Graph Neural Networks for Music Genre Classification

Shardul Dhongade
Virginia Tech
shardul21@vt.edu

Nikolas Rovira
Virginia Tech
nikolasr20@vt.edu

Abstract

Music genre classification is a well-explored application of deep learning, often employing Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) to identify patterns in audio features. With the growing popularity of Graph Neural Networks (GNNs) and their emerging applications, this paper explores the use of GNNs for music genre classification. Prior work has been done using GNN's for music recommender systems and classification through representation learning, but is not as extensive. We aim to further contribute to this domain and explore the practicality of GNN's for these applications.

This paper will explore ideas from previous work to build a node classification model through a Graph Convolutional Network (GCN). The graph edges were determined through the use of cosine similarity, and the graph representation was passed into the GNN.

We demonstrate the success the GNN are able to achieve on the popular GTZAN music dataset. The GNN model scores roughly 89% accuracy, and has a performance comparable to the baseline model built as a Dense Neural Network, along with various publications on CNN results.

We will additionally discuss the potential benefits and tradeoffs of GNNs within this domain, and explore other approaches and fine-tuning methods that could establish better results.

Keywords

Graph, Neural Networks, Music, Genre, Classification

ACM Reference Format:
Shardul Dhongade and Nikolas Rovira. 2018. Graph Neural Networks for Music Genre Classification. In . VT, Blacksburg, VA, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Music genre classification is a common project done within Deep Learning, with further applications ranging from music recommendation systems to personalized playlist generation. Classification operates by categorizing music tracks into predefined genres based on the given song's audio features and patterns. As the music industry continues to expand with new forms of music and overlapping genres, classification of songs into single categories becomes more challenging. Genres like pop and rock are infamous for being difficult to classify as in many cases they can sound similar. Traditional machine learning approaches for audio tasks initially rely on feature extraction, where domain-specific features such as Mel-Frequency Cepstral Coefficients (MFCC) and chroma features [3] are selected and pre-processed, and then fed into the model.

Spectrograms are also commonly used, when data is presented in the form of audio files. The audio can be converted to a spectrogram, which is an image that represents the strength of the signal over time at several frequencies. Amplitude can also be displayed as color on the graph, indicating where it is higher and lower. From here, a Mel-Spectrogram can be created, which is just a spectrogram with the frequencies being converted onto the mel scale. A Mel-spectrogram creates better representations for lower frequencies and for the frequencies that humans can hear. This makes it a great tool for audio analysis. [2]

The most commonly employed deep learning techniques for music genre classification include Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). CNNs excel with audio spectrograms [1], and RNN's have shown promise with Long Short Term Memory Network (LSTM) due to its better handling of spatial and temporal features [4]. Both approaches have shown promising performance in several studies.

While CNNs and RNNs have been proven to be successful, they lack the degree of flexibility that GNNs may offer. Neural Networks can be limited when capturing complex relationships in data elements, such as the interactions between different musical features. GNNs provide a more flexible framework to model data with graph structures, which is beneficial for non-Euclidean data. GNNs have already proven effective in various domains like social networks, molecular chemistry [5], and natural language processing, where data can be represented as nodes and edges.

Some of the main GNN tasks include graph classification, node classification, and link prediction. In this paper we explore a node classification model, which aims to determine assign labels to nodes in the graph based on the features of the neighboring nodes and its relationships between them. Performance for node classification models are generally measured using accuracy and F1 scores. [6,7]

This task will use a Graph Convolutional Network (GCN), which will directly operate on the graph. GCNs are standard for node classification tasks, and are similar to the convolutions in CNNs. The GCN works by learning the features from its neighboring nodes, and can handle the varying number of neighbors for each node.[10]

All of this work will be done through Python, largely relying on the PyTorch and Torch Geometric libraries. These will be responsible for building the graph respresentation and training the neural network model through GCN layers.

These components make up the Graph Neural Network we are using for music genre classification. Previous work in this field discusses the use of GNNs for music recommendation systems [9]. We aim to explore the effectiveness of GNNs in capturing relationships between musical features and genres, and compare their performance against CNN approaches and our baseline DNN model.

