Music Genre Recognition

Under the supervision of Prof. Guoliang Xue

CSE 575: Statistical Machine Learning

Computing, Informatics, and Decision Systems Engineering

Arizona State University

Presented By Group 12:

Abantika Basak (1217117655)

Abhilasha Mandal (1217160477)

Ajinkya Dande (1217613956)

Omkar Muglikar (1217158150)

Rounak Sengupta (1215043206)

Outline

- Background
  - Music Information Retrieval
  - Need for Automatic music genre classification
- Overall flow of the proposed systems
- Dataset used
- Classifiers
  - Neural Networks: CNN, CNN-LSTM, BBNN
  - SVM
  - Logistic Regression
- Contributions and Results
- Comparison with existing work and Future scope

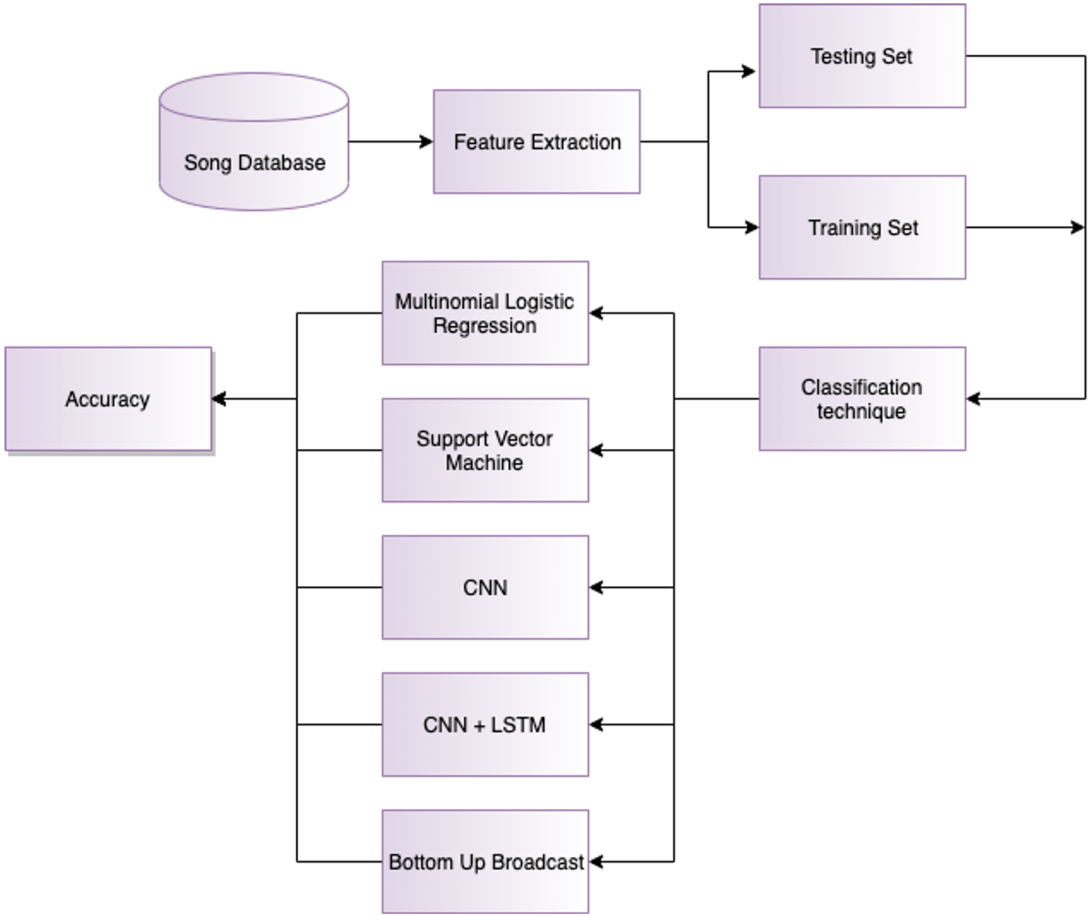
Music Information Retrieval

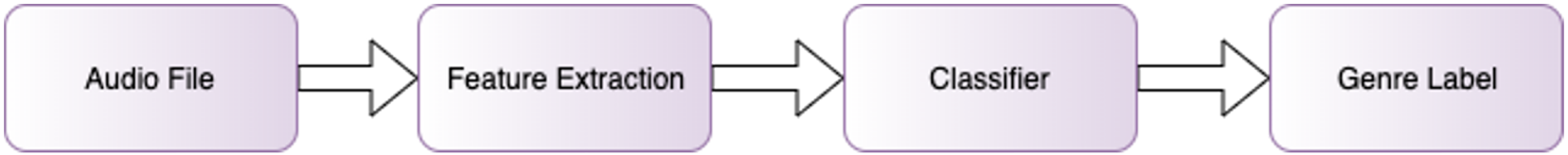
- Extracting meaningful information from songs
- Applications
  - Copyright Monitoring
  - Music transcript
  - Music management
    - Find performer/artist
    - Track separation
- Digital Music platform
  - Identify song mood
  - Music recommendation system
  - Identify songs from audio (Shazam, Spotify)

