques for signals on graphs. As a first step, we extract patches of the seismic data centered around the points of concern. Each patch is then represented in a graph domain, with the seismic amplitudes as the graph signals. We attenuate the low-frequency components of the signal with the aid of a graph high-pass filter. By applying the graph Fourier transform, we obtain the graph high-frequency components. These graph high-frequency components act as inputs to a graph convolutional network (GCN). By classifying the patches using GCN, we identify the faults in data.

Index Terms— Graph convolutional network, graph signal processing, high pass filter, seismic faults, total variations.

1. INTRODUCTION

A fault in seismic data is one kind of fracture or discontinuity in the volume of rocks that can appear due to various reasons such as earthquakes, the movement of tectonic plates, or the pressure built up over many years due to friction and rigidity of constituent rocks. The structure of seismic data is highly nonlinear and nonuniform, because of which often, an expert's interpretation is needed to identify the faults. Locating faults manually, however, is a complex and often inaccurate process. Hence, it is vital to automate the fault identification process to reduce human dependencies. In addition, when a fault is present in seismic data, there is a high likelihood of a hydrocarbon trap being present. So appropriate information about faults is necessary for identifying reservoirs.

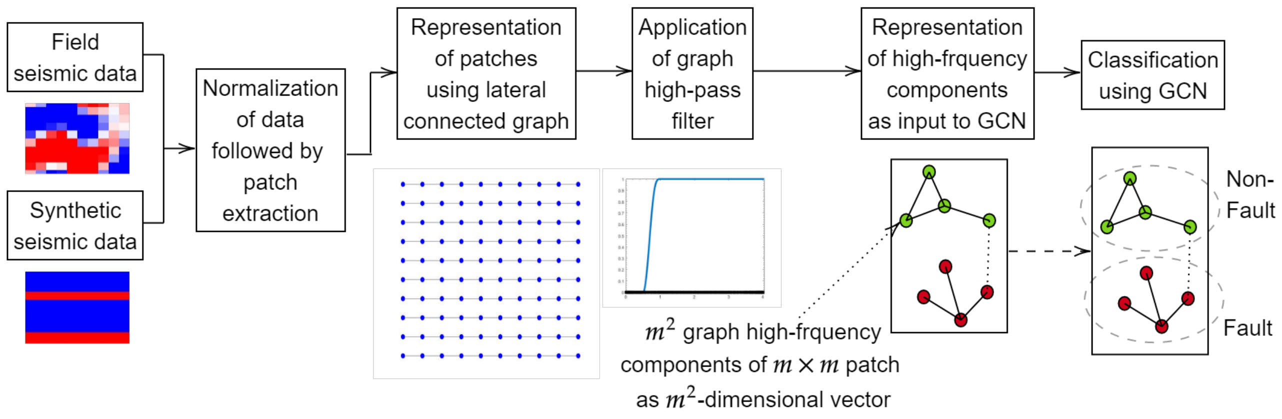
Images, graphs, analysis tools, and several other methods can be used to present and visualize data. When data are irregular and complex in structure, they are more challenging to represent and analyze, and in such cases, a graph helps as an essential tool. Signal processing techniques such as Fourier transform, filters, and frequency response are applied to data residing on graphs in graph signal processing (GSP). In [1], Laplacian-based signal analysis is presented based on spectral graph theory. Then, it is developed on manifold discovery in [2]. An alternative approach for GSP utilizes the adjacency matrix [3, 4] instead of Laplacian based approaches. The concept of graph Fourier transform (GFT) is developed using the classical Fourier transform [5], and authors in [6] build the notions for graph shift operator. Additionally, the definitions of the total variation, frequency ordering, and low-pass, high-pass, and band-pass graph filters for graph signals are presented in [5].

