Graph Neural Networks For Music Genre Recognition

Shuo Feng sf587
Jingyi Chen jc2498
Cornell University

Introduction

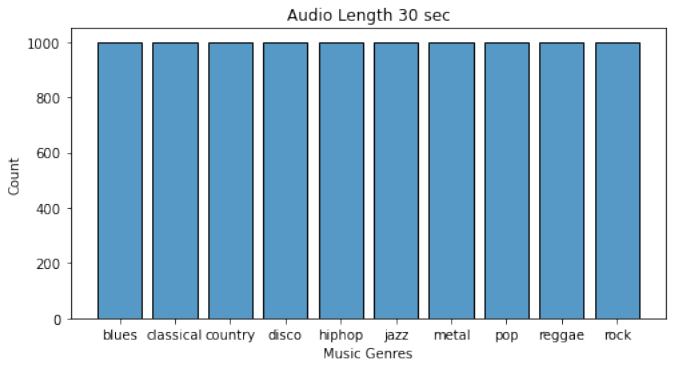
In the last decade or so, the big messy data is well trained by deep learning in many different fields like natural language processing, computer vision, and speech recognition. The innovation speed of deep learning algorithms is very fast while no one network is considered perfect. Most generally, the standard neural network can only capture the dependency information by only regarding the features of nodes, and using Euclidean distance to detect their similarities. However, there are numerous structures of data that are generated from non-Euclidean domains which are represented as graphs with interrelationships and more complex dependencies between various entities. Hence, graph neural networks(GNNs) have emerged in machine learning and demonstrated a superior performance from the message passing between the nodes of graphs by representing information from their neighborhood with arbitrary depth.

Since music genres have been an increasingly important category of discriminative data, on which classification techniques are intensively explored to correctly address the music genre classification problem. In this project, the goal is to discover the knowledge graph structural relations using GNN on music genres classification problem where the input music sample could be represented in a form of vector or matrix in high dimension via spectrograms, MEL spectrogram and MFCC, etc. As preparation for GNN, we will use various algorithms to find sample(node) features, connection(edge) between each node and construct graph data. After implementation of GNN on data set and accuracy analysis, we will conduct several improvements on feature engineering including feature extractions and feature importances, and also try different model inner structures to improve the overall performance of GNN.

Dataset Description

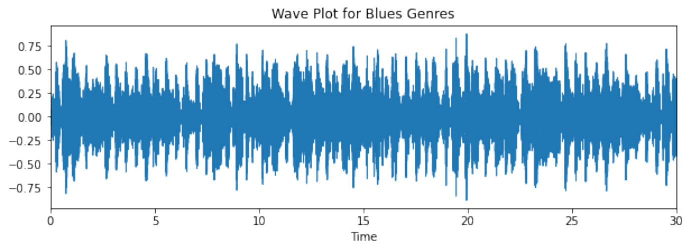
The original data is GTZAN (Tzanetakis and Cook (2002))1 in this project, collected in 2000-2001, which is a widely used dataset for evaluation in machine listening research for music genre recognition. GTZAN provides a clean distribution on a variety of audio sources and conditions in order to represent a variety of recording conditions.

GTZAN consists of two audio lengths, 3 sec and 30 sec. Each audio length contains 10 music genres. For 30 sec audio length, each genre contains 100 samples and total sample size is 1000. For 3 sec audio length, each genre contains 1000 samples and total sample size is 10000. The following figure is the histogram of 10 music genres for 3 sec audio length.



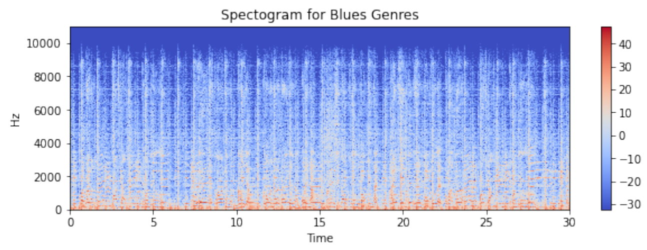
Feature Extraction

For this project, the features of audios are the physical features that refer to mathematical measurements captured from sound waveform. Waveform is formed by sample rate(frequency) and sample data(amplitude).



