**Final Project Report**

**COSC 6368: Artificial Intelligence**

**University of Houston**

**Department of Computer Science**

# Exploratory Analysis on Music Genre Classification

| Name | PeopleSoft ID |
|---|---|
| Nirupam Bidikar | 1878058 |
| Pranav Saineni | 1884587 |
| Rahul Raj Mogili | 1900425 |

# 1. SURVEY

## Domain - Deep Learning:

Deep learning is a subset of machine learning where algorithms that have multiple layers which increasingly extract higher-level features from the raw input. For instance, in image processing, lower layers might determine edges, whereas higher layers might determine digits or faces. Deep learning architectures like deep neural networks, recurrent neural networks, and convolutional neural network are being implemented in fields such a computer vision for automatic face recognition, natural language processing for Speech Recognition, drug design for drug properties description, medical image analysis for accelerating MRI image processing, and many other fields where they have produced results comparable and in some cases surpassing results produced by humans. In our project we try to map genres to audio clips using Deep Learning and Bayesian Learning techniques.

## Backpropagation:

It is an algorithm used in training feedforward neural networks for supervised learning. The process of backpropagation takes in the final decisions of a model's training pass, and then it determines the errors in these decisions. The errors are calculated by contrasting the outputs of the network and the expected outputs of the network. Once the errors in the network's decisions have been calculated, this information is transported back (backpropagated) through the network and

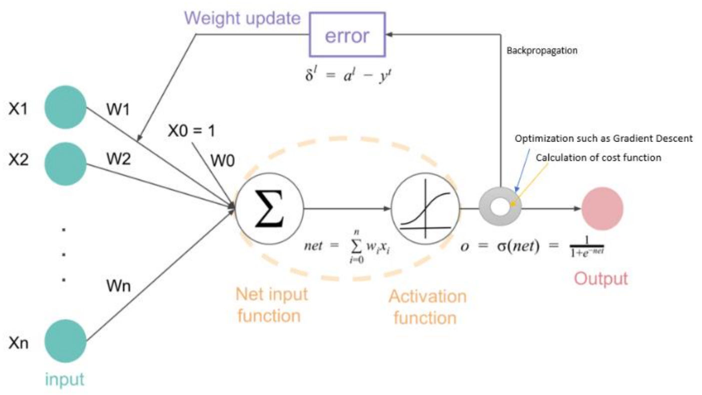

Weight update
error
$\delta^l = a^l - y^l$
Backpropagation
X1
X2
Xn
W1
W2
Wn
X0 = 1
W0
Optimization such as Gradient Descent
Calculation of cost function
$net = \sum_{i=0}^{n} w_i x_i$
$o = \sigma(net) = \frac{1}{1+e^{-net}}$
Output
Net input function
Activation function
input

*Figure 1: An overview of Backpropagation*

the parameters of the network are altered along the way. The method that is used to update the weights of the network is based on the chain-rule.

**Bayes by Backpropagation:**

Our goal in training a neural network is to find an optimal point estimate for the weights which best represents the data. Networks trained using this way performs well when we have large samples of data and fails to express uncertainty in small samples of data, leading to overconfident decisions. To overcome this drawback, we use Bayesian learning to neural networks. Bayesian inference for neural networks calculates the posterior distribution of the weights given the training data, *P(w|D)*. Each possible configuration of the weights, weighted according to the posterior distribution, makes a prediction about the unknown label given the test data. To make it easier and to have a better understanding of the distribution at each weight, we will use a Gaussian distribution.

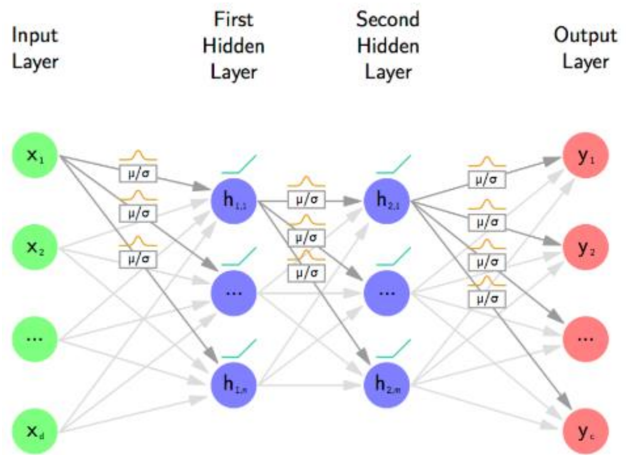


*Figure 2 Representation of network using Bayes by Backprop. Taken from [3]]*

Exact Bayesian inference on the parameters is not tractable, to address this problem we used the "Bayes by Backprop" algorithm[1] which derives a variational approximation to the true posterior and as we are using a Bayesian for the network, we have to define a probability distribution over a set of distributions which is also known as a prior.