Several methodologies have recently been developed in modern machine learning on graphs that draw features from structured graph data. The graph neural network (GNN) is a basic application of deep learning methods in the graph domain [7]. Furthermore, the graph convolutional network (GCN) generalizes convolutional neural network (CNN) to graphs. In [8], the authors introduce a graph convolutional propagation algorithm to do graph-level classification. However, [9] implements the same method using a single weight matrix per layer. With fast localized convolutions, spectral graph convolutions are commonly used methods to implement GCN [10]. A first-order spectral filter is learned by GCN first before the layers are stacked and activated. GCN has found its application in domains such as NLP [11], citation networks [9], social networks [12] among others.

The Hough transform method [13] is one of the commonly used approaches to automate seismic fault identification. The Hough transform method can also be applied after highlighting likely fault points [14] or after enhancing the data using the dip attribute [15]. In addition, CNN-based approaches are the most popular method for detecting seismic faults. Seismic amplitudes are used as input in some CNN-based methods such as FaultNet3D [16], transfer learning [17], and patch classification [18]. Another way to use a neural network is to use seismic attributes as input [19, 20].

Following normalization, we extract 2D patches centered around a single seismic data point (pixel). Then, the patches are represented in the graph domain by viewing each data point as a node and the seismic amplitudes as graph signals. When the signal varies smoothly across the graph, its energy will be concentrated in low frequencies. But, a seis-

We are grateful to the Oil and Natural Gas Corporation (ONGC), India, for supporting the work and providing real seismic data used in this paper.



Field seismic data
Synthetic seismic data
Normalization of data followed by patch extraction
Representation of patches using lateral connected graph
Application of graph high-pass filter
Representation of high-frquency components as input to GCN
Classification using GCN
m^2 graph high-frquency components of $m \times m$ patch as m^2-dimensional vector
Non-Fault
Fault

Fig. 1. Complete block diagram of the proposed methodology.

mic fault will create a significant difference in signal variation, causing graph high-frequency components. The graph high-frequency components are obtained using a graph high-pass filter and GFT. Next, we train the GCN using these high-frequency components. We use both synthetic data and real data for training the GCN. Our main contribution is identifying the seismic fault by classifying the extracted patches using graph high-frequency components as input to GCN. Fig. 1 shows the block diagram of our work.

2. THEORY

GSP analysis is performed on data whose elements are connected through some relationship property. A graph is expressed as $G = (V, \mathbf{A})$, where $V = \{v_0, ..., v_{N-1}\}$ is a set of N nodes, and $\mathbf{A}$ is the $N \times N$ adjacency matrix. A weight $A_{i,j} \in \mathbb{C}$, with $\mathbb{C}$ being the set of complex numbers, indicates if an edge exists between nodes i and j. The graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$, the diagonal matrix, $D_{i,i} = \sum_{j=0}^{N-1} A_{i,j}$. So in [21], a graph signal is defined as $\mathbf{s} : V \to \mathbb{C}$. The vector form of signal for N elements will be

$$\mathbf{s} = [s_0, s_1, ..., s_{N-1}]^T \in \mathbb{C}^N \quad (1)$$

The graph filter system $\mathbf{H}(\cdot)$ produces the filtered output $\mathbf{H}(\mathbf{s})$ by taking the graph signal $\mathbf{s}$ as input. Graph shift operator, the basis of GSP, is defined as

$$\tilde{s_j} = \sum_{j \in M_i} A_{i,j} s_j \quad (2)$$

where M_i is the neighbourhood of the node v_i. So the output of graph shift can be written as

$$\tilde{\mathbf{s}} = [\tilde{s_0}, \tilde{s_1}, ..., \tilde{s_{N-1}}]^T = \mathbf{A}\mathbf{s}. \quad (3)$$