Current Practices

- Current practices rely on song file metadata and user input/feedback
  - Unscalable, tedious, highly subjective
  - More than 75000 albums released every year in USA alone
- Therefore, an automatic technique for genre classification is required

Proposed Architecture





Application

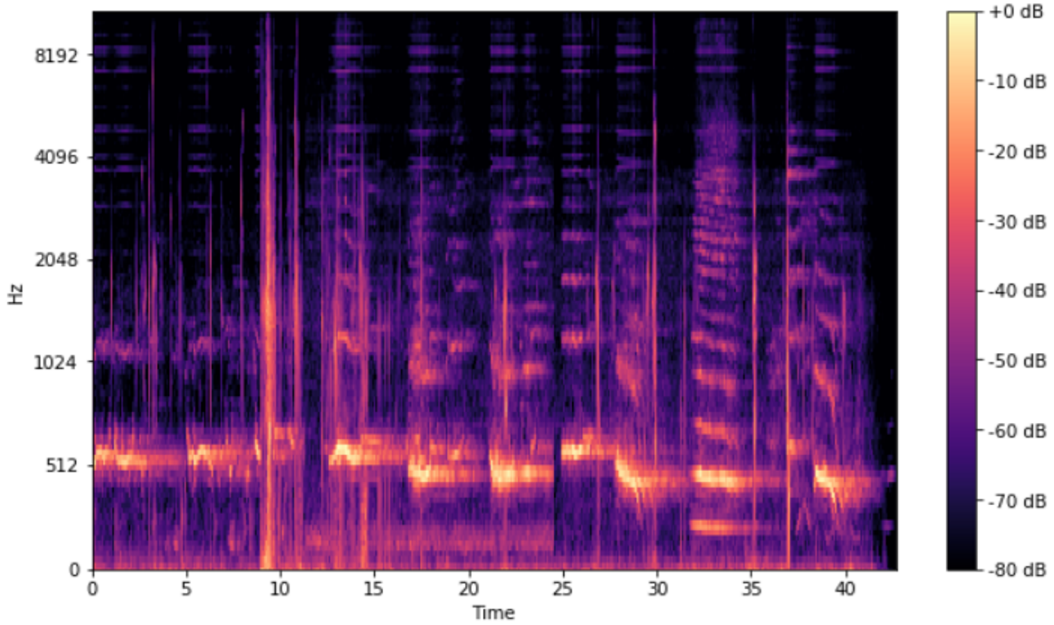
Song Dataset

- GTZAN Dataset
  - 1000 Audio files in .au format
  - 16 bit, 22050 Hz, 30 secs each
- {Blues, Classical, Country, Disco, Hip-Hop, Jazz, Metal, Pop, Reggae, Rock}

Generation of Mel Spectrograms

- The mel-spectrogram images are generated for every audio file
- Normalised the pixel values
- Fed these images to the CNN
- Image of a song is given in next slide

- **Mel spectrogram**



ML models used and the test accuracies achieved

- CNN-LSTM - 48%
- SVM - 60 %
- CNN - 76 %
- Bottom up broadcast - 88%
- Multinomial Logistic regression - 89 %

A few other RNN models were tried but were inaccurate.