2 List of Contributions

Below we will list the following contributions. We would like to note that the architecture for GNNs and node classification tasks are generally implemented in a standard way. While we have made all the following contributions, the next several sections will list all references for where the intuition for certain approaches came from, as well as citations to the bare architectures that were used to implement our models.

We process the dataset ourselves and build the graph for the GNN. Initial experimentation with Mel-Spectrograms is also conducted. We create the cosine-similarity matrix to determine the edge connections for the nodes on the graph, and convert them to edge tensors for graph representation. We also handle the training and testing masks for the graph, so that the graph utilizes transductive learning. The GNN model is also built and trained in our code, using PyTorch and Torch Geometric capabilities. We modify them further by experimenting with hyperparameter tuning ourselves. This includes increasing layer count of GCN, regularization experimentation, and changing neighbor counts.

We also reproduce the baseline model adapted from *Music Genre Classification Using CNN* by Arsh Chowdhry [12]. This is then modified and experimented with to see if performance can be further improved.

We then implement accuracy and precision metrics, along with recall and F1 scores to evaluate the performances of both models (tracked across each epoch for the GNN model). These are standard evaluation methods that we utilize for our purposes as well.

3 Related Work

3.1 GNNs for Music Recommendations

Given the large amount of music data, as well as the economic incentive for the efficient supply of music, a large amount of research has been performed focusing on music processing and recommendation. One example of this is Ben Alexander, Jean-Peic Chou, and Aman Bansal's *Implement Your Own Music Recommender with Graph Neural Networks (LightGCN)* [9]. This research used data from the Spotify Million Playlist Dataset to train a GNN to recommend songs to be added to playlists. They utilized the playlists and the songs within them to suggest tracks for playlists considered similar. This nature of the dataset indicates that the music recommendations are based on user behavior. If multiple users listen to the same songs in the same playlists, the suggested songs will reflect those patterns.

3.2 GNNs for Social Networks

Social networks are a prime example of where graphs are widely used, and as a result were utilized heavily for research with Graph Neural Networks. The team at Vizuara analyzed a a social network using GNN, where they attempt to classify the students based on their country. The goal is to determine if students from the same country interact with each other more. Here, they also apply a GCN for classification. They split the labels into four countries map relationships in this manner. This is an example of a node classification task, where they attempt to determine the labels based on the neighboring nodes.[11]

3.3 GCNs for Semi-Supervised Node Classification

The GCN is the backbone of many Graph Neural Network models, and is also commonly associated with node classification tasks. In the paper *Hierarchical Graph Convolutional Networks for Semi-supervised Node Classification* by Fenyu Hu, Yanqiao Zhu, Shu Wu, Liang Wang, and Tieniu Tan explore this further. The paper takes a look at how to expand the aggregation of GCNs, by reaching further into the graph rather than just one or two neighborhoods down. This way, information further away can be captured and allow the model to perform better. This is good for node classification tasks as capturing global information can improve results.[16]

3.4 Node Classification vs Link Prediction

Initially, our intention was to use link prediction as we referred to some other research papers on GNNs, but we quickly switched to node classification. Our initial logic was that we could use groupings of nodes by predicted edges to then prescribe genres but we realized it is better just to use node classification. Node classification is directly applicable to this problem as determining the genre of each node is directly a classification problem.

All of this is not to say link prediction cannot be used in this domain. Link prediction could be used in cases where undefined connections between music want to be found or other graph topologies such as each node representing playlists.[9]

4 Model Description

Two models were used for music genre classification: a graph neural network, and a fully connected dense neural network. The DNN was developed as a baseline model, where the code was largely adopted from *Music Genre Classification Using CNN* by Arsh Chowdhry. [12]

4.1 Dense Neural Network

The dense neural network was fairly standards, and consisted of five hidden layers. The neuron counts were, in order, 512, 256, 128, 64, 10. The first four layers used ReLu activation functions accompanied by a dropout rate of 0.3. ReLu was picked for its good back propagation characteristics and computational efficiency, and dropout was implemented to prevent overfitting and promote regularization. The final layer used a softmax activation function to output the final 10 length vector, representing the probbailites for the 10 genres. For training, we used an Adam optimizer for its efficient convergence and saddle point avoidance. Finally, the loss function used was sparse categorical cross entropy due to the integer representation of our ten class labels.