A linear, shift-invariant graph filter is a polynomial in the adjacency matrix of the form [3]

$$h(\mathbf{A}) = h_0 \mathbf{I} + h_1 \mathbf{A} + ... + h_L \mathbf{A}^L \ \ \& \ \ \mathbf{H}(\mathbf{s}) = h(\mathbf{A})\mathbf{s} \quad (4)$$

where $L \leq N$. Further, in GSP, the Fourier basis for a graph corresponds to the Jordan basis of the graph adjacency matrix $\mathbf{A}$. Additionally, a graph's spectrum is defined by its distinct eigenvalues $\lambda_0, \lambda_1, ..., \lambda_{P-1}$ of the graph adjacency matrix $\mathbf{A}$, called its graph frequencies, and its Jordan eigenvectors correspond to its frequency components [5]. The Jordan decomposition of $\mathbf{A}$ can be expressed as $\mathbf{A} = \mathbf{V}\mathbf{J}\mathbf{V}^{-1}$. Hence, the GFT is defined as $\hat{\mathbf{s}} = \mathbf{F}\mathbf{s}$ [3], where $\mathbf{F} = \mathbf{V}^{-1}$ is the GFT matrix. On the other hand, for graphs, the concept of total variation refers to the similarity of a graph signal and its shifted version [5].

$$TV_G(\mathbf{s}) = ||\mathbf{s} - \mathbf{A}^{norm}\mathbf{s}||_1 \ \ \& \ \ \mathbf{A}^{norm} = \frac{1}{|\lambda_{max}|}\mathbf{A} \quad (5)$$

where $TV_G(\mathbf{s})$ is the graph total variation, and $\mathbf{A}^{norm}$ is a normalized form of $\mathbf{A}$ with λ_{max} being the largest eigenvalue of $\mathbf{A}$. According to the definition of the total variation, if two eigenvalues are ordered as $\lambda_j < \lambda_i$, the TVs will be ordered as $TV_j > TV_i$. So for a graph having real spectrum, the frequencies will be reverse ordered as that of eigenvalues. The maximum eigenvalue represents the lowest frequency, and the minimum eigenvalue represents the highest frequency. The high pass filter $h(\mathbf{A})$ is defined as (we take $\lambda_{cut} = 0.8\lambda_{max}$)

$$h_l = \begin{cases} 1, & \lambda_l \leq \lambda_{cut} \\ 0, & \lambda_l > \lambda_{cut} \end{cases} \quad (6)$$

3. METHODOLOGY

Using our proposed methodology, we first extract patches, represent patches as graphs, implement a graph high-pass filter, apply the GFT, and finally apply GCN.

3.1. Description of Data

Acquired seismic data contain seismic amplitude values, which indicate the relative change in rock property impedance when moving from one layer to the next. The seismic impedance (I) of a layer in seismic data is the product of seismic velocity (v) and density (ρ), represented as $I = \rho \times v$. The seismic amplitude can take positive, negative, or zero values. The data used for the analysis comes from the Krishna-Godavari (KG) Basin, located in the Bay of Bengal, India. A small section of the KG-Basin data is represented in Fig. 2. Time data samples are into the earth and crossline and inline along with the earth. The seismic data can also be presented as a simple convolution operation that can be used to create synthetic data, shown in Eq. 7.

$$s(t) = w(t) * r(t) + n(t) \quad (7)$$

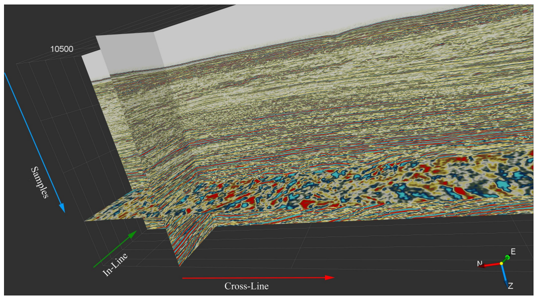
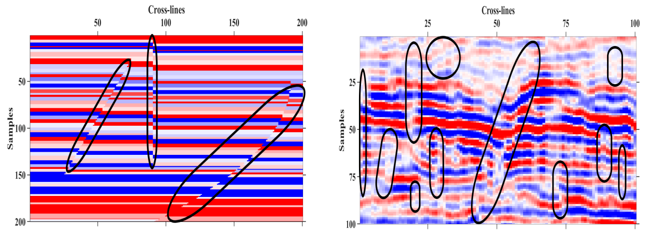