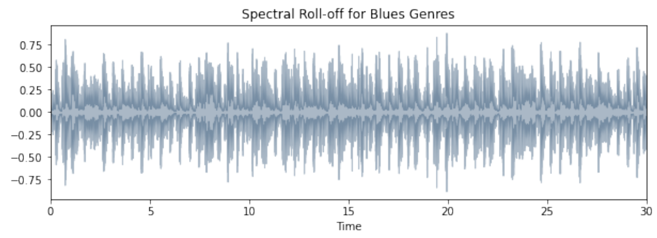
Spectrogram

Spectrogram represents the signal strength(energy) of a signal over time at various frequencies. Time runs along the horizontal axis, the frequency is represented by the vertical axis. The color is amplitude refers to energy.



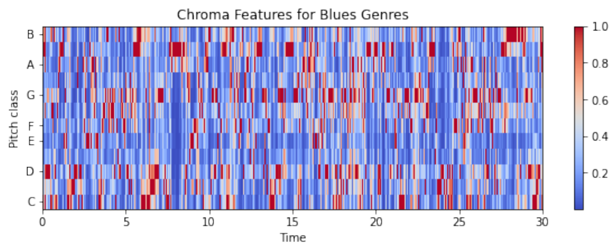
Spectral Roll-Off

Spectral Rolloff is the frequency under some specified percentage of the total spectral energy. This feature can detect the difference between the harmonic which is below the roll off and noisy sounds which is above the roll-off.



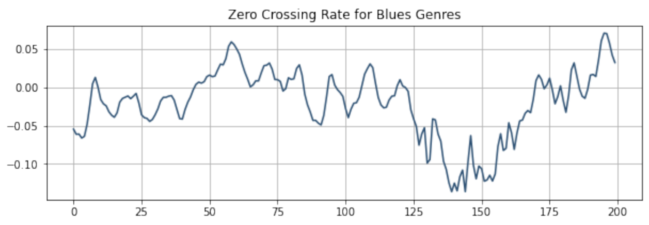
Chroma Features

Chroma Features is related to pitches using the entire spectrum to project onto 12 distinct chroma. For music genres classification, it is important to include the pitches since it can capture harmonic and melodic characteristics of music.



Zero Crossing Rate

Zero crossing is defined by the number of the speech signal changes from positive to zero to negative or inverse.



MFCCs

In sound processing, the mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency. Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC. They are derived from a type of cepstral representation of the audio clip (a nonlinear "spectrum-of-a-spectrum") [1].

ComParE2016

OpenSMILE is an open-source toolkit for audio feature extraction and classification of speech and music signals. This dataset use ComParE2016 Feature Set, which contains 6373 static features resulting from the computation of various functionals over low-level descriptor (LLD) contours.