- Prior Probability: the prior probability of a random event or an uncertain proposition is the unconditional probability that is assigned before any relevant evidence is considered. Since we are using a Bayesian for the network, we need to define a Prior over the weights. The prior over the weights vector simply corresponds to the product of the individual Gaussians.

$$\log P(w) = \sum_i \log N(w_i | 0, \sigma_p^2)$$

Where N = arbitrarily sample size

w = Weight

$\sigma^2$ = variance

- Likelihood: Although a likelihood function might look just like a probability density function, it is fundamentally different. A probability density function is a function of your data point, and it will tell you how likely it is that certain data points appear. A likelihood function, on the other hand, takes the data set as a given, and represents the likeliness of different parameters for your distribution. It is the probability of the evidence given the parameters. We will use the softmax to define our likelihood. (P(Di | w)).
- Posterior Probability: the posterior probability of a random event or an uncertain proposition is the conditional probability that is assigned after the relevant evidence or background is considered. It is the probability of the parameter θ given the evidence. The variational posterior on the weights is centered on the mean (μ) and has variance ($\sigma^2$ )

$$\log q(\mathbf{w} \mid \theta)=\sum_i \log \mathrm{N}(\mathbf{w}_i \mid \mu,\sigma^2)$$

- Loss: Now that we have defined our likelihood, the prior, annzd the variational posterior, we are now able to build our combined loss function as

$$F(Di,\theta)=1/M(\log q(w \mid \theta)-\log P(w))-\log P(Di \mid w)$$

we need to ensure that the variance is a non-negative, which we will so by using the softplus function to express variance in terms of an unconstrained parameter

**Logistic Regression:**

The logistic regression is used when the data is categorical. It is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. We can also set a decision boundary in logistic regression which acts like a threshold. Based on this threshold, the obtained estimated probability is classified into classes. A cost function cannot be used in logistic regression as there is a chance the gradient descent will converge into global minimum.

The logistic regression uses the sigmoid function and was developed by statisticians to describe properties of population growth in ecology and maxing out at the carrying capacity of the environment. It looks like S-shaped curve taking any real valued number and mapping it between 0 and 1 but never exactly at those limits like the shown in Figure 3.

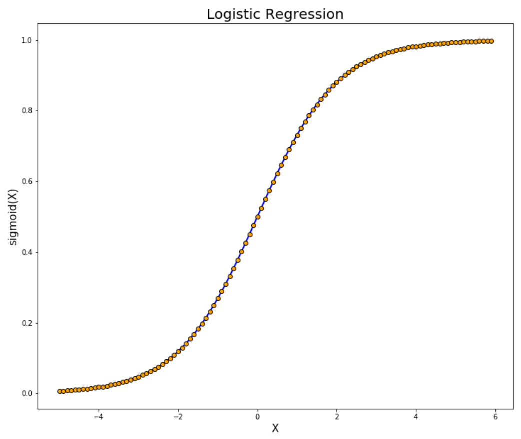


*Figure 3: Logistic function (sigmoid)*

An example of logistic regression equation looks like,

$$P = \frac{e^{a+bX}}{1+e^{a+bX}}$$

Where *P* is predicted output, a is the bias or intercept term and b is the coefficient for a single input value (X).

**Random Forest Classifier:**

There are several classification algorithms such as logistic regression, support vector machines, decision trees, naïve Bayes classifier but in the hierarchy of classifiers is topped by random forest classifier. Random Forest algorithm is a supervised classification algorithm. It consists of large number of individual decision trees and each individual tree spits out a class prediction and the class with most votes becomes our model prediction. In other words, we can say, many uncorrelated trees operating as a committee outperform any of the individual constituent models.

Alternatively, the random forest can apply weight concept for considering the impact of result from any decision tree. The tree with high error rate is given low weight values and vice versa. This would make the decision impact of trees with low error rate. Overfitting is one critical problem that may make the results worse but if there enough trees in the forest, the classifier won’t overfit the model. Random forest can also handle missing values. Basic parameters to Random

Forest Classifier can be total number of trees to be generated and decision tree related parameters like minimum split, split criteria etc.

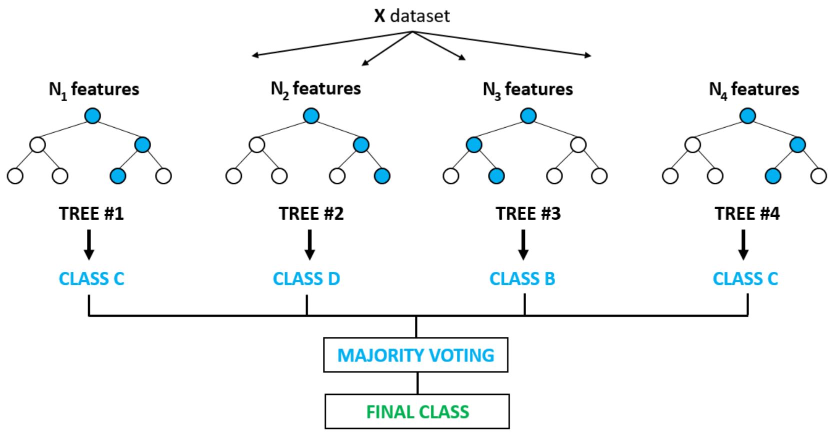


*Figure 4: Concept of RFC (Image taken from Google)*

**Convolutional Neural Network:**

Typical neural networks pass signals along the input-output channel in a single direction, without allowing signals to loop back into the network. This is called a forward feed. While forward feed networks were successfully employed for image and text recognition, it required all neurons to be connected, resulting in an overly complex network structure.