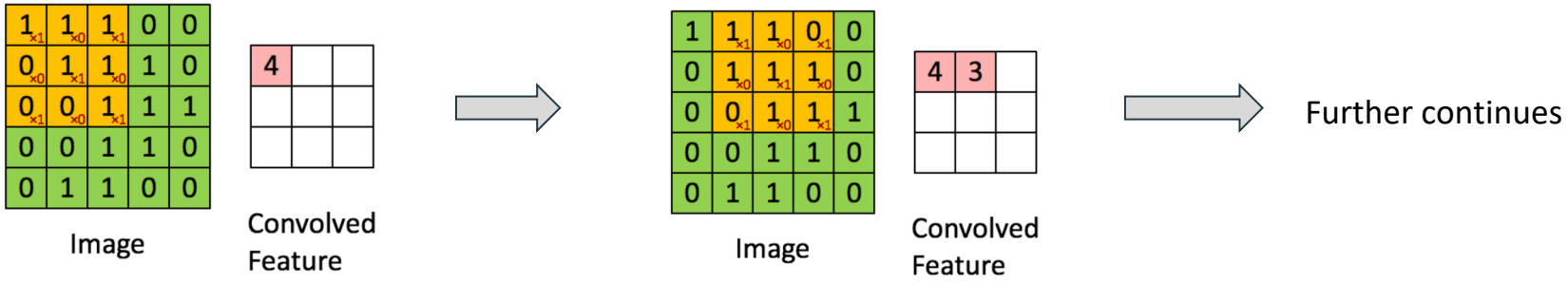
Convolutional Neural Network

- Deep Learning Algorithm
- A basic question : Why ConvNets over Multi Layer Perceptron ?
- Answer -
  - In MLP, total number of parameters - very high, ConvNets allow parameter sharing, weight sharing
  - Preprocessing in ConvNets - much lower than in MLP
  - CNNs capture local connectivity
  - Spatial information is lost in MLP

Convolutional Neural Network

- Role of ConvNet - reduction of sizes of images such that they are easier to process, without losing important features
- 3 types of layers :
  - Convolution Layer
  - Pooling Layer
  - Fully Connected Layer

Convolution Layer



Pooling Layer (Max Pooling, Average Pooling)

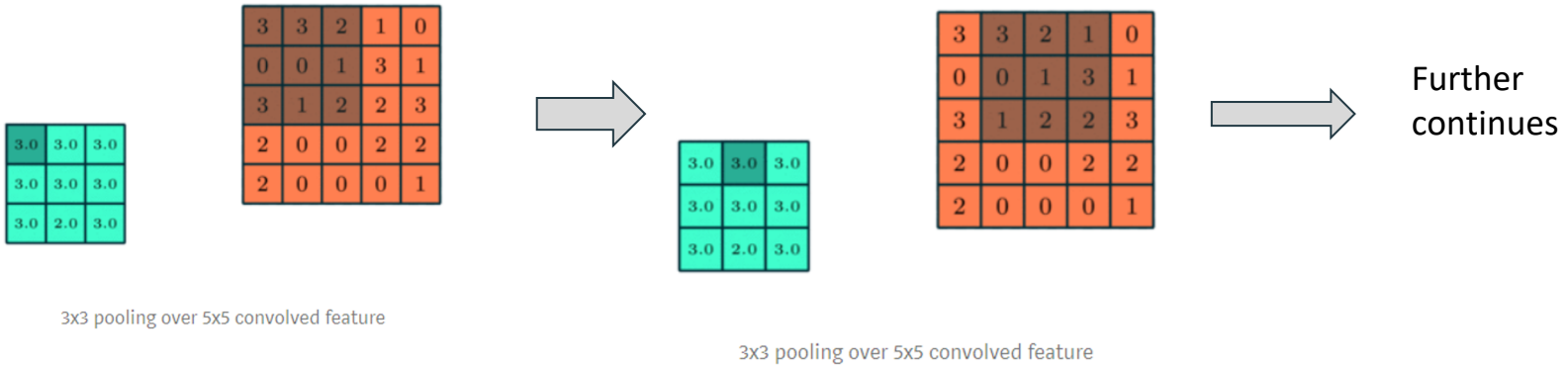


Fig : Max Pooling

Fully Connected Layer (as in MLP)