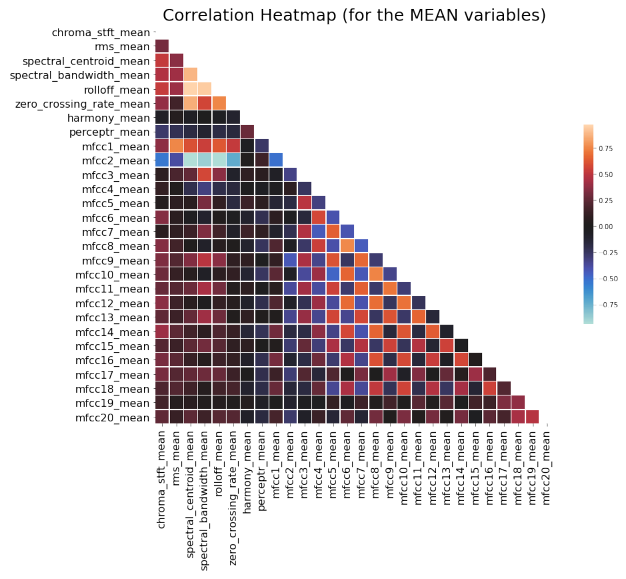
CNN

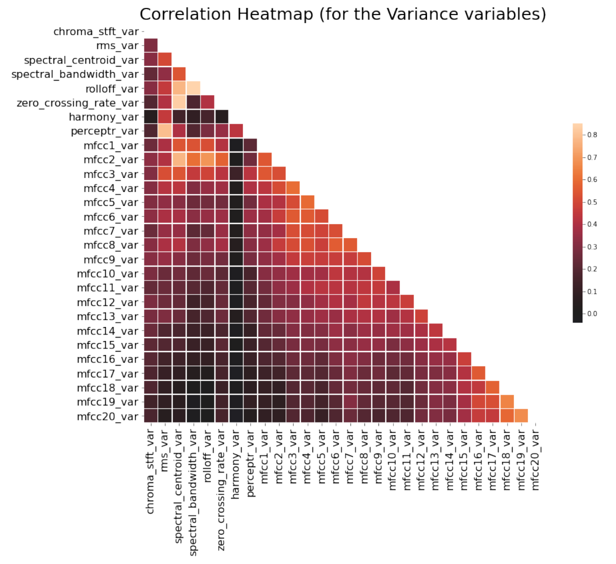
Convolutional Neural Network(CNN) is a neural network that extracts input image features. Compare to traditional feature extraction, CNN's have a strong ability to extract complex features that express the image in much more detail, learn the task specific features and are much more efficient. For this dataset, each music audio sample is translated to a image in dimension of 256×256 by MEL Spectrum, and then are translated into a vector in dimension 1×128 as the input of GNN graph preparation.
For feature extraction, we combined the features that were mentioned before and computed each feature's mean and variance to represent each music genre. Therefore, the total features we have are 57 feature sets.

Data Visualization

Correlation Heatmap

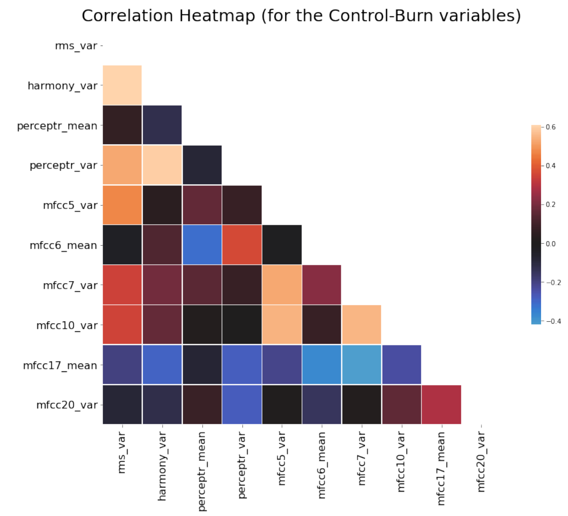
The correlations between mean variances variables having many high correlated relationships. This might influence the model overfitting issue. We will use control burn technique to get rid of it.





Control Burn

Since there 57 features by **MCFF** feature extraction for each sample, and even more for **CNN** and **COMPARE**, there might be some correlated features which reinforce these feature groups, decreasing interpretability and increasing complication and overfitting. In such a scenario, control Burn is a powerful algorithm on feature selection. Control Burn uses a weighted LASSO-based feature selection method to prune unnecessary features from tree ensembles and assigns all the feature importance of a correlated group of features to a single feature[2]. To illustrate, for **MCFF** feature extraction, we implement Control Burn on the 57 features to prune unnecessary and correlated features, and keep only 10 features as final features. Control Burn has demonstrated that it can largely improve our classification accuracy. The validation set accuracy without Control Burn is 63% while validation set accuracy after Control Burn is 72%.



Graph Neural Network Model

GNN

Before get into **Graph neural network(GNN)**, first we need understand what is **graph**. A graph is a data structure consisting of two components: vertices and edges. A graph G can be well defined by the set of vertices V and edges E as following formula.

$$G = (V, E)$$

GNN is a deep learning method that works in the graph domain, providing an easy way to do node-level, edge-level, and graph-level prediction tasks. GNN is used in this data set to model music features interaction and connection as knowledge graphs. In our case, we are trying to use GNN to apply a node classification problem.