The convolutional neural network (CNN) is a Deep Learning algorithm which can take in an input image, assign importance to various elements in the image and provide differentiation from one another. The pre-processing required in a CNN is lower as compared to other classification algorithms. A CNN can successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. It performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. Hence, the network can be trained to understand the sophistication of an image better.

The CNN takes an image, pass it through a series of convolutional, nonlinear, pooling and fully connected layers, and get an output. The output can be a single class or a probability of classes that best describes the image.

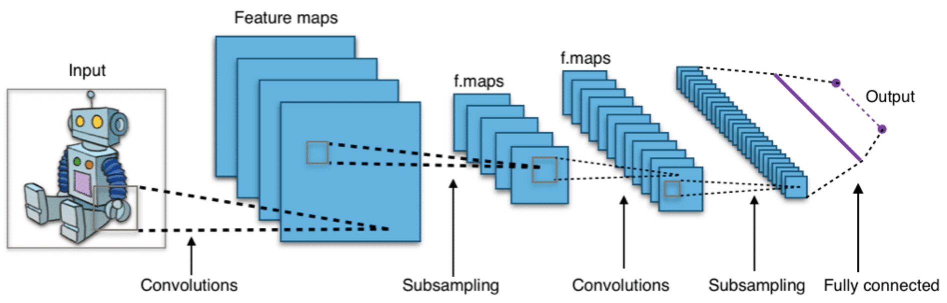


*Figure 5: Concept of CNN. (Image taken from Google)*

## Support Vector Machine-

Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification and regression challenges. In this project we use it for classification. In this model, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well. The learning of the hyperplane in linear SVM is done by transforming the problem using some linear algebra. This is where the kernel plays role.

For linear kernel the equation for prediction for a new input using the dot product between the input (x) and each support vector ($x_i$) is calculated as follows:

$$f(x) = B(0) + sum(a_i * (x,x_i))$$

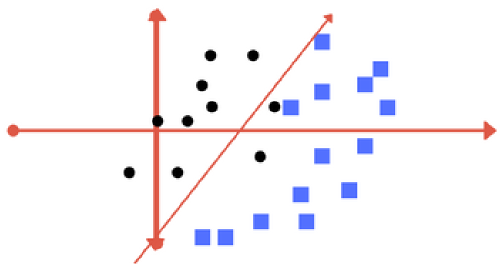


*Figure 6: Misclassification due to lower regularization values*

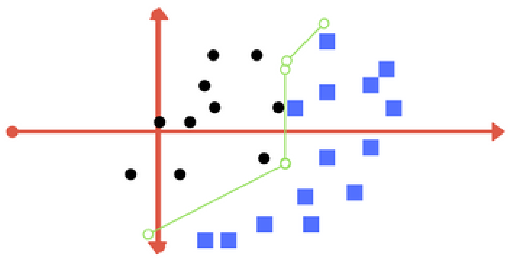


*Figure 7: Higher regularization values leads to this graph*

## 2. DEVELOPMENT OF THE IDEA

### 2.1 Dataset:

To conduct the exploratory analysis, we use the GTZAN[4] and FMA[5] datasets. A brief description about the datasets is given below.

The Z dataset consists of 1000 audio tracks each 30 seconds long. It contains 10 genres, each represented by 100 tracks. The tracks are all 22050 Hz monophonic 16-bit audio files in .wav format. We used the liborsa library to extract features such as *Mel Frequency Cepstral Coefficient* (MFCC), Chromagram, Zero Crossing Rate, Root Mean Square Error, Spectral Rolloff, Spectral Bandwidth and Spectral Centroid. It also consists on image data of the Spectrogram for each track. A spectrogram is a visual representation of the spectrum of frequencies of a signal as it varies with time.

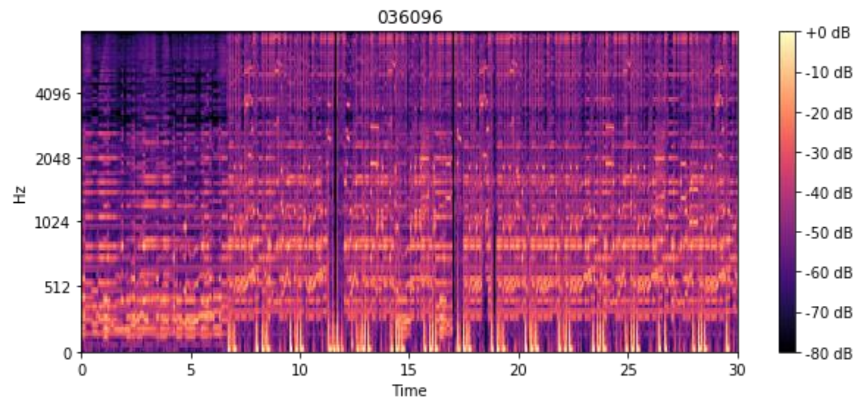


*Figure 6 Spectrogram*

The FMA dataset contains extracted features for 106,747 songs. This dataset contains track metadata, genres list, feature sets for all the songs and other metadata files that includes songs data taken from Spotify. The Features file consists of the common features extracted from 30 second audio clips using the *librosa* API in python. This will give us features such as MFCC and Chromagram. MFCC is widely used in speech analysis which relates with timbre and music instruments. Chromagram gives the information about the 12 pitch cases of the audio clip. MFCC consists of 140 features which are all being considered in this study.