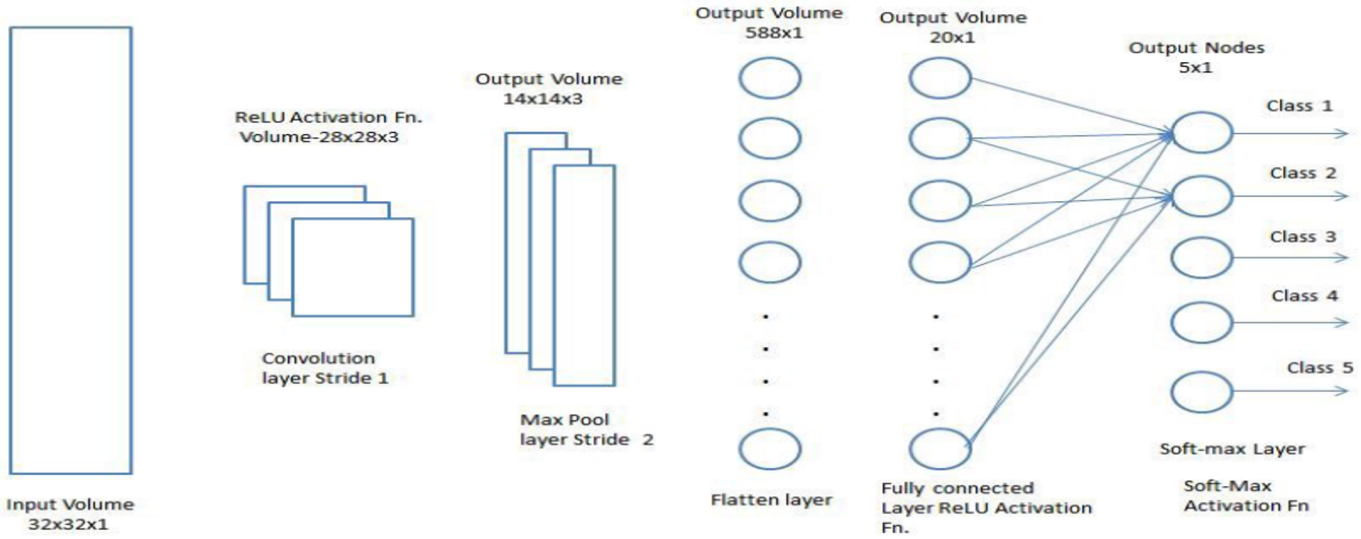


Fig : Basic Architecture of CNN

conv2d_61 (Conv2D)	(None, 480, 640, 32)	896
conv2d_62 (Conv2D)	(None, 478, 638, 32)	9248
average_pooling2d_31 (Averag	(None, 239, 319, 32)	0
dropout_41 (Dropout)	(None, 239, 319, 32)	0
conv2d_63 (Conv2D)	(None, 120, 160, 64)	32832
conv2d_64 (Conv2D)	(None, 59, 79, 64)	65600
average_pooling2d_32 (Averag	(None, 29, 39, 64)	0
dropout_42 (Dropout)	(None, 29, 39, 64)	0
conv2d_65 (Conv2D)	(None, 29, 39, 64)	65600
conv2d_66 (Conv2D)	(None, 26, 36, 64)	65600
average_pooling2d_33 (Averag	(None, 13, 18, 64)	0
dropout_43 (Dropout)	(None, 13, 18, 64)	0
flatten_11 (Flatten)	(None, 14976)	0
dense_35 (Dense)	(None, 1000)	14977000
dense_36 (Dense)	(None, 512)	512512
dense_37 (Dense)	(None, 64)	32832
dense_38 (Dense)	(None, 10)	650

Total params: 15,762,770
Trainable params: 15,762,770
Non-trainable params: 0

```
test_eval = model.evaluate(X2, Y3, verbose=0)
print('Test loss:', test_eval[0])
print('Test accuracy:', test_eval[1])
```

```
Test loss: 1.7577875727415084
Test accuracy: 0.76
```

- Trained for 100 epochs
- Training set - 900 images
- Validation Set - 100 images
- Categorical Cross Entropy used as loss function

Confusion Matrix

```
[[ 9  0  0  0  0  0  0  0  1  0]
 [ 0  6  0  0  0  2  1  0  1  0]
 [ 0  0  9  0  0  0  0  0  1  0]
 [ 0  0  0  6  0  2  0  0  1  1]
 [ 0  0  0  0  8  0  2  0  0  0]
 [ 0  0  0  1  0  9  0  0  0  0]
 [ 0  1  1  0  1  0  7  0  0  0]
 [ 2  1  0  0  0  0  0  6  1  0]
 [ 0  1  1  0  1  0  1  0  6  0]
 [ 0  0  0  0  0  0  0  0  0 10]]
```

	precision	recall	f1-score	support
Class 0	0.82	0.90	0.86	10
Class 1	0.67	0.60	0.63	10
Class 2	0.82	0.90	0.86	10
Class 3	0.86	0.60	0.71	10
Class 4	0.80	0.80	0.80	10
Class 5	0.69	0.90	0.78	10
Class 6	0.64	0.70	0.67	10
Class 7	1.00	0.60	0.75	10
Class 8	0.55	0.60	0.57	10
Class 9	0.91	1.00	0.95	10
accuracy			0.76	100
macro avg	0.77	0.76	0.76	100
weighted avg	0.77	0.76	0.76	100

Convolutional Recurrent Neural Network

- Stands for Convolutional Neural Network - Long Short Term Memory
- Deep Learning Algorithm
- Why use both CNN and RNN?
  - CNN - makes sense - each spectrogram is an image
  - RNN - excel in understanding sequential data (Here, we use CNN-LSTM as LSTM captures long-term dependencies)

CNN-LSTM Architecture (next slide)

