Cornell University

Graph Neural Network for Music Genre Recognition

Shuo Feng sf587

Introduction

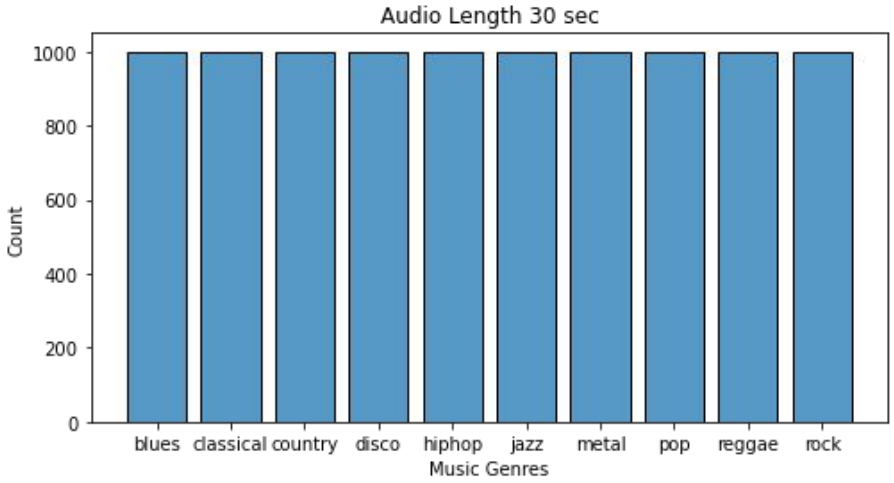
Cornell University

- Big messy data is well trained by deep learning in many different fields like natural language processing, computer vision, and speech recognition. Standard neural network usually captures the dependency information by Euclidean distance to detect data similarities. While graph neural networks(GNNs) can demonstrate a superior performance from the message passing between the nodes of graphs by representing data information as a graph.
- This project aims to discover the knowledge graph structural relations using GNNs on music genres classification problem.
- Steps:
  - ➢ Represent input music sample as a vector or matrix in high dimension via various feature extraction
  - ➢ Construct sample(node) features, connection(edge) and graph data
  - ➢ Implement GNN on data set and improve performance

Dataset

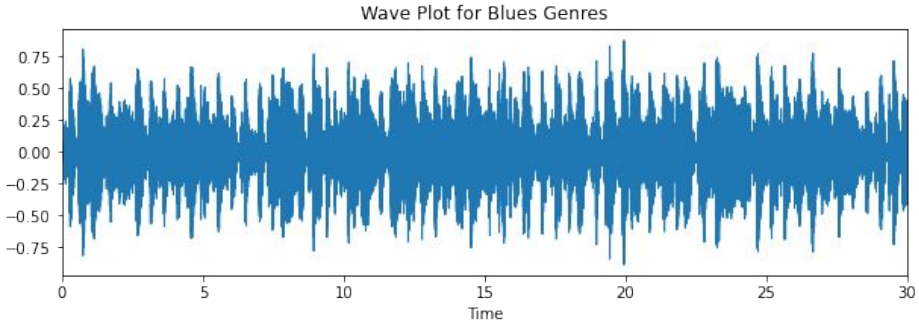
GTZAN

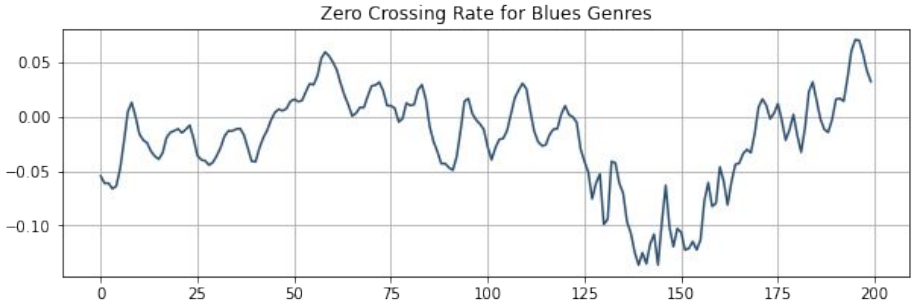
- Music Genre Recognition Dataset
- Classes: 10 Genres
- Audio length: 3 sec / 30 sec
- Sample Size
  - ➢ for 30 sec: 1000 Samples
  - ➢ for 3 sec: 10000 Samples