Fig. 2. 3D view of a small section of KG-Basin data.

where $s(t)$ presents a seismic trace, and $w(t)$ is the wavelet, $r(t)$ presents a reflectivity series, and $n(t)$ is noise. We generate synthetic data by convolving a synthetic reflectivity series with a ricker wavelet and assuming zero noise. Fig. 3a shows an example of synthetic data, and Fig. 3b shows real data. The training data contain equal parts synthetic and real data.



(a) Synthetic seismic (b) Real seismic

Fig. 3. e.g. of seismic data of 1 inline section with faults.

3.2. Extraction of Patches

A patch extraction is done by taking a patch whose central point is the point that we want to analyze; we determine if it is a fault point or not. A patch will always have an odd dimension since even patches cannot have a single central point. Before extracting patches, we normalize the data with mean 0 and standard deviation 1. A 10×10 dataset is shown in Fig. 4 with a simple example of patch extraction from two central points, O_F and O_{NF}. Similarly, we extract patches for all points in a given dataset. Patches of 3×3 dimensions are shown in Fig. 4, but we vary the patch size for our experiments, extracting patches up to 17×17.

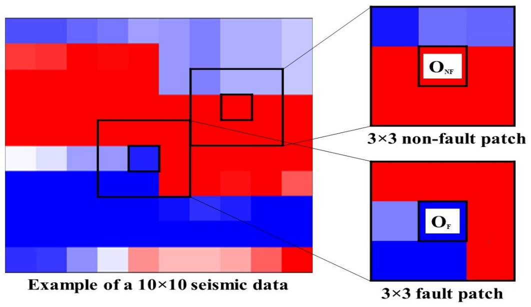
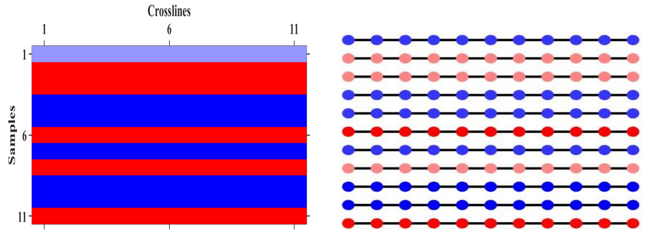


Fig. 4. Illustration of extraction of patches of dimension 3×3.