For illustration purposes of classification problem, this approach may be expressed as:

$$\mathbf{h}_v = f(\mathbf{x}_v, \mathbf{x}_{co[v]}, \mathbf{h}_{ne[v]}, \mathbf{x}_{ne[v]})$$

Each node v is characterized by its features x_v, and ground trues label t_v. In our case, x_v is the feature space of PCA features in two dimensions. $x_{co[v]}$ denotes the features of the edges connecting with v, and the embedding of the neighboring nodes of v is denoted by $x_{ne[v]}$. In our case, the neighboring nodes are calculated by KNN algorithms. The model digest the information from the given labels graph G to predict the unlabeled nodes by finding the best solution of function f which is denoted as h_v(weights).

Construct Graph

Since GNN takes a knowledgeable graph as input, we need to first construct the graph data for GNN. Suppose each sample serves as a node, we need to know the edge between each node, in the other words, how those nodes are connected to each other, whether they are connected and how strong is the connection. Unlike other data such as matrix and vector, since graph data works in the non-euclidean space, we need algorithms to compute the edge between each pair of nodes. We implement several steps to compute the edge between nodes.

PCA

First, we use Principal Component Analysis(PCA) to reduce each feature vector in dimension of $1 \times m$ to a vector in dimension of $1 \times n$, with $n \le m$, which enables nodes to be represented in a lower dimension space.

KNN

Second, after PCA, for each node, use k-nearest neighbors to find k nearest neighbours of each node, and corresponding distance for its each neighbour, hence, this node is connected to these k neighbours, with corresponding distance. In distance computation, since different distance function might be intended for specific kind of data, we implemented different functions to calculate distance metric, such as Euclidean, Manhattan, Matching, Jaccard, etc. Repeat this KNN step to find node and edge connection for each music node.

Adjacency Matrix

Finally, introduce Adjacency Matrix to represent these edge and node connection in a knowledge graph as the input of GNN. Adjacency Matrix is a square matrix used to represent a finite graph. The elements of the matrix indicate whether pairs of vertices are adjacent or not in the graph. In this case, if two music nodes are connected, the element in the Adjacency Matrix is the corresponding distance between this pair of nodes.

$$A_{i,j} = distance_{i,j} \qquad \text{if node i, j is connected} \tag{1}$$

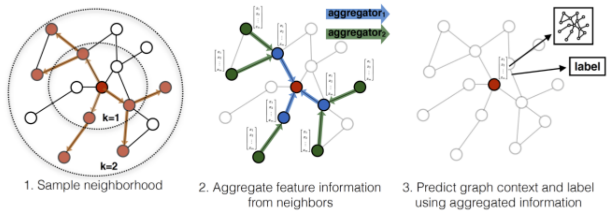
$$A_{i,j} = 0 \qquad \text{if node i, j is not connected} \tag{2}$$

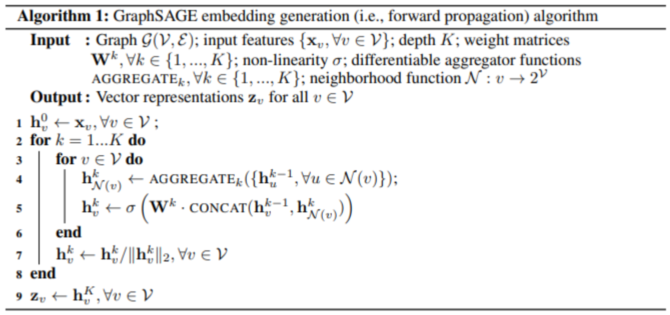
$$A_{1000,1000} = \begin{pmatrix} 1 & dist_{1,2} & \cdots & dist_{1,1000} \\ dist_{2,1} & 1 & \cdots & dist_{2,1000} \\ \vdots & \vdots & \ddots & \vdots \\ dist_{1000,1} & dist_{1000,2} & \cdots & 1 \end{pmatrix}$$

Then, the last preparation is to convert the Adjacency matrix to network using tools that will be used for input into GraphSage.