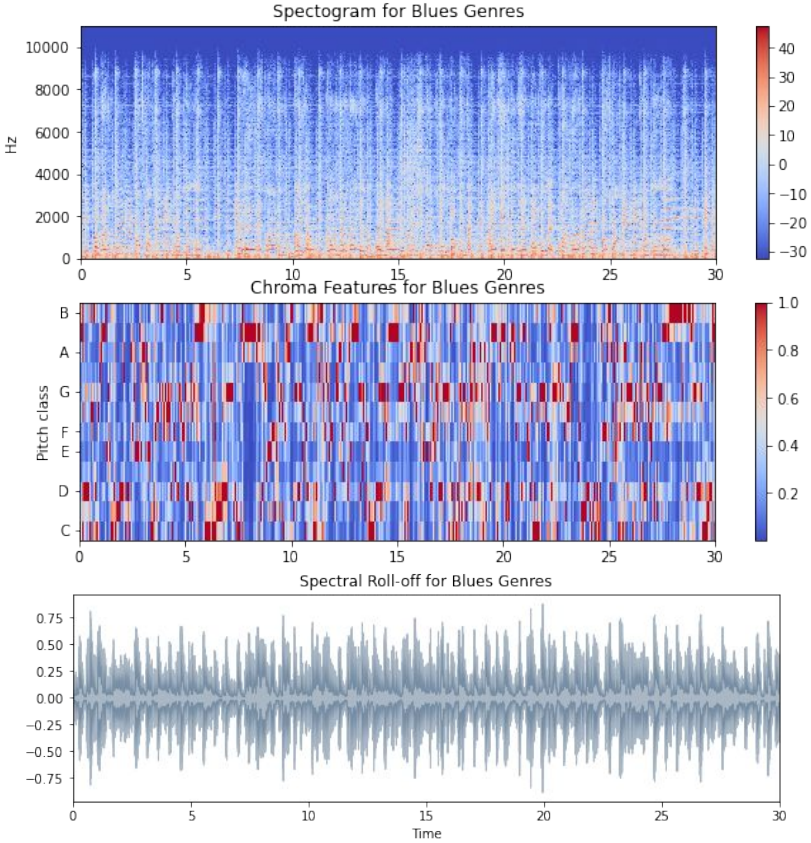


Audio Feature Extraction

Cornell University







Audio Feature Extraction

Cornell University

MFCC

- Mel- frequency cepstral coefficients (MFCCs) are coefficients that collectively make up 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. For feature extraction, computed each feature's mean and variance to represent each music genre with 40 features

OpenSMILE: ComParE_2016 Feature Set

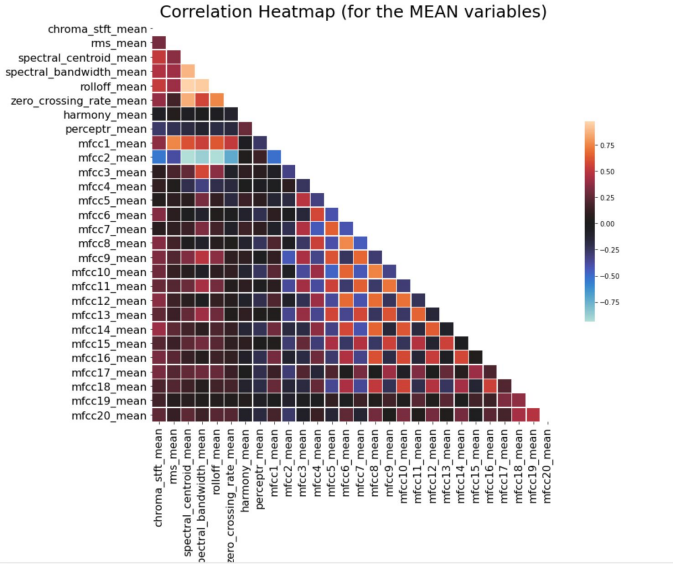
- The COMPARE feature set contains 6373 static features resulting from the computation of various functionals over **low-level descriptor (LLD)** contours.