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	(None, 480, 640, 3)	0
conv2d_7 (Conv2D)	(None, 480, 640, 32)	896
conv2d_8 (Conv2D)	(None, 478, 638, 32)	9248
average_pooling2d_4 (Average	(None, 239, 319, 32)	0
dropout_6 (Dropout)	(None, 239, 319, 32)	0
conv2d_9 (Conv2D)	(None, 239, 319, 64)	32832
conv2d_10 (Conv2D)	(None, 236, 316, 64)	65600
average_pooling2d_5 (Average	(None, 118, 158, 64)	0
dropout_7 (Dropout)	(None, 118, 158, 64)	0
conv2d_11 (Conv2D)	(None, 118, 158, 64)	65600
conv2d_12 (Conv2D)	(None, 115, 155, 64)	65600
average_pooling2d_6 (Average	(None, 57, 77, 64)	0
dropout_8 (Dropout)	(None, 57, 77, 64)	0
time_distributed_2 (TimeDist	(None, 57, 4928)	0
lstm_3 (LSTM)	(None, 57, 1000)	23716000
lstm_4 (LSTM)	(None, 500)	3002000
dense_4 (Dense)	(None, 100)	50100
dropout_9 (Dropout)	(None, 100)	0
dense_5 (Dense)	(None, 50)	5050
dropout_10 (Dropout)	(None, 50)	0
dense_6 (Dense)	(None, 10)	510

- Trained for 200 epochs
- Training set - 900 images
- Validation Set - 100 images
- Categorical Cross Entropy used as loss function
- Test Accuracy : 48 %