GraphSage Model

To avoid the computationally expensive modifications in an inductive setting which is requiring additional rounds of gradient descent before new predictions can be made, we conduct GraphSage model. Instead of training a distinct embedding vector for each node, we train a set of aggregator functions that learn to aggregate feature information from a node's local neighborhood[3].


k=1
k=2
1. Sample neighborhood
aggregator₁
aggregator₂
2. Aggregate feature information from neighbors
label
3. Predict graph context and label using aggregated information



Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

```
Input  : Graph $\mathcal{G}(\mathcal{V},\mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth $K$; weight matrices
         $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity $\sigma$; differentiable aggregator functions
         $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \to 2^{\mathcal{V}}$
Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$
1 $\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$ ;
2 for $k = 1...K$ do
3     for $v \in \mathcal{V}$ do
4         $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
5         $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
6     end
7     $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
8 end
9 $\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$
```

The input consists of the knowledge graph G we constructed, the features for all nodes x_v, adjacency matrix as initial weight matrices W^k including the KNN's embedding nodes information and relationship(distance) between neighboring nodes as the base case.

GraphSage model aggregates information from node neighbors in K iterations. For each iteration, the algorithm will update each node $v \in V$ aggregates the representations of the nodes in its immediate neighborhood $\{h_u^{k-1}, \forall$ u$\in N(v)\}$ into a single vector $h_{N(v)}^{k-1}$. Note that this aggregation step depends on the representations generated at the previous iteration of the outer loop (i.e., k-1), and the k=0 ("base case") representations are defined as the input node features. After aggregating the neighboring feature vectors, GraphSage then concatenates the node's current representation, h_u^{k-1} , with the aggregated neighborhood vector, $h_{N(v)}^{k-1}$, and this concatenated vector is fed through a fully connected layer with nonlinear activation function , which transforms the representations to be used at the next step of the algorithm (i.e., $h_u^k, v \in V$). As the updating process iterates, nodes aggregate more and more information from theri local neighbors when further reaches of the graph. Finally in the last itertaion K, the algorithm will assign the depth h_u^K as output z_v[3].

The aggregator we used for this project is the mean aggregator which takes the average of the latent vectors of the node and its related neighborhood, by using mean aggregator, it contains all the information of its neighboring nodes.

Model Evaluation

Model results have two stages, test period and improvement period.

For test period, we first only explore two music genres classification(Blues and Jazz) to initialize the construction of GNN model. This stage provides a fundamental basis for understanding training process and how to pruning the model.

Based on test period, model and result before midterm report, we improved our model in various aspects, including Replacing PCA dimension reduction by TSNE, embedding edge distance, instead of 1, with distance between node, adding more features by different feature selection and subsets, changing Cosine Similarity to other functions in KNN calculation, adding more genres, etc. Model evaluation results are summarized in the following.

Loss Function

For classification problem, we used cross entropy as loss function which calculated the scores that summarized the average difference between the true label and predicted probability distributions for predicting different classes.

Cross Validation

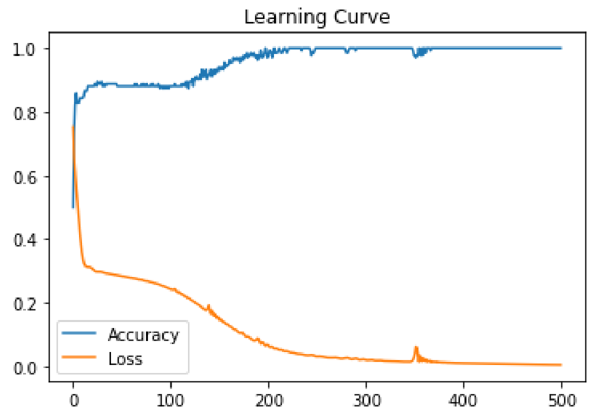
We design 10 times cross validation to get rid of the influence of initialization of model and have more robust results.