4.2 Graph Neural Network

4.2.1 Creating the Graph. Before creating the graph neural network, the graph representation of the music dataset needed to be created. Some other implementations of GNNs use a fully connected graph where every node is connected to every other node. This was unrealistic for our implementation given the layer depth and data size. If every node were connected to every other node, there would be 99,790,110 edges. [13]

The number of edges needs to be minimized, and to determine these reduced edge connections, cosine similarity was used on each node's feature vectors. Performing a cosine similarity on the data creates a similarity matrix which can be used to determine which nodes have the highest similarity. Although the matrix is computed, we still need to determine how many similar nodes will have edges between them. For this, we implement an approach akin to K-Nearest Neighbor to determine the number of neighbors for each node. The top k nodes with the highest similarity will have an edge with the node, which is bidirectional since the graph is undirected. This way, the number of edges in the graph is exponentially minimized and will speed up computation time. Using the cosine similarity matrix with a max neighbor count of 10, reduced the number of edges for our dataset to 179,820, which is 0.18% of a fully connected graph. We argue that this should greatly reduce training time and memory requirements as well as allowing for more productive information passing between nodes.[14,15]

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2}\sqrt{\sum\limits_{i=1}^{n} B_i^2}}$$

Figure 1: Cosine Similarity Equation used for Edge Prediction [17]

4.2.2 Creating the Graph Neural Network. The architecture for the GNN was learned from BASIRA Lab, specifically from their Deep Graph Learning series on YouTube, taught by Professor Islem Rekik at the Imperial College London. The content covered discusses the architecture as well as regularization techniques for GNNs.[18]

A single layer of a GNN generally consists of the convolutional or linear layer with its activation function. Along with this, batch normalization and dropout can be implemented for regularization. Dropout has been shown to increase the expressiveness of Graph Neural Networks, and so we implemented it in our model.[19]

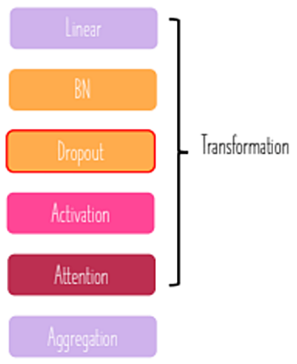

Figure 2: Components of a single GNN layer.[29]

Our GNN consists of the GCN, three graph convolutional layers and ReLu activation. Batch normalization and dropout were both included for regularization. This will further help the model to better generalize.

The initial input into the model is the audio feature vector, length of 57, from the dataset that were passed into the PyTorch Data object. We set the hidden dimension to be 128, and had the output dimension be size 10 to match the number of labels.

The forward pass will first take in the node feature matrix and the edge tensors, which represent the graph's connectivity. It will apply the graph convolution here, then call the ReLU activation function. Batch normalization and a dropout rate of 0.4 are both applied to the layer as well. The second layer will perform the exact same operation another time, in an attempt to visit deeper nodes and strengthen relationships.

The third and final graph convolution layer will take in the feature matrix and edge tensors, and return this output.

Same as the dense neural network, Adam optimizer was used for optimization and sparse categorical cross entropy was used to calculate loss.

Because cross entropy loss is already used, there is no reason to set the output to softmax since this is redundant. Python's cross entropy loss function contains a softmax calculation already.

For training and testing this model, we had the option of choosing between transductive learning and inductive learning, two common approaches for GNNs. With inductive learning, we keep the test graph separate from the training graph. After training, we would take the test graph and assign labels to each node. With transductive learning, the exists one graph which contains both the training and testing sets. Essentially, there will be some nodes which have labels, the training set, and some nodes which do not, the test set. This way, the model will only train using the labeled data, and then use the patterns from the unlabeled data to make decisions. This way, loss is computed only on the training subset of the data, while also retaining access to the testing node vector representation for forward passes. A disadvantage of this is that it incurs a higher memory cost due to the fact it stores information for both the training and testing nodes. However in our case, this is not a major issue since the graph size is not absurdly large.[20,21]