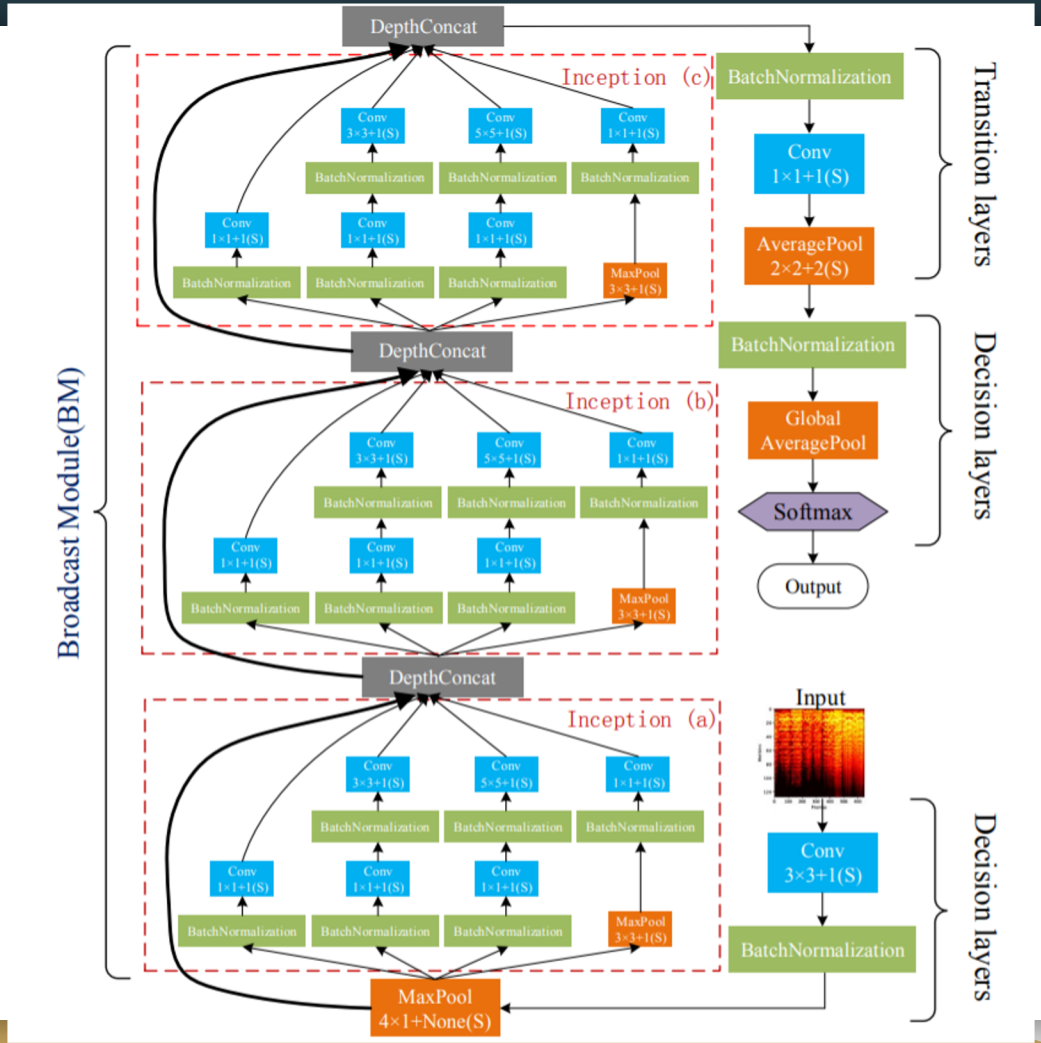
Bottom-up Broadcast Neural network

- Input is an image of mel-spectrogram (288 X 432 X 3)
- It is an architecture of CNN
- Trained for 100 epochs (training set 900 samples, validation set 100 samples)
- Consists of 3 inception blocks
- Categorical cross entropy used as loss function

Intuition behind this architecture -

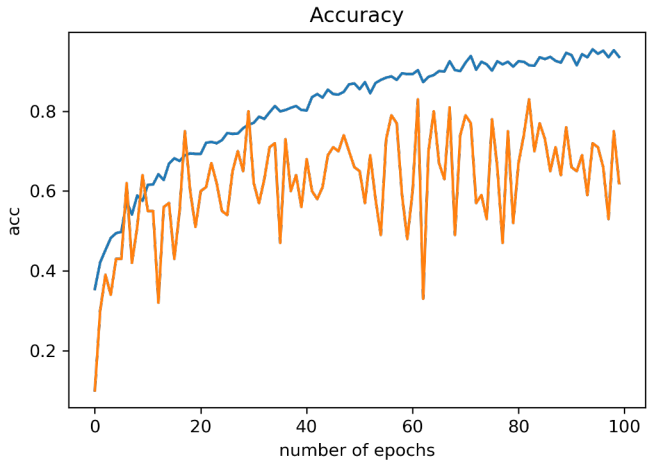
1) [4] low-level features tend to be more contributed for improving the genre classification performance (Choi et al., 2017). Therefore, how to construct an appropriate CNN structure to maximally abstract high-level information and preserve the lower-level features simultaneously

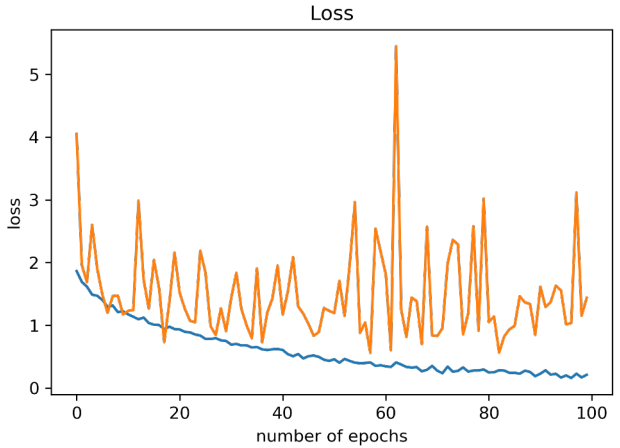
1) [4] sound events are accumulated by frequency over time domain which causes the individual genre has different performance sensitivity to different time scales and levels of features. Therefore, it is necessary to design a specific CNN structure which can comprehensively handle multi-scale of audio features


Broadcast Module(BM)
DepthConcat
Inception (c)
Conv 3×3+1(S)
Conv 5×5+1(S)
Conv 1×1+1(S)
BatchNormalization
MaxPool 3×3+1(S)
DepthConcat
Inception (b)
DepthConcat
Inception (a)
MaxPool 4×1+None(S)
BatchNormalization
Conv 1×1+1(S)
AveragePool 2×2+2(S)
Global AveragePool
Softmax
Output
Input
Conv 3×3+1(S)
Transition layers
Decision layers
Decision layers

```
[[ 4  5  0  0  0  0  0  0  1  0]
 [ 0 10  0  0  0  0  0  0  0  0]
 [ 2  0  5  1  0  0  0  1  0  1]
 [ 0  0  0 10  0  0  0  0  0  0]
 [ 1  0  0  1  8  0  0  0  0  0]
 [ 0  0  0  0  0 10  0  0  0  0]
 [ 1  0  0  0  0  0  9  0  0  0]
 [ 0  0  0  0  0  0  0 10  0  0]
 [ 0  0  0  0  0  0  0  0 10  0]
 [ 0  0  0  0  0  0  0  0  0 10]]
```