**2.2 Preprocessing**:

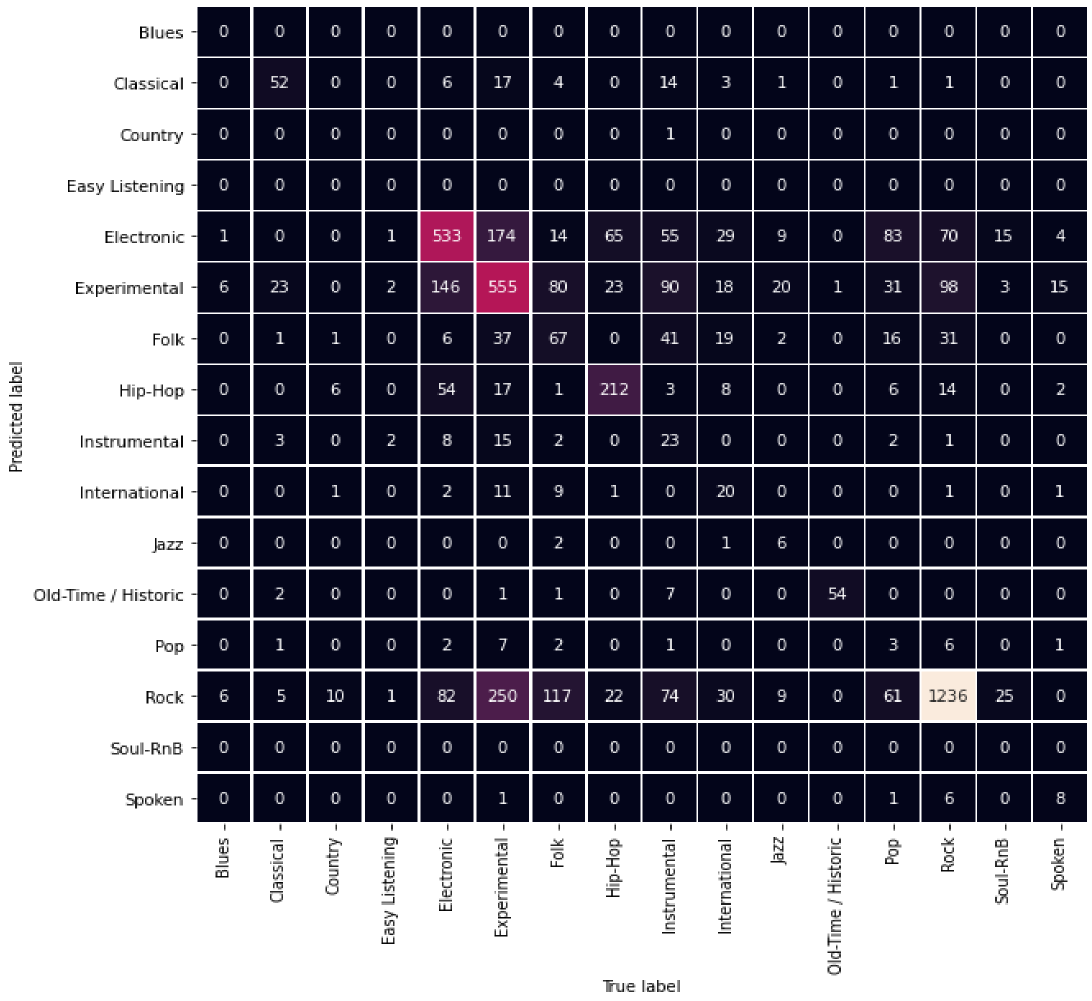
We first begin with the track metadata. This consists mappings between the track and its respective genre. Since a track can have multiple genres, we removed such entries to reduce the complexity of the dataset. We then trim the features dataset by selecting the MFCC feature. There have been studies [2] showing that MFCC is better suited for predicting the genre which is why we selected it. We then prune all those tracks from earlier which had multiple genres and remove their data from the feature set.

We then scale the data and encode the labels and prepare the data for training our models. While performing scaling, we used Normalization for the FMA dataset and Standard Scaling for the GTZAN dataset to obtain best possible fitting models.