★ Test Period – Binary Classification

Data
200 music samples of 30 seconds, 58 MFCCs features of each sample

Parameters
•Feature Selection: Using PCA and ControlBurn
• Number of Neighbours: 80
• Distance Metric: Euclidean distance
• Model: Graph Sage Model
  * Hidden Layer: 1
  * Activation Function: Relu
  * Neours for hidden Layer: 80
  * Learning Rate: 0.001
• Cross Validation: 10

Accuracy
• Train Accuracy: 1
• Test Accuracy
  * without ControlBurn:0.63
  * with ControlBurn:0.72

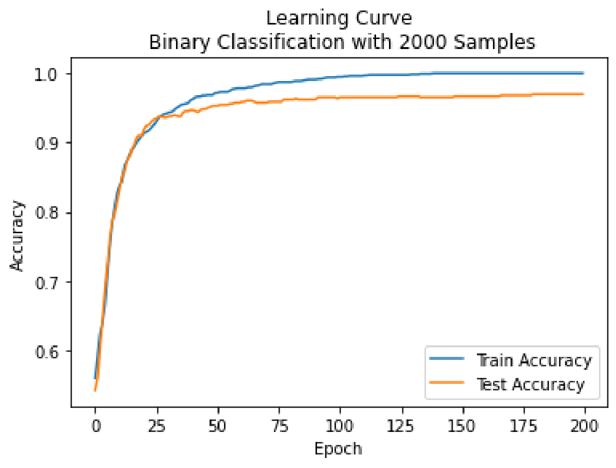


★ Improvement Period – Binary Classification More Samples

Data
Binary Classification: Binary Classification: 2000 music samples of 3 seconds, 58 MCFF features of each sample

Parameters
•Feature Selection: without PCA or ControlBurn
• Number of Neighbours: 800
• Distance Metric: jaccard
• Model: Graph Sage Model
  * Hidden Layer: 1
  * Activation Function: Relu
  * Neours for hidden Layer: 80
  * Learning Rate: 0.001
• Cross Validation: 10

Accuracy
• Train Accuracy: 1
• Test Accuracy: 0.9698



★Improvement Period – Multi-classification More Labels

Data

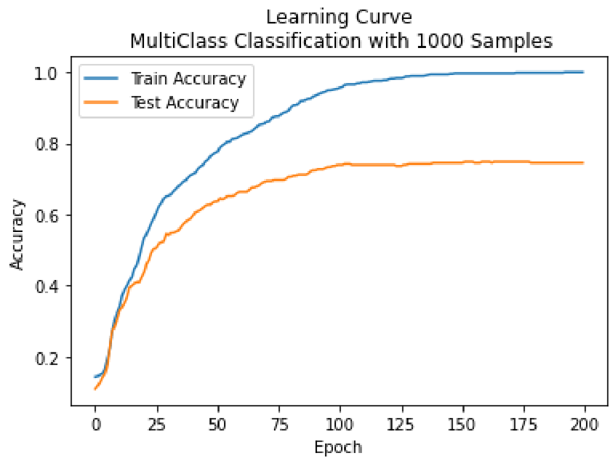
Multi-classification: 10 genres, 1000 music samples of 30 seconds, 57 features of each sample

Parameters

- Feature Selection: without PCA or ControlBurn
- Number of Neighbours: 100
- Distance Metric: jaccard
- Model: Graph Sage Model
    - Hidden Layer: 1
    - Activation Function: Relu
    - Neours for hidden Layer: 150
    - Learning Rate: 0.001
- Cross Validation: 10

Accuracy

- Train Accuracy: 1
- Test Accuracy: 0.7203030303030303



★Improvement Period – Multi-classification More Features

Data

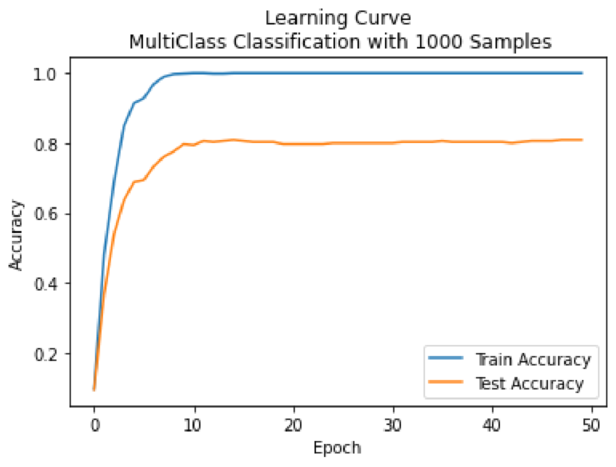
Multi-classification: 10 genres, 1000 music samples of 30 seconds, 6431 features of each sample