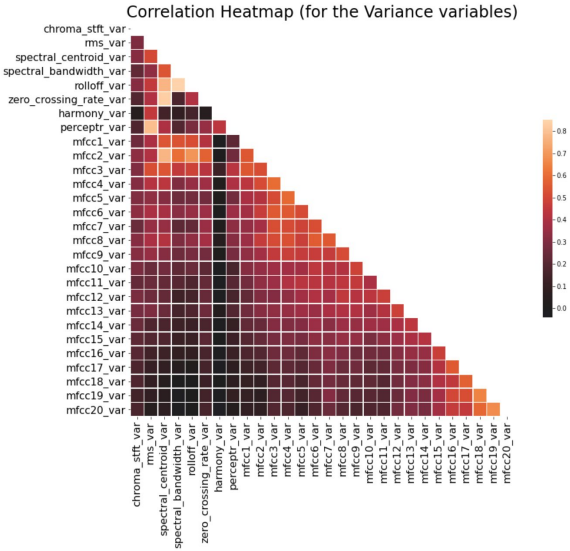
Data Visualization

Cornell University

Correlation Heatmap:

- Visualize the correlations between mean variances variables
- Correlation Heatmap shows many high correlated relationships which might influence the model overfitting issue. Therefore, need to apply feature selection technique.



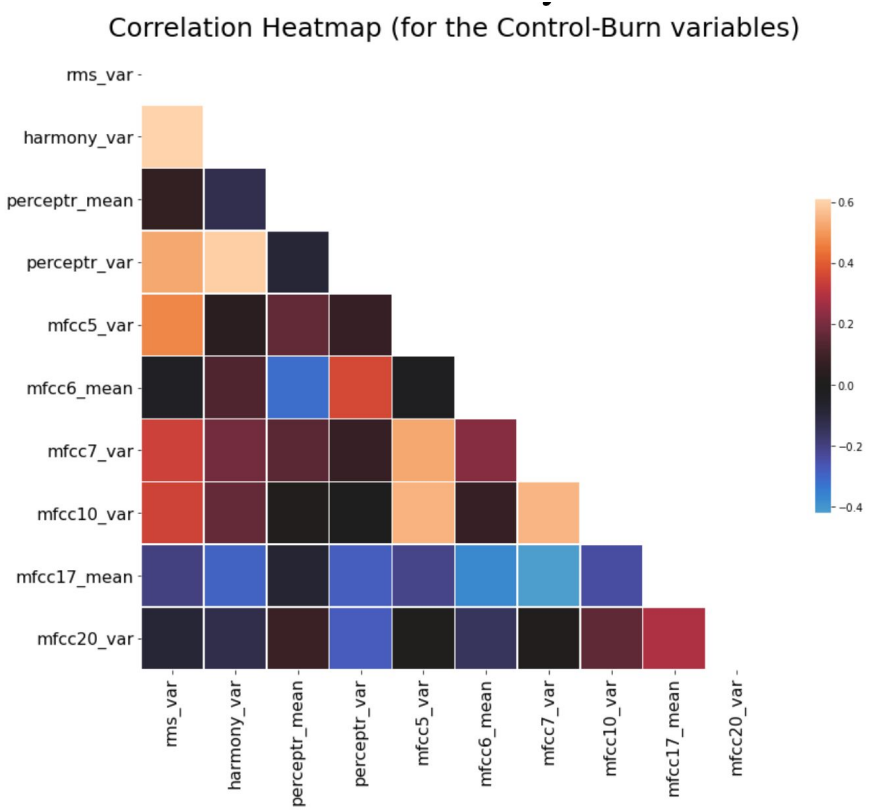


Feature Selection

Cornell University

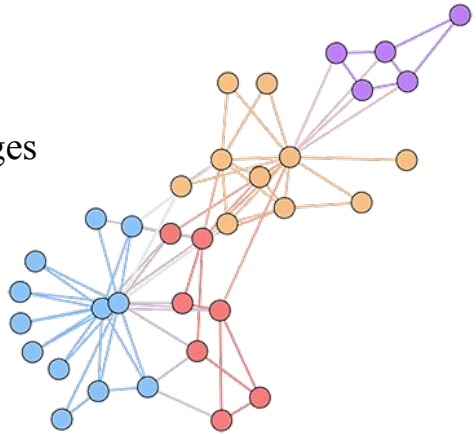
Control Burn

- Since there are 57 features for each sample, where correlated features might reinforce these feature groups, decrease interpretability and increase complication and overfitting.
- Control Burn is a powerful algorithm on feature selection, using a weighted LASSO-based feature selection method to prune. We implement Control Burn to prune unnecessary and correlated features and keep only 10 features as final features.