## 3. IMPLEMENTATION

### 3.1 Classical Models:

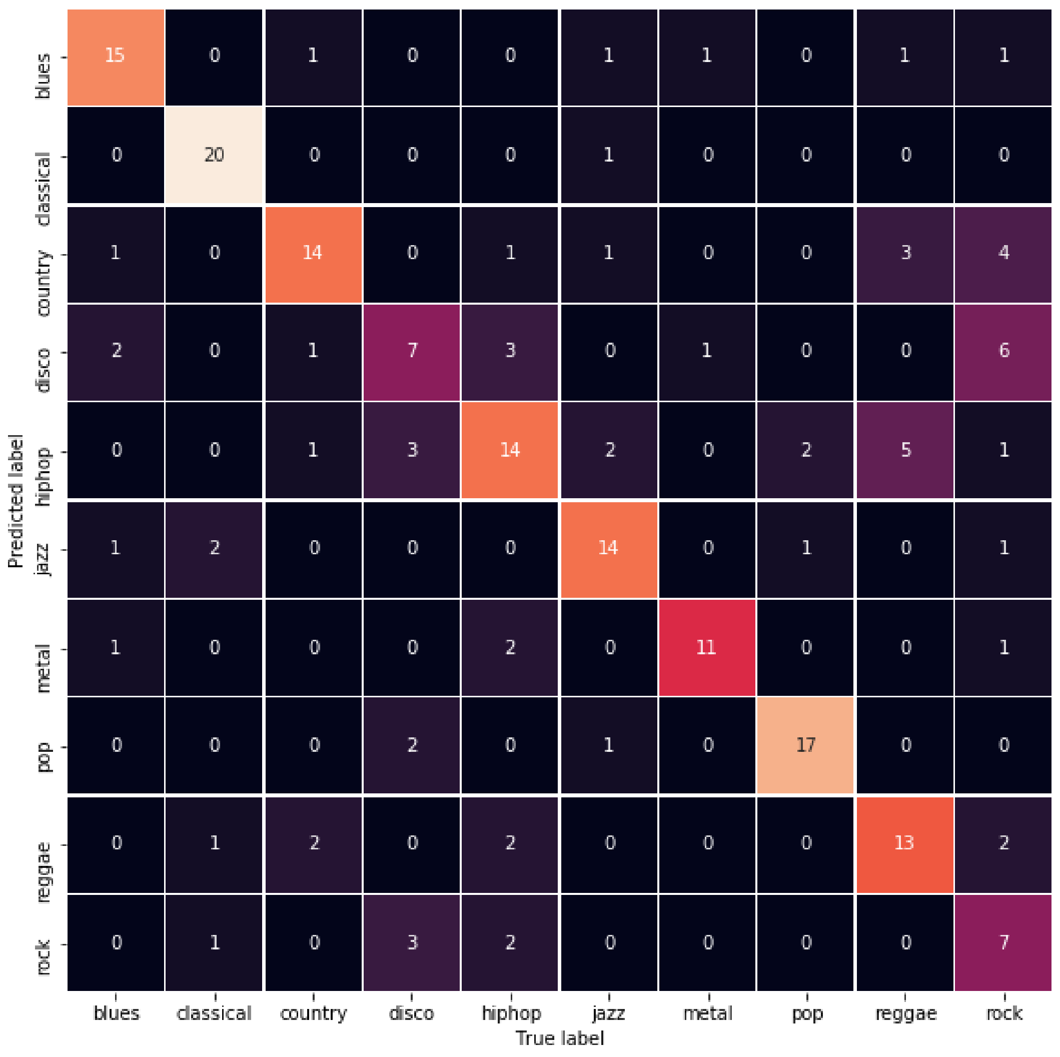
To get an initial impression of the dataset, we begin by testing it with classical models used in classification problems. We used multinomial logistic regression with a variety of solvers like “newton-cg”, “lbfgs”; “SVC” model from SVM and tested it with the MFCC features from the datasets. The accuracy results are elaborated in terms of a confusion matrix and shown below.



| | Blues | Classical | Country | Easy Listening | Electronic | Experimental | Folk | Hip-Hop | Instrumental | International | Jazz | Old-Time / Historic | Pop | Rock | Soul-RnB | Spoken |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Blues | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Classical | 0 | 52 | 0 | 0 | 6 | 17 | 4 | 0 | 14 | 3 | 1 | 0 | 1 | 1 | 0 | 0 |
| Country | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Easy Listening | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Electronic | 1 | 0 | 0 | 1 | 533 | 174 | 14 | 65 | 55 | 29 | 9 | 0 | 83 | 70 | 15 | 4 |
| Experimental | 6 | 23 | 0 | 2 | 146 | 555 | 80 | 23 | 90 | 18 | 20 | 1 | 31 | 98 | 3 | 15 |
| Folk | 0 | 1 | 1 | 0 | 6 | 37 | 67 | 0 | 41 | 19 | 2 | 0 | 16 | 31 | 0 | 0 |
| Hip-Hop | 0 | 0 | 6 | 0 | 54 | 17 | 1 | 212 | 3 | 8 | 0 | 0 | 6 | 14 | 0 | 2 |
| Instrumental | 0 | 3 | 0 | 2 | 8 | 15 | 2 | 0 | 23 | 0 | 0 | 0 | 2 | 1 | 0 | 0 |
| International | 0 | 0 | 1 | 0 | 2 | 11 | 9 | 1 | 0 | 20 | 0 | 0 | 0 | 1 | 0 | 1 |
| Jazz | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 1 | 6 | 0 | 0 | 0 | 0 | 0 |
| Old-Time / Historic | 0 | 2 | 0 | 0 | 0 | 1 | 1 | 0 | 7 | 0 | 0 | 54 | 0 | 0 | 0 | 0 |
| Pop | 0 | 1 | 0 | 0 | 2 | 7 | 2 | 0 | 1 | 0 | 0 | 0 | 3 | 6 | 0 | 1 |
| Rock | 6 | 5 | 10 | 1 | 82 | 250 | 117 | 22 | 74 | 30 | 9 | 0 | 61 | 1236 | 25 | 0 |
| Soul-RnB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Spoken | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 6 | 0 | 8 |