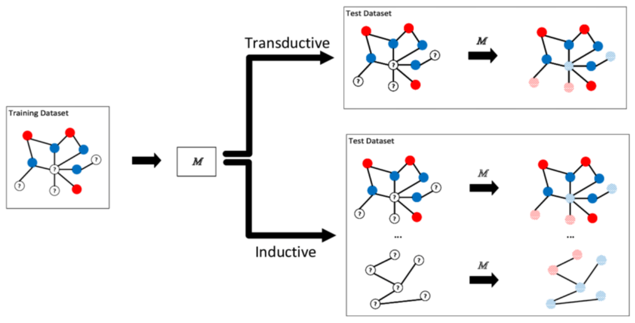

Figure 3: Transductive vs Inductive Learning.[22]

5 Data Description

5.1 Dataset

The dataset we used to train our models was the GTZAN Audio Dataset. This dataset contains 1000 tracks, each of 30 second length [8]. The data was taken from a Kaggle set, and we utilized the features 3 CSV file. This file contained data on all major attributes of each given audio track. These include those as discussed above, such as the MFCCs, chroma, and their mean values. MFCCs, or Mel-frequency Cepstral Coefficients, help to see the short term power spectrum of a given sound. The Mel-scale was developed based on how humans distinguish frequencies and perceive sound. This way, using the Mel-scale for MFCCs makes it easier to distinguish the sounds humans hear, and better for feature extraction. [23]

Chroma Features essentially aid in capturing the harmonic characteristics of music and remain robust to any changes in the timbre. This can be used for categorizing pitches.

It also contains additional important features such as zero cross rate, spectral bandwidth and rolloff.

Zero cross rate is used as a measure of frequency content, so given an interval it determines how many times the signal amplitude passes through the value of zero.

Spectral rolloff looks at the spectrum and determines the frequency that exists below a given percentage of the total spectral energy that is present. It provides information on how the energy is distributed across frequencies.

These are some major components of the data, and they all largely focus on the frequencies of given signals, amplitudes, and different musical attributes.

Several research papers on music classification and analysis use this dataset.[24]

The data set structure consists of 10 labeled groups, each with 100 songs. These groups are split by genre, those being: blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, and rock. For each of these songs, the audio wave file, feature vector, and labels are provided. Each vector is of length 60 with 57 of them being audio attributes as described above. This feature set will be used as the main dataset for the model.

5.2 Preparing the Data

Standard preprocessing was done for the dataset using python libraries such as pandas, numpy, and sci-kit learn. The unnecessary labels were dropped from the dataset, and the labels column was then encoded so that they would be represented in numeric terms. This means a label of zero would include the Blues genre, a label of one would indicate the Classical genre, and so on. We then extract the features and standardize them for use.

Since we are using the PyTorch libraries for training, along with the torch geometric for graph capabilities, we must convert the data to tensors first. Tensors are the data structure used by PyTorch to encode the model's parameters, inputs, and outputs. For this reason, we convert both the data set and the label set to tensors. We also must do the same for the graph representation created from the cosine similarity above. Additional information and tutorials on tensors can be found in the PyTorch documentation.[26]

6 Results

6.1 Evaluation Results

The model was trained over 1000 epochs and performed very well on the test data. It received approximately 89.2% accuracy on the test set, which indicated that the model fit the data well and is able to generalize.

The metrics used here were accuracy and precision scores since this is a classification task, along with recall and F1 which are considered standard for GNNs.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Figure 4: Accuracy, Precision, Recall, F1 equations.[25]

The accuracy is just the performance, and looks at how many correct classifications were made divided by the total classifications. The recall is a similar accuracy metric, but instead will look at the correctly classified positives divided by the total actual positives. This helps to better see the likelihood of correct classification.

Precision will take a look at how many of the so called positive predictions made by the model were in fact correct.

The F1 score will be a combination of the precision and recall, and is the harmonic mean of the two scores. Because we expect correct classifications to be high, it encourages the precision and recall scores to be similar. The further apart they are, the lower the F1 score will be.

The model had an accuracy and recall score of 89.19% and a precision score of 89.26%. The F1 score was 89.18%. This shows great performance of the model.