GNN Model

Cornell University



Graph

- A graph is a data structure consisting of two components: Vertices and Edges
- Graph representation: $G = (V, E)$

Graph Neural Network(GNN)

- Deep learning method that works in the graph domain
- Node embeddings: a way of representing nodes as vectors
- Applications: Node Classification, Link Prediction, Network Analysis
- Node classification:

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

- Every node **v** is associated with a ground-truth label **t_v**, characterized by its features **x_v**
- $\boldsymbol{x}_{co[v]}$: features of the edges connecting with *v*
- $\boldsymbol{h}_{ne[v]}$: embedding of the neighboring nodes of *v*
- $\boldsymbol{x}_{ne[v]}$: denotes the features of the neighboring nodes of *v*
- Learning Process: Given a partially labeled graph *G*, the goal is to leverage these labeled nodes to predict the labels of the unlabeled nodes. It learns to represent each node with a *d* dimensional vector *h_v* which contains the information of its neighborhood.

GNN Model

Cornell University

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 information between each node. Unlike other data such as matrix and vector, since graph data works in the non-euclidean space, we need several algorithms by step to compute the edge between each pair of nodes.
- Framework:
  - ➢ Dimensional Reduction: PCA

    First, use Principal Component Analysis(PCA) to reduce each feature vector in dimension of 1×57 to a vector in dimension of $1 \times n$, which enables nodes to be represented in a lower dimension space

  - ➢ In each feature space compute 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. We implemented different functions to calculate distance metric. Repeat this KNN step to find node and edge connection for each music node.

GNN Model

Cornell University

Construct Graph

- Framework:
  - ➢ Construct 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.

$$A_{i,j} = distance_{i,j} \quad \text{if i,j is connected}$$

$$A_{i,j} = 0 \quad \text{if i, j is not connected}$$

    Then, the last preparation is to convert the Adjacency Matrix to network using networkx

Graph Sage Model

Cornell University

GraphSAGE is an **inductive** framework that leverages node attribute information to efficiently generate representations on previously unseen data.

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 σ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$; Set the initial node embedding to be the node's feature vector

for $k = 1...K$ **do** Iterate over all the k-hop neighbourhoods

  for $v \in \mathcal{V}$ **do** Iterate through all the nodes in the graph

    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;

    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$ Run the aggregate-update cycle for every node in this k-th iteration

  end

  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$ Normalize the node embedding

end

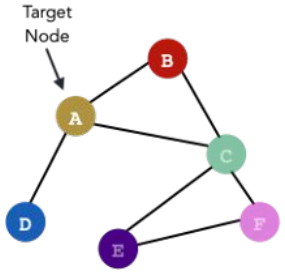
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

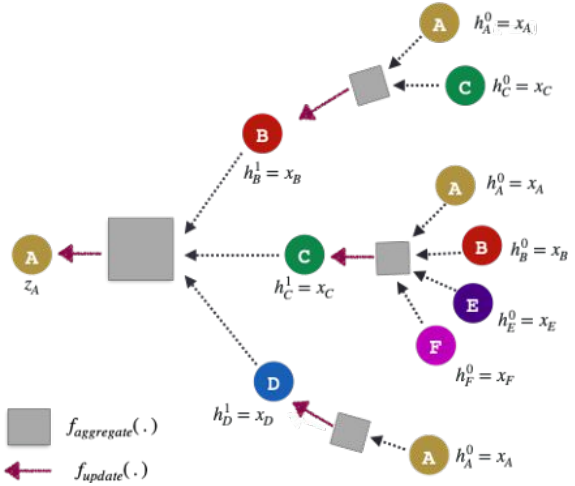
Source: https://medium.com/analytics-vidhya/ohmygraphs-graphsage-and-inductive-representation-learning-ea26d2835331

Graph Sage Model

Aggregation for updating node representations:

- Each node v is updated by aggregated representation based on its neighbourhood aggregated representations and node v's previous representation.
- Advantage: learning aggregator functions to generate node embeddings, instead of learning the embeddings themselves, is inductivity. When the aggregator weights are learned, the embedding of unseen node can be generated from its features and neighborhood.
- Mean aggregators: Takes the average of the latent vectors of a node and all its neighborhood.