*Figure 7: Confusion matrix of Logistic Regression – newton-cg (FMA)*

We can observe from the matrix that electronic and experimental are being mismatched with each other. The values of the features must be quite similar for the model to inaccurately predict one of them as the other class. Same is the case with experimental and rock, fold and rock. The most accurate genre predicted by the model.



| Predicted label \ True label | blues | classical | country | disco | hiphop | jazz | metal | pop | reggae | rock |
|---|---|---|---|---|---|---|---|---|---|---|
| blues | 15 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| classical | 0 | 20 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| country | 1 | 0 | 14 | 0 | 1 | 1 | 0 | 0 | 3 | 4 |
| disco | 2 | 0 | 1 | 7 | 3 | 0 | 1 | 0 | 0 | 6 |
| hiphop | 0 | 0 | 1 | 3 | 14 | 2 | 0 | 2 | 5 | 1 |
| jazz | 1 | 2 | 0 | 0 | 0 | 14 | 0 | 1 | 0 | 1 |
| metal | 1 | 0 | 0 | 0 | 2 | 0 | 11 | 0 | 0 | 1 |
| pop | 0 | 0 | 0 | 2 | 0 | 1 | 0 | 17 | 0 | 0 |
| reggae | 0 | 1 | 2 | 0 | 2 | 0 | 0 | 0 | 13 | 2 |
| rock | 0 | 1 | 0 | 3 | 2 | 0 | 0 | 0 | 0 | 7 |

*Figure 8: Confusion matrix of Logistic Regression – GTZAN*

For the GTZAN dataset using the Newton-cg solver, we were able to achieve a test accuracy around 71%. We can clearly see the classifier correctly predicted most of the classes by looking at the diagonal elements with few errors across the board. This might have occurred due the concise nature of the dataset and the features at play in that set. LBFGS also ended up having nearly the same confusion matrix. SVC performed the best of all the three algorithms with respect to the training accuracy but a lower test accuracy showing the model did not learn the patterns well.

*Table 1: Results with FMA dataset*

| **Method used** | **Accuracy (Train)** | **Accuracy (Test)** |
|---|---|---|
| Newton-cg | 59.1% | 54.8% |
| lbfgs | 59.1% | 55.9% |
| SVC | 73.1% | 59.6% |

*Table 2: Results with GTZAN dataset*

| **Method used** | **Accuracy (Train)** | **Accuracy (Test)** |
|---|---|---|
| Newton-cg | 70.1% | 61.5% |
| lbfgs | 70.1% | 61.5% |
| SVC | 77.3% | 64% |

As we can see, we have obtained a better result than the previous study [2]. The author concluded that one of the reasons for low accuracy was the size of the dataset. With our ~40,000 training samples, we can say that given more training data the accuracy increases.

### 3.2 Random Forest Classifier:

Following the results on classical models, we proceeded to experiment with Random Forest Classifier. We once again see that the classifier benefits with a larger dataset giving better accuracy with a greater number of samples. We can also see the increase in accuracy with the increase in levels. We chose to go with 12 levels as we start getting diminishing returns with huge computational overhead due to which we decided to stop at 12 levels and calculate the accuracy scores for the model at that level.