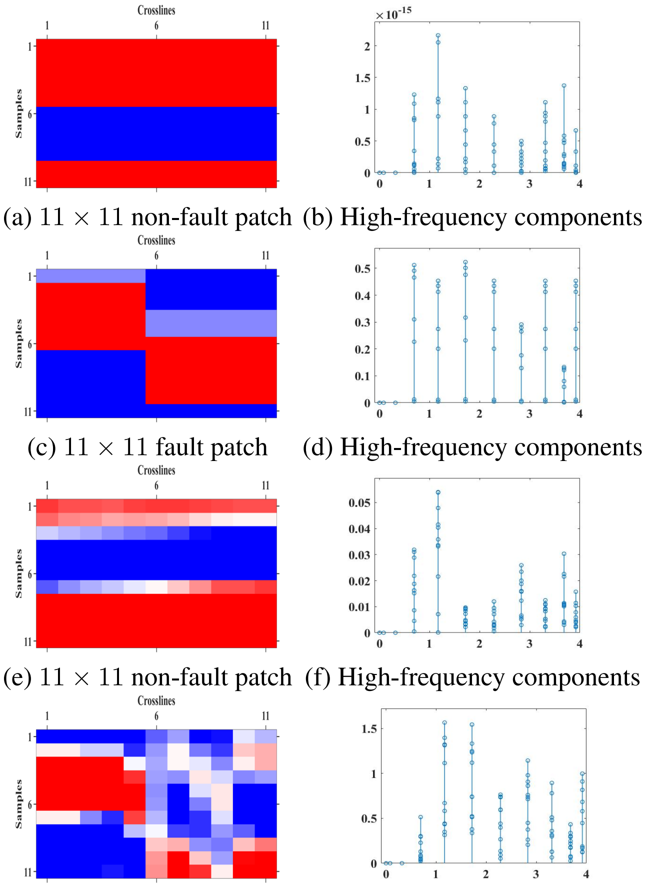
3.3. Graph High-Frequency Component

For the representation of patches in the graph domain, we consider a 2D-grid graph without vertical connections. Fault in seismic data is a kind of discontinuity among the seismic layers in a horizontal direction. In order to capture the variation in the horizontal direction, we consider unweighted and undirected lateral edges in a graph. Due to non-uniformity in the data, the vertical connections will capture the vertical variation, which is not relevant, and cause some graph high-frequency components even in real non-fault data. Fig. 5 shows an example of a graph representation of an 11×11 patch. The signal's most energy will be concentrated in low frequencies, which gets eliminated by implementing a graph high pass filter (Eq. 6). Fault points will have a large variation in seismic signal values along the lateral direction, and high-frequency components will result. Fig. 6a and 6c show examples of synthetic data 11x11 patch with non-fault and fault, and the high-frequency components in Fig. 6b and 6d, respec-



(a) 11×11 patch (b) Graph representation

Fig. 5. e.g. of graph representation of a patch.



(a) 11×11 non-fault patch (b) High-frequency components

(c) 11×11 fault patch (d) High-frequency components

(e) 11×11 non-fault patch (f) High-frequency components

(g) 11×11 fault patch (h) High-frequency components

Fig. 6. e.g. of high-frequency components of 11×11 patches.

tively. Fig. 6e and 6g show examples of real data patches, and their high-frequency components in Fig. 6f and 6h. The range of frequency coefficients is the distinguishing factor. For synthetic non-fault, it is 2×10^{-15}, but for synthetic fault, it is 0.5. Similarly, for real non-fault, it is 0.05, while for real fault, it is 1.5. Moreover, in the case of fault data, there are a lot more frequency coefficients in a higher range than that of non-fault.

3.4. Application of GCN

Node classification, link prediction, and clustering are the main objectives of graph representation learning. We use GCN for node classification, representing the $m \times m$ graph high-frequency components from $m \times m$ patch as a single node of m^2-dimensional vector in a single k-nearest neighbor (kNN) graph. In the kNN graph, each node is connected to its six nearest neighbors using undirected edges with weight $A_{i,j} = exp(-d_{i,j}^2)$, where $d_{i,j}$ is the distance between the high-frequency components of ith and jth node. The kNN graph is the input to GCN. The label information can be smoothed using the Laplacian regularization term in the loss function E for node classification using GCN [9].

$$E = E_0 + \lambda L_{reg}, with$$
$$L_{reg} = \sum_{i,j} A_{i,j}||f(X_i) - f(X_j)||^2 = f(X)^T L_u f(X) \quad (8)$$

where E_0 = supervised loss with respect to the label information, $f(\cdot)$ = the neural network, λ = weighing factor, X = matrix of feature (graph high-frequency components) vectors, and L_u = Unnormalized graph Laplacian of a graph $G = (V, \mathbf{A})$ with N nodes and an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$. Layer-wise propagation is used for the application of multi-layer GCN [22].