Model Evaluation

Cornell University

- Test Result Before Midterm
  - ➢ Binary Classification: 200 music samples of 30 seconds,58 MCFF features of each sample

- Improvement Results After Midterm
  - ➢ Binary Classification: 2000 music samples of 3 seconds,58 MCFF features of each sample
  - ➢ Multi-classification: 10 genres, 1000 music samples of 30 seconds, 58 MCFF features of each sample
  - ➢ Multi-classification: 10 genres, 1000 music samples of 30 seconds, 6431 features of each sample

- Methods for improvement:
  - ➢ Add more samples
  - ➢ Add more classes
  - ➢ Use different distance metric in KNN
  - ➢ Add feature Set
  - ➢ Tune Hyperparameters

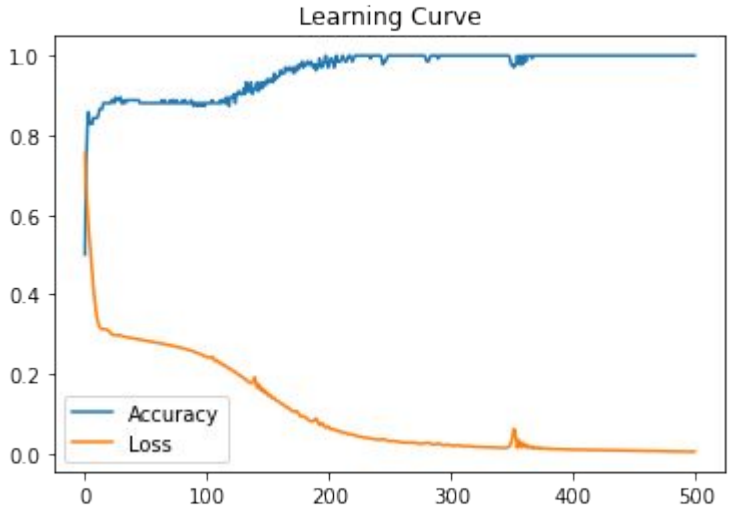
Test Result -- Binary Classification

Cornell University

- **Model Evaluation:** Binary Classification: 200 music samples of 30 seconds, 58 MCFF 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: 0.63 without ControlBurn/0.72 with ControlBurn

Improved Result -- Binary Classification

Cornell University

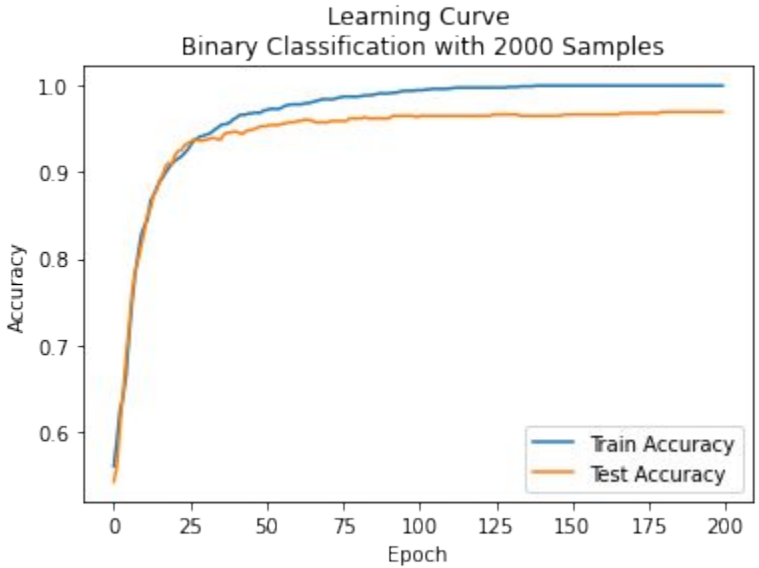
- **Model Evaluation:** Binary Classification: 2000 music samples of 3 seconds, 58 MCFF features of each sample