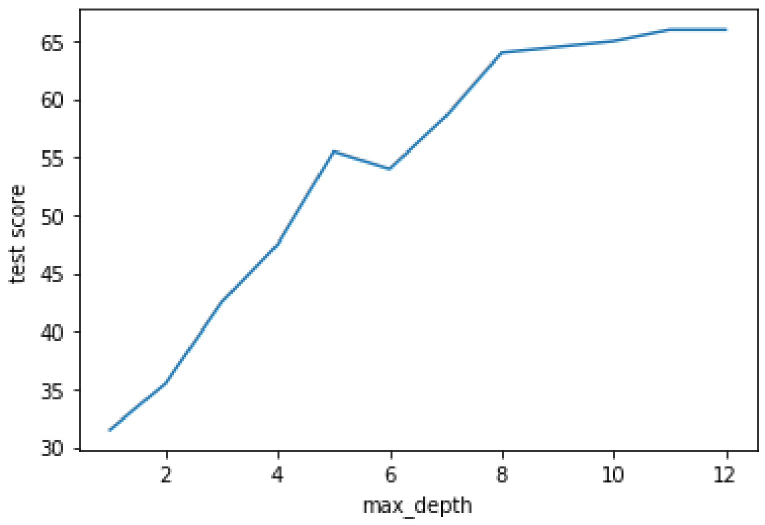


*Figure 9: Plot for GTZAN dataset*

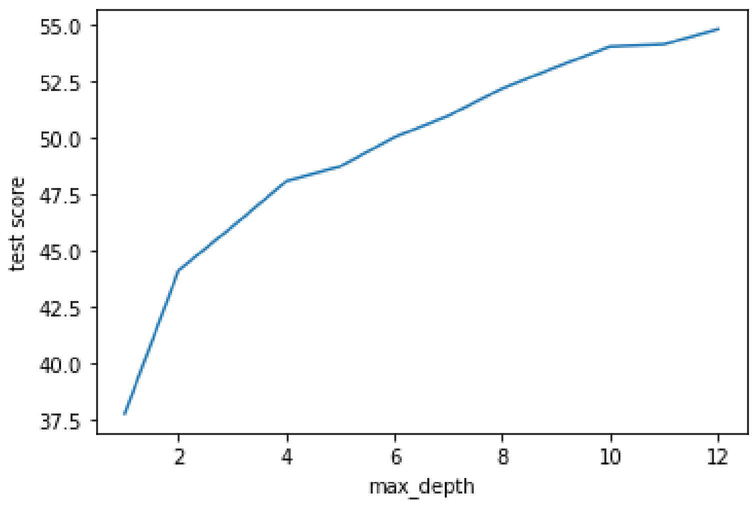


*Figure 10: Plot for FMA dataset*

As we see from Figure 9, there is a decline in accuracy while the classifier is being trained on the GTZAN dataset. This might be due to slight overfitting occurring but then the model recovers and plateaus at a 65% accuracy. For the FMA dataset Figure 10 we a steady increase in accuracy up to levels 10 and then we reach the point of diminishing returns.

## 3.3 Convolutional Neural Network:

Given the excellent performance of CNNs in the field of Computer Vision and pattern recognition, we thought of applying it to the music domain to find such patterns and predict the genre of an audio clip given to it. We applied these to both the datasets and found relatively similar results and trends. We had to choose different network structures for each dataset to better tune the model for that specific data.

**Structure of the Network:**

**Network for FMA dataset**

Input nodes: 140

Hidden Layers: 2 (20 nodes each)

Activation Function: '*ReLU'*

Output Layer: 1 layer (16 nodes)

Activation function: '*softmax'*

Optimizer: '*adam'*

Learning rate: *default*

Loss function: 'sparse categorical cross entropy'

**Network for GTZAN dataset**

Input nodes: 26

Hidden Layers: 3 (256 nodes, 128 nodes and 64 nodes respectively)

Activation Function: '*ReLU'*

Output Layer: 1 layer (10 nodes)

Activation function: '*softmax'*

Optimizer: '*adam'*

Learning rate: *default*

Loss function: ‘sparse categorical cross entropy’

The networks were built using the Tensorflow library with Keras acting as a high-level wrapper over it to simplify construction of models. We test the models with various hyper parameters, and we have demonstrated in the report the combination which has fared the best among those.

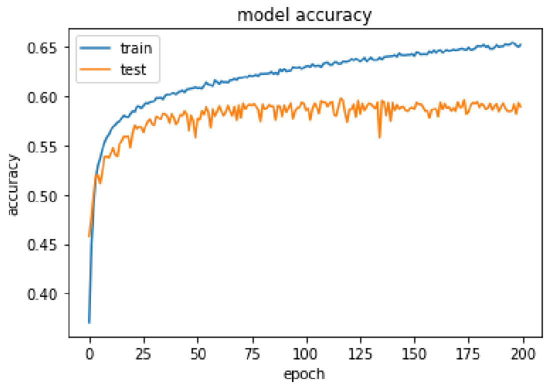


*Figure 11: Accuracy plot for CNN trained on FMA*

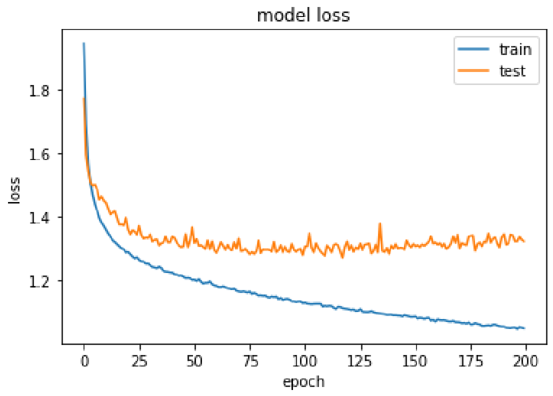


*Figure 12: Loss plot for CNN trained on FMA*

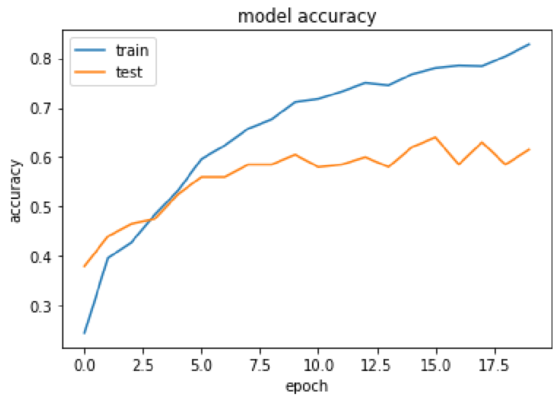


*Figure 13: Accuracy plot for CNN trained on GTZAN*

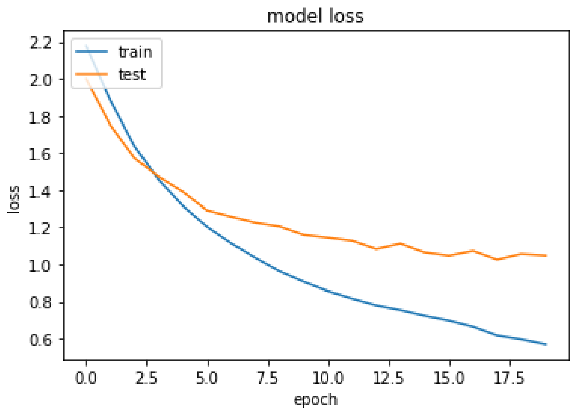


*Figure 14: Accuracy plot for CNN trained on GTZAN*

While training both the datasets, we observe commonalities in certain trends – loss is minimized (steady decline) and the increase in accuracy over number of epochs. Again due to the nature of the dataset we see the model trained on GTZAN achieve higher train accuracy but both the models have relatively the same test accuracy.