$$H^{(l+1)} = \sigma(\bar{D}^{-1/2}\bar{A}\bar{D}^{-1/2}H^{(l)}W^{(l)}) \quad (9)$$

where $\bar{A} = \mathbf{A} + \mathbf{I}_N$, and $\bar{D} = \sum_j \bar{A_{i,j}}$, σ = activation function, $H^{(l)}$ = matrix of activation in lth layer, $H^{(0)} = X$, and $W^{(l)}$ = trainable weight matrix. Therefore, a two-layer GCN can be defined as a forward model using $\tilde{A} = \bar{D}^{-1/2}\bar{A}\bar{D}^{-1/2}$

$$Y = softmax(\tilde{A}ReLU(\tilde{A}XW^{(0)})W^{(1)}) \quad (10)$$

$W^{(0)}$ is an input-to-hidden weight matrix for a hidden layer, whereas $W^{(1)}$ is a hidden-to-output weight matrix. For training the weights $W^{(0)}$ and $W^{(1)}$, we use batch gradient descent and Eq. 10. In addition, we use dropout regularization, cross entropy loss, and we train the GCN model with sixteen hidden layers having ReLU activation for hidden layers and softmax for output. The GCN architecture we use has an initial learning rate of 0.01, and a drop out of 0.5.

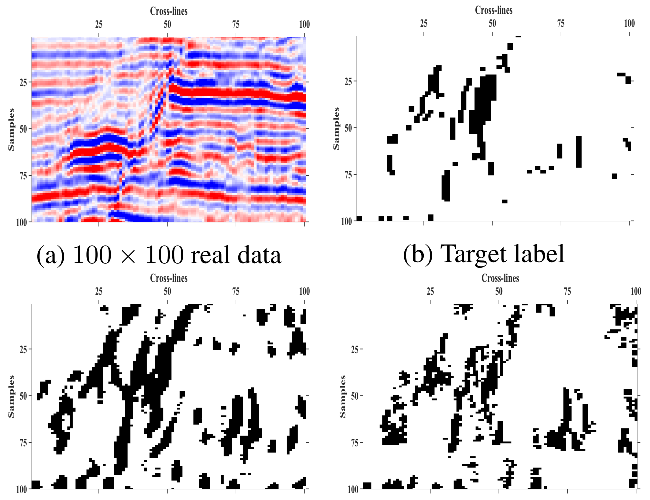
4. RESULTS AND DISCUSSIONS

In order to construct the graphs, we use the graph signal processing toolbox [23], and PyTorch is used to design the GCN. We take 10000 for each fault and non-fault data for the training dataset and validation 2000 for each fault and non-fault

Patch dimension	Accuracy	Patch dimension	Accuracy
3×3	83.19%	11×11	**90.67%**
5×5	88.35%	13×13	90.11%
7×7	89.02%	15×15	89.44%
9×9	89.44%	17×17	88.65%

Table 1. Accuracies for different dimensions of patches.

data. Using these 24000 data, we test 100×100 real field data that is shown in Fig. 7a. So when we extract patches from each point, there will be 10000 testing data patches. Fig. 7b shows the manually created label for the data. These labels are created under the expert guidance of geophysicists, and we compute the accuracy values by referring to these manually created labels. We compare the performance of our method by implementing a CNN-based method. We apply CNN to extracted 2D patches in the same way as the authors describe in [18]. Fig. 7c shows the results obtained from using CNN. False fault point detection happens because the points located close to faults are detected as fault points, which affects the accuracy in the case of CNN. Fig. 7d shows the results obtained using GSP high-frequency components as input to GCN. Table 1 shows the accuracies obtained from the different sizes of patches. When we vary the dimension of patches, the accuracy is the highest for 11×11 patches.



(a) 100×100 real data (b) Target label

(c) Label using CNN (Accuracy: 82.89%) (d) Label using GSP & GCN (Accuracy: **90.67%**)

Fig. 7. e.g. of results obtained on a 100×100 real data.

5. CONCLUSION

We present a method to identify faults in seismic data by using graph high-frequency components as inputs to a graph convolutional network. We first extract the patches and then present the patches using a laterally connected graph. Next, we obtain the graph high-frequency components by implementing a graph high-pass filter and graph Fourier transform. Then we identify the fault by classifying the patches using GCN, and it shows good accuracy when applied to real field data.