Parameters

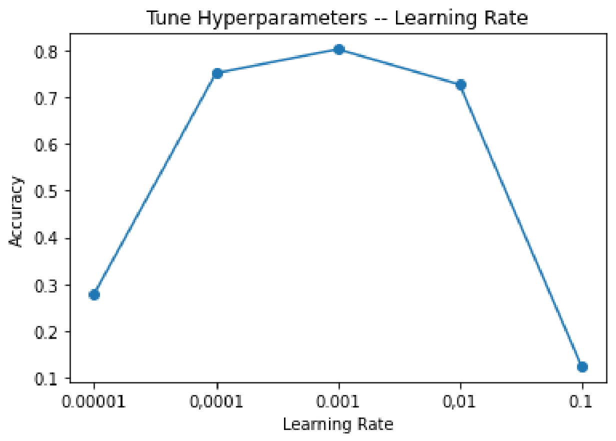
- Feature Selection: without PCA or ControlBurn
- Number of Neighbours: 100
- Distance Metric: jaccard
- Model: Graph Sage Model
    - Hidden Layer: 1
    - Activation Function: Relu
    - Neours for hidden Layer: 150
    - Learning Rate: 0.001
- Cross Validation: 10

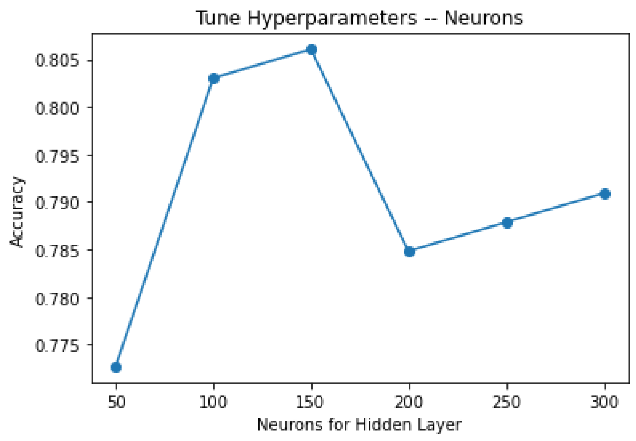
Accuracy

- Train Accuracy: 1
- Test Accuracy: 0.7976



In this model setting, we did hyper-parameter tuning for neurons of hidden layer and learning rate. The following figures shows neurons is 150 and learning rate 0.001 are the best value of this model.





Conclusion

Using knowledge graph structural relations to feed in Graph Neural Network for music genres classification problem not only was beneficial from the construction of deep learning model which can learn more details information of complex audio extracted features but also captured the graph dependency information from spectrogram based features representation. Overall, the model for binary classification and multi-classification both give robust results. GNN model is not limited by the dependencies of features complexity and it can digested them by learning iterates.

References

[1] Min Xu; et al. (2004). "HMM-based audio keyword generation" (PDF). In Kiyoharu Aizawa; Yuichi Nakamura; Shin'ichi Satoh (eds.). Advances in Multimedia Information Processing – PCM 2004: 5th Pacific Rim Conference on Multimedia. Springer. ISBN 978-3-540-23985-7. Archived from the original (PDF) on 2007-05-10.

[2] Liu, Brian and Xie, Miaolan and Udell, Madeleine. ControlBurn. Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. Aug 2021.

[3] Hamilton, Ying, and Leskovec, Inductive Representation Learning on Large Graphs,2017

[4] Laurens van der Maaten, Geoffrey Hinton. Visualizing Data using t-SNE.Journal of Machine Learning Research 9 (2008) 2579-2605. 11/08.