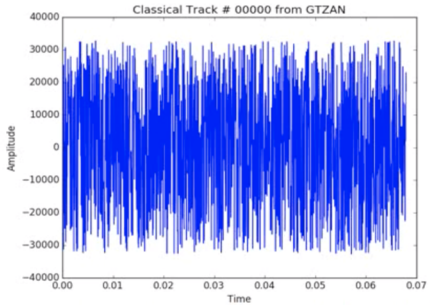
	precision	recall	f1-score	support
Class 0	0.50	0.40	0.44	10
Class 1	0.67	1.00	0.80	10
Class 2	1.00	0.50	0.67	10
Class 3	0.83	1.00	0.91	10
Class 4	1.00	0.80	0.89	10
Class 5	1.00	1.00	1.00	10
Class 6	1.00	0.90	0.95	10
Class 7	0.91	1.00	0.95	10
Class 8	0.91	1.00	0.95	10
Class 9	0.91	1.00	0.95	10
accuracy			0.86	100
macro avg	0.87	0.86	0.85	100
weighted avg	0.87	0.86	0.85	100



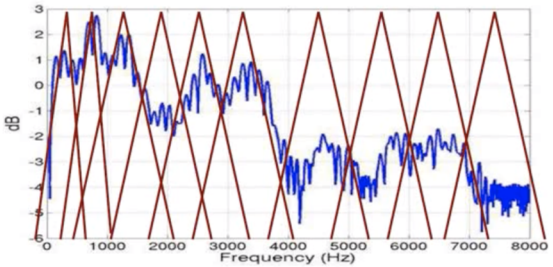


Feature Extraction

- Mel Frequency Cepstrum Coefficients(MFCCs)
  - Widely used feature for speech/sound recognition as timbral feature
  - Takes into consideration human perception of frequencies



Time series data of signal divided into frames (fixed no. of samples)



Triangular filters concentrated on lower frequencies, spaced to Mel scale [4]

Feature Extraction

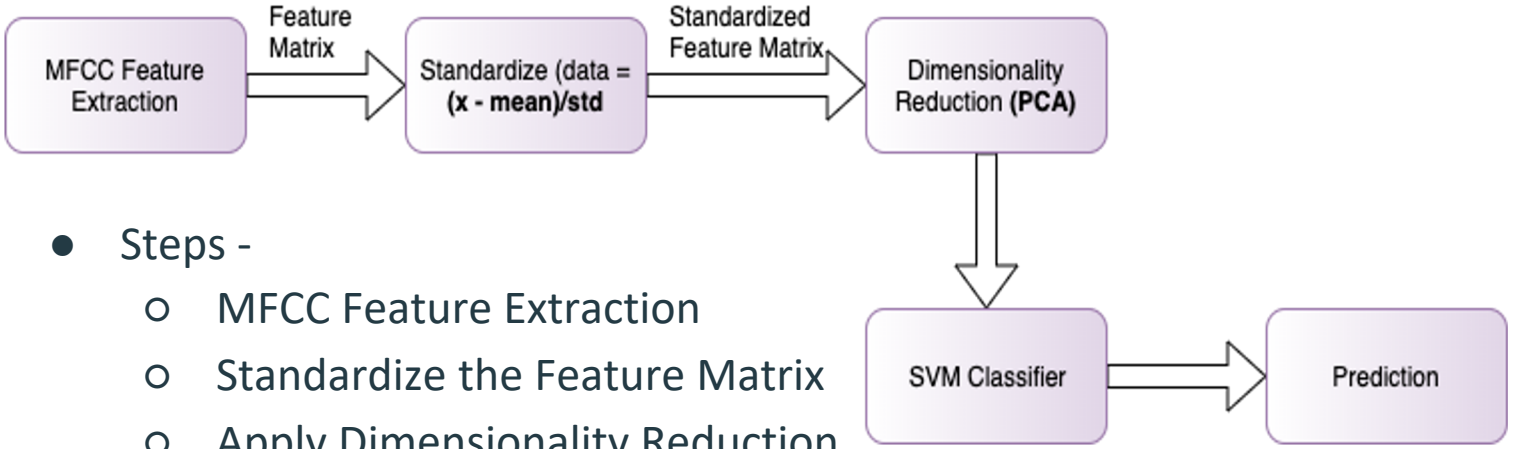
- After MFCC Extraction, each song can be visualized as an array of size N_{Frames} X N_{MFCC}
- For the GTZAN Dataset, each frame is configured as 256 samples

$$N_{Frames} = \frac{(\text{Sample Rate} \times \text{Song Length})}{\text{Frame Size}} = \frac{22050\ \text{Hz} \times 30\ \text{sec}}{256} \approx 2584\ \text{Frames}$$

$$\begin{bmatrix} MFCC_{1,F1} & \cdots & MFCC_{14,F1} \\ \vdots & \ddots & \vdots \\ MFCC_{1,F2584} & \cdots & MFCC_{14,F2584} \end{bmatrix}$$