6. REFERENCES

[1] F. R. K. Chung, *Spectral Graph Theory*, Providence, RI, USA: American Mathematical Society, 1997.

[2] J. F. Tenenbaum, V. Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319–2323, 2000.

[3] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs: Graph filters," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 6163–6166.

[4] X. Zhu and M. Rabbat, "Approximating signals supported on graphs," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2012, pp. 3921–3924.

[5] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs: Frequency analysis," *IEEE Transactions on Signal Processing*, vol. 62, no. 12, pp. 3042–3054, June 2014.

[6] David I Shuman et. al., "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[7] Zonghan Wu et. al., "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2021.

[8] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*. 2016, NIPS'16, p. 3844–3852, Curran Associates Inc.

[9] Thomas N. Kipf and Max Welling, "Semi-supervised classification with graph convolutional networks," *CoRR*, vol. arXiv:abs/1609.02907, 2016.

[10] David K Duvenaud et. al., "Convolutional networks on graphs for learning molecular fingerprints," in *Advances in Neural Information Processing Systems*. 2015, vol. 28, pp. 2224–2232, Curran Associates, Inc.

[11] Liang Yao, Chengsheng Mao, and Yuan Luo, "Graph convolutional networks for text classification," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 7370–7377, Jul. 2019.

[12] Sami Abu-El-Haija, Amol Kapoor, Bryan Perozzi, and Joonseok Lee, "N-GCN: multi-scale graph convolution for semi-supervised node classification," *CoRR*, vol. arXiv:abs/1802.08888, 2018.

[13] Nasher M. AlBinHassan and Kurt Marfurt, "Fault detection using hough transforms," in *SEG Technical Program Expanded Abstracts*, 2005, pp. 1719–1721.

[14] Zhen Wang and Ghassan AlRegib, "Fault detection in seismic datasets using hough transform," in *IEEE International Conference on Acoustic, Speech and Signal Processing*, May 2014, pp. 2372–2376.

[15] Rahul Mahadik and Aurobinda Routray, "Fault detection and optimization in seismic dataset using multiscale fusion of a geometric attribute," in *IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society*, 2019, vol. 1, pp. 107–112.

[16] X. Wu et. al., "Faultnet3d: Predicting fault probabilities, strikes, and dips with a single convolutional neural network," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 11, pp. 9138–9155, Nov 2019.

[17] Augusto Cunha, Axelle Pochet, Hélio Lopes, and Marcelo Gattass, "Seismic fault detection in real data using transfer learning from a convolutional neural network pre-trained with synthetic seismic data," *Computers and Geosciences*, vol. 135, pp. 104344, 2020.

[18] Wei Xiong et. al., "Seismic fault detection with convolutional neural network," *GEOPHYSICS*, vol. 83, no. 5, pp. O97–O103, 2018.

[19] Haibin Di, Amir Shafiq, and Ghassan Alregib, "Seismic-fault detection based on multiattribute support vector machine analysis," in *SEG Technical Program Expanded Abstracts*, 08 2017, pp. 2039–2044.

[20] Lei Huang, Xishuang Dong, and T. Edward Clee, "A scalable deep learning platform for identifying geologic features from seismic attributes," *The Leading Edge*, vol. 36, no. 3, pp. 249–256, 2017.

[21] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1644–1656, 2013.

[22] Ming Chen et. al., "Simple and deep graph convolutional networks," in *Proceedings of the 37th International Conference on Machine Learning*. Jul 2020, vol. 119, pp. 1725–1735, PMLR.

[23] Nathanaël Perraudin, Johan Paratte, David Shuman, Lionel Martin, Vassilis Kalofolias, Pierre Vandergheynst, and David K. Hammond, "GSPBOX: A toolbox for signal processing on graphs," *ArXiv e-prints*, August 2014.