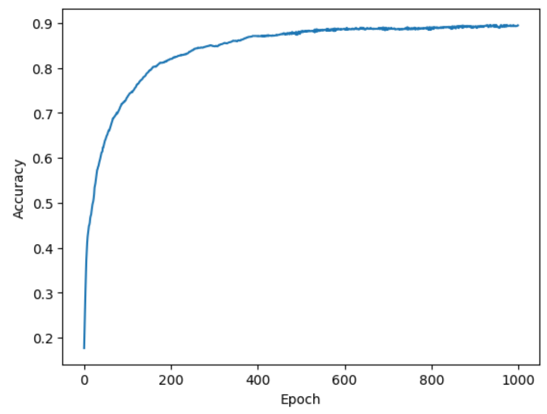

Figure 5: Training Accuracy by Epoch of GNN

The baseline model used was the DNN, which ran for 100 epochs and had a test accuracy of roughly 94.3%. The precision and recall scores for this model were 94.37% and 94.34% respectively, producing a strong F1 score of 94.34%.

We also examine the paper *Convolutional Neural Networks Approach for Music genre Classification* by Yu-Huei Cheng, Pang-Ching Chang, and Che-Nan Kuo, where they built a CNN using the GTZAN dataset. Their results indicated that the highest accuracy their model received was 83.3%. [27]

Overall, we can see that our GNN model had comparable performance to others. The model managed to perform better than some CNN models, those listed in the above paper. However, it did not perform better than the baseline DNN model we had trained.

6.2 Experimental Results

Additionally, once the models were built, we experimented with several changes in parameters to try and increase model effectiveness. For the GNNs, we first tried to determine if adding additional layers would benefit. We had two layers and noticed a slight improvement with three layers. However, anything more did not cause any significant improvements to the performance. Therefore, we decided to leave it at three layers. We also experimented with dropout. As stated above dropout has been shown to help for GNNs, so we tested out different values between 0.2 to 0.5. We found that anything less than 0.3 decreased the model performance, so we decided to restrict the dropout to 0.4. While we did not implement early stopping, we found that our accuracy had no significant increase after approximately 800 to 1100 epochs. We landed on testing it for 1000 epochs.

For the DNN model, the one we borrowed already had strong performance, but we were able to strengthen it slightly by modifying the dropout and epoch count. Increasing the dropout to 0.3 helped and the epoch count was lowered to 100. It was able to achieve the 94.3% accuracy on the test set, while the initial version in the paper was 92.93%. These were some areas we experimented with and were able to find success on.

7 Lessons Learned

Overall, this project was a great learning experience for us. Graph Neural Networks was a novel topic that is not extensively covered in many curriculums, and it required us to conduct a lot of research on our own. Since we are both relatively new to the field of Deep Learning, it was a learning curve to understand several concepts aside from just Graph Neural Networks. Working with PyTorch and grapsing concepts such as tensors and masks proved to be challenging as well. This did make it all the more rewarding too though, as we were able to successfully complete what we had set out to do.

This project was able to show us a lot about how Graph Neural Networks operate and their vast capabilities in several domains. When our initial model failed due to the large number of edges, we understood just how computationally intensive these tasks can be and the GPU overhead required. It demonstrated that preprocessing your data is just as important as the training model, given how large a difference it can make.

One thing we would have done differently if given the opportunity is to try different versions of GNNs to determine which would perform the best. We initially had in mind to convert the data into MEL-Spectrograms, and use that as the feature vectors. This would have been additional steps, but it would be cool to see how this would perform if done through GNNs. Another idea is to use graph based classification instead of node classification, by having a sparse network and classifying graphs into separate categories.

Barring a few roadblocks, this project went well and was generally successful. We did run into some issues initially with the complete graph having too many edges, causing massive computation times and very slow performance. Moreover, as stated in our midterm report, we attempted to use Link Prediction initially for our graph. This did not work as we expected, as we quickly realized building the adjacency matrix was not suitable for this specific case. It is commonly used for social networks, and would not translate well into this particular classification task.

8 Broader Impacts

A well tuned graph neural network music classifier, although not revolutionary, should assist music creation by musicians and distribution by platforms. Music is a continuously consumed commodity with consumer's access to new music being pivotal. Finding similarities and classifying music, like the model in the paper does, should aid in the recommendation of new music to these consumers.