- We further take mean vector and covariance of this MFCC matrix to create our final feature vectors

Support Vector Machine



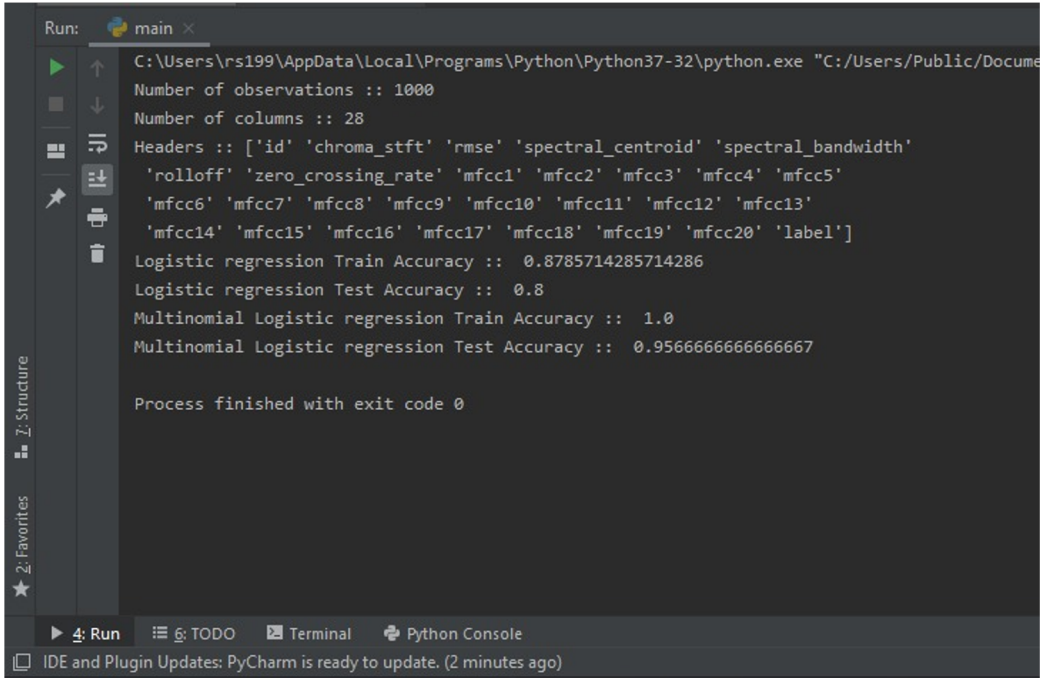
- Steps -
  - MFCC Feature Extraction
  - Standardize the Feature Matrix
  - Apply Dimensionality Reduction
  - Train SVM based on the output of feature extracted and reduced matrix
  - Based on the training, predict the results for the test data.

Result of SVM

```
(base) Ajinkyas-MacBook-Pro:MusicGenreClassification ajinkyadande$ python TrainData_SVM.py
Loading data
Shape of training set after vectorization:
(900, 121800)
Shape of test set after vectorization:
(100, 121800)
Running PCA
Shape of training set after pca:
(900, 100)
Shape of test set after pca:
(100, 100)
Training SVM
Train Accuracy: 99.9%
Test Accuracy: 60.0%
```

Multinomial Logistic Regression

- We have followed the same procedure as SVM
- Accuracy - 96%



Lesson learned

- Data preprocessing - how to generate spectrograms and use CNN to their advantage.
- Comparison of different models based on accuracy
  - Logistic regression - best accuracy
  - BBNN - second best
  - CNN - third best
- Therefore, we may safely conclude that complex models don’t always work.

Contributions

- Abantika - CNN, CNN-LSTM
- Abhilasha - SimpleRNN
- Ajinkya - SVM
- Omkar - Bottom up Broadcast Neural Network
- Rounak - Multinomial Logistic regression
- Github link: https://github.com/omkar1729/Music-genre-Recognition---SML

References

[1] G. Tzanetakis and P. Cook. Musical genre classication of audio signals. *IEEE Transactions on Speech and Audio Processing, 10(5):293{302, July 2002.*

[2] Hareesh Bahuleyan. Music Genre Classication using Machine Learning Techniques. *arXiv e-prints, page arXiv:1804.01149, Apr 2018.*

[3] Archit Rathore and Margaux Dorido. Music Genre Classication. *Indian Institute of Technology, Kanpur, Department of Computer Science and Engineering, 2010.*

[4] Caifeng Liu et al. “Bottom-up Broadcast Neural Network For Music Genre Classification”. *In:CoRR abs/1901.08928 (2019). arXiv: 1901.08928.url :http://arxiv.org/abs/1901.08928*.