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

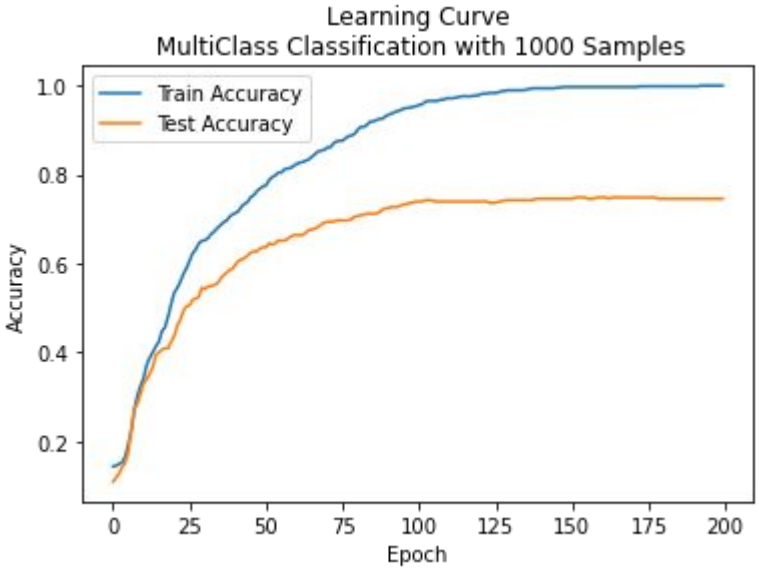


Improved Result -- Multi-classification

- **Model Evaluation:** 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

Improved Result -- More Features

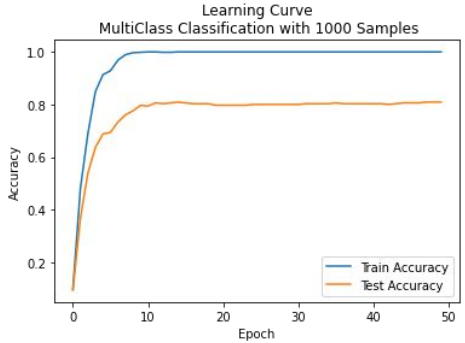
- **Model Evaluation:** Multi-classification: 10 genres, 1000 music samples of 30 seconds, 6431 features of each sample

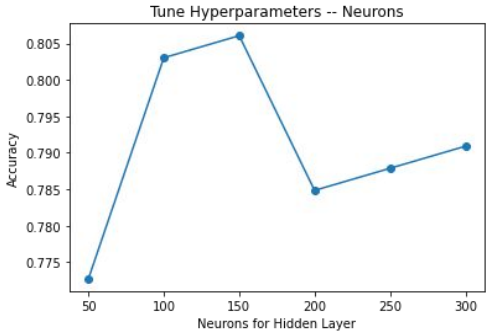
Parameters:

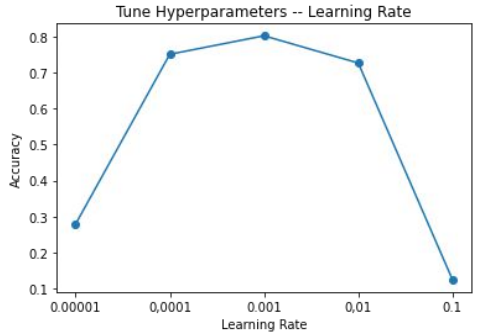
- ➢ Feature Selection: without PCA or ControlBurn
- ➢ Number of Neighbours: 100
- ➢ Distance Metric: jaccard
- ➢ Model: Graph Sage Model
  - ➢ Hidden Layer: 1
  - ➢ Activation Function: Relu
  - ➢ Neours for hidden Layer: 150
  - ➢ Learning Rate: 0.001
- ➢ Cross Validation: 10

Accuracy

- ➢ Train Accuracy: 1
- ➢ Test Accuracy: 0.7976







Cornell University

Thank You

Contact Information:
sf587@cornell.edu