The classification of music, particularly older compositions, could play a beneficial role in music preservation and historical research. Analyzing large datasets of music from past centuries and identifying classifications or similarities among them can broaden our understanding of these works.

Looking past just music genre classification, the goal was also to see how powerful Graph Neural Networks are. While they have their drawbacks such as higher computation cost and weak performance with small datasets, several improvements have been made over the years to improve these models. In the modern world, more machine learning problems are presented with data that in inherently non-Euclidean. GNNs tend to excel with non-Euclidean data as compared to current traditional networks. As the problems faced in machine learning become more and more difficult, GNNs could serve as novel solutions. Contributing to this domain could help make their potential stand out further, and play a pivotal role in machine learning over the next few years.[28]

9 Conclusion

Overall, the use of graph neural networks has demonstrated its potential to be a viable option for music genre classification. Through with this model achieving a final accuracy of 89.26%. With further tweaking of hyper-parameters and more advanced edge assignment, this type of model should be able to achieve even higher accuracies. This will allow for more efficient and accurate music classifications, which is becoming increasingly important as music streaming grows in popularity.

10 Work Distribution

The work was split equally among both of us.

Shardul handled the data preprocessing and exploratory analysis of the features in the dataset. He also tested different correlation methods that could be beneficial for lowering the edge counts. He also handled the evaluation metrics. The baseline DNN model borrowed was incorporated by him as well.

Nikolas handled the initial versions of the graph and dealing with the train-test masks for the graph representation. He looked at the tensor implementations with PyTorch and utilizing them. He outlined the initial GNN architecture and looked at key metrics to use.

Both of us worked together on several aspects of this process. We both looked at creating the cosine-similarity matrix and determining the neighbor count. We both also incorporated regularization into the GNN and experimented with different parameter values and counts. We handled the evaluation functions and the training. Overall, equal amounts of work was done by both members of the team.

References

[1] Reddy, A. (2023) Convolutional Neural Networks (cnns) for audio data or audio feature extraction, Medium. https://anudeepareddy-s.medium.com/convolutional-neural-networks-cnns-for-audio-data-or-audio-feature-extraction-1c41f4aac35d

[2] Roberts, L. (2024b) Understanding the mel spectrogram, Medium. Available at: https://medium.com/analytics-vidhya/understanding-the- mel-spectrogram-fca2afa2ce53

[3] Julian Urbano, Dmitry Bogdanov, Perfecto Herrera, Emilia Gomez and Xavier Serra. "What is the effect of audio quality on the robustness of MFCCs and chroma features?". Proceedings of the 15th Conference of the International Society for Music Information Retrieval (ISMIR 2014); 2014 Oct 27-31; Taipei, Taiwan. International Society for Music Information Retrieval; 2014. p. 573-78.

[4] Prem Tibadiya. 2020. Music Genre Classification using RNN-LSTM. (July 2020). Retrieved October 21, 2024 from https://medium.com/@premtibadiya/music-genre-classification-using-rnn-lstm-1c212ba21e06.

[5] Yuyang Wang, Zijie Li, and Amir Barati Farimani. Graph Neural Networks for Molecules. A Chapter for Book “Machine Learning in Molecular Sciences". arXiv:2209.05582v2 (2023).

[6] Node classification (no date) Papers With Code. Available at: https://paperswithcode.com/task/node-classification

[7] Kipf, T.N. and Welling, M. (2017) Semi-supervised classification with graph Convolutional Networks, arXiv.org. Available at: https://arxiv.org/abs/1609.02907v4

[8] Bob L. Sturm. (2013) The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use. arXiv:1306.1461v2

[9] Alexander, B. (2021) Implement your own music recommender with Graph Neural Networks (LightGCN), Medium. Available at: https://medium.com/@benalex/implement-your-own-music-recommender-with-graph-neural-networks-lightgcn-f59e3bf5f8f5

[10] Awan, A.A. (2022) A comprehensive introduction to Graph Neural Networks (GNNS), DataCamp. Available at: https://www.datacamp.com/tutorial/comprehensive-introduction-graph-neural-networks-gnns-tutorial

[11] Analysing a Social Network using Graph Neural Network Google colab. Available at: https://colab.research.google.com/drive/1x-MRiW0fooJFnFTFv9FvWW-lsWZJNb17?usp=sharingscrollTo=xG4_WrAFGAkj

[12] Chowdhry, A. Music genre classification using CNN, Clairvoyant. Available at: https://www.clairvoyant.ai/blog/music-genre-classification-using-cnn

[13] Dokania, S. and Singh, V. (2019) Graph Representation Learning for Audio & Music Genre Classification. Available at: https://arxiv.org/pdf/1910.11117.pdf.

[14] H. Yao, Y. Huang, J. Hu and W. Xie, "Cosine similarity distance pruning algorithm Based on graph attention mechanism," 2020 IEEE International Conference on Big Data (Big Data), Atlanta, GA, USA, 2020, pp. 3311-3318, doi: 10.1109/BigData50022.2020.9378189.

[15] Li, L. et al. (2024) ‘Knn-GNN: A powerful graph neural network enhanced by aggregating K-nearest neighbors in common subspace’, Expert Systems with Applications, 253, p. 124217. doi:10.1016/j.eswa.2024.124217.

[16] Hu, F. et al. (2019) ‘Hierarchical graph convolutional networks for semi-supervised Node Classification’, Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, pp. 4532–4539. doi:10.24963/ijcai.2019/630.

[17] Varun (2020) Cosine similarity: How does it measure the similarity, maths behind and usage in Python, Medium. Available at: https://towardsdatascience.com/cosine-similarity-how-does-it-measure-the-similarity-maths-behind-and-usage-in-python-50ad30aad7db

[18] DGL-4-clean.PDF (no date) Google Drive. Available at: https://drive.google.com/file/d/1oDmri4iCqD6vE7iJIEgW5ghnMqW16V7s/view

[19] András Papp, P. et al. (no date) Neurips. Available at: https://proceedings.neurips.cc/paper_files/paper/2021/file/d56b9fc4b0f1be8871f5e1c40c0067e7-Paper.pdf

[20] [Deep Graph Learning] 4.5 Generalized GNN layer and Dropout (no date) YouTube. Available at: https://www.youtube.com/watch?v=3e5zjVKsbsw&list=PLug43ldmRSo14Y_vt7S6vanPGh-JpHR7T&index=19

[21] G. Lachaud, P. Conde-Cespedes and M. Trocan, "Comparison between Inductive and Transductive Learning in a Real Citation Network using Graph Neural Networks," 2022 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Istanbul, Turkey, 2022, pp. 534-540, doi: 10.1109/ASONAM55673.2022.10068589.

[22] Comparison between transductive and inductive setting in GSSL. for... | download scientific diagram. Available at: https://www.researchgate.net/figure/Comparison-between-transductive-and-inductive-setting-in-GSSL-For-transductive-setting$_fig2_3$49682561

[23] GeeksforGeeks (2024) Mel-frequency cepstral coefficients (MFCC) for speech recognition, GeeksforGeeks. Available at: https://www.geeksforgeeks.org/mel-frequency-cepstral-coefficients-mfcc-for-speech-recognition/

[24] Andrada (2020) GTZAN dataset - music genre classification, Kaggle. Available at: https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification

[25] Machine learning model metrics – trust them? | FTI consulting. Available at: https://www.fticonsulting.com/en/canada/insights/articles/machine-learning-model-metrics-trust-them

[26] Tensors (no date) Tensors - PyTorch Tutorials 2.5.0+cu124 documentation. Available at: https://pytorch.org/tutorials/beginner/blitz/tensor_tutorial.html?highlight=cuda

[27] Y. -H. Cheng, P. -C. Chang and C. -N. Kuo, "Convolutional Neural Networks Approach for Music Genre Classification," 2020 International Symposium on Computer, Consumer and Control (IS3C), Taichung City, Taiwan, 2020, pp. 399-403, doi: 10.1109/IS3C50286.2020.00109.

[28] Sanborn, S. et al. (2024) Beyond Euclid: An illustrated guide to modern machine learning with geometric, topological, and algebraic structures, arXiv.org. Available at: https://arxiv.org/abs/2407.09468v1

[29] Basiralab Basiralab/DGL: Deep graph-based learning course (ICL, computing), GitHub. Available at: https://github.com/basiralab/DGL