**Overall Performance Comparison of Classical Models and Neural Network**

**[FMA, GTZAN]**

Here we can see that traditional classification algorithms perform better than the CNN as Neural

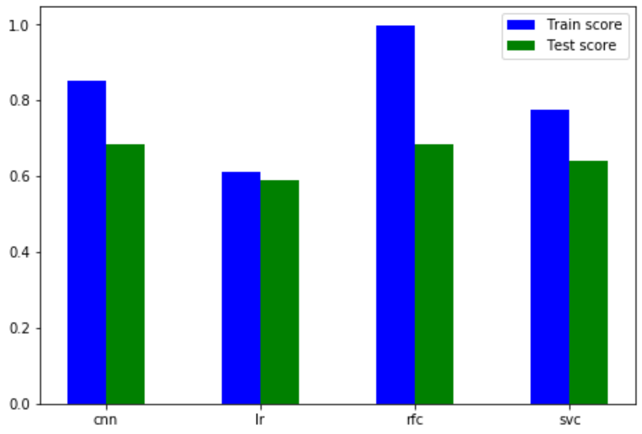


Figure 12: Train and Test scores (FMA)

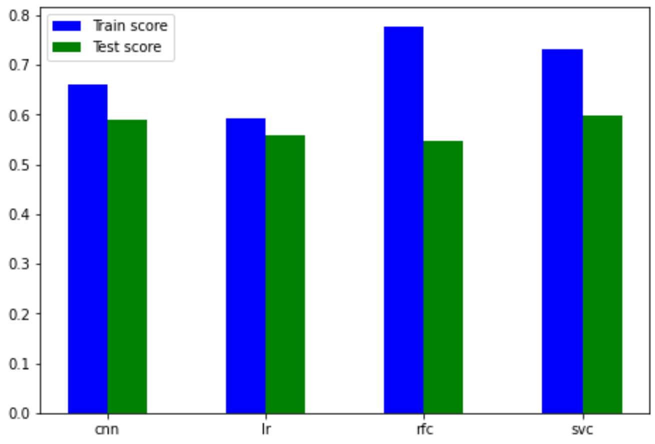


Figure 11: Train and Test scores (GTZAN)

Networks are well suited for image classification applications. Here the data being highly complex and there being a lot of similarities among genres, we can see how it my negatively affect the performance of the network.

**Bayes by Backpropagation:**

Our main idea was to see how Bayesian Neural Networks compare with a traditional neural network in classifying problems. By inducing uncertainty in the model, it is better equipped to perform well in predicting unseen data. We tested out this concept on the standard MNIST dataset to classify the digits. This gave us promising results and is shown in the plot below.

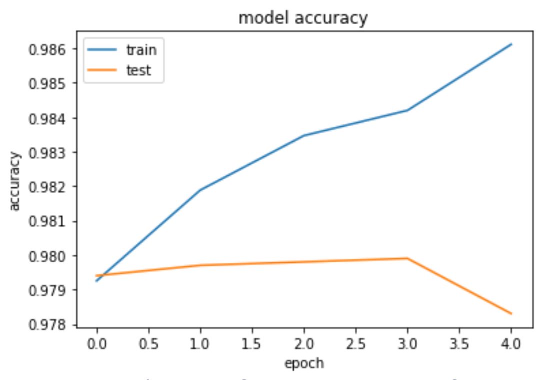


Figure 14: CNN accuracy on MNIST

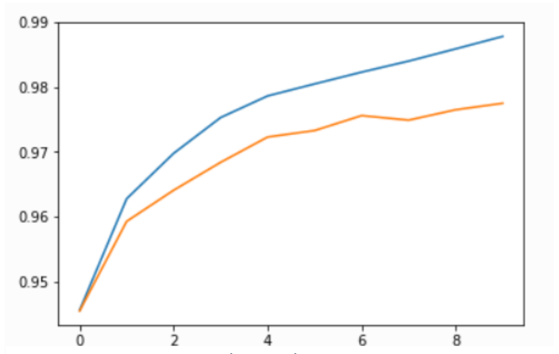


Figure 15: Bayes by Backprop accuracy on MNIST

(epoch vs accuracy)

We started observing irregular patterns when we applied Bayes by backprop on the music dataset. Using GTZAN set, we could not get a model with a uniform trend on accuracy which goes on to say the model was not learning. The same happened with FMA dataset; again, we could not get a uniform learning trend on models even with modifying the hyperparameters and standard deviations of the distribution of weights. The visualization for the observation is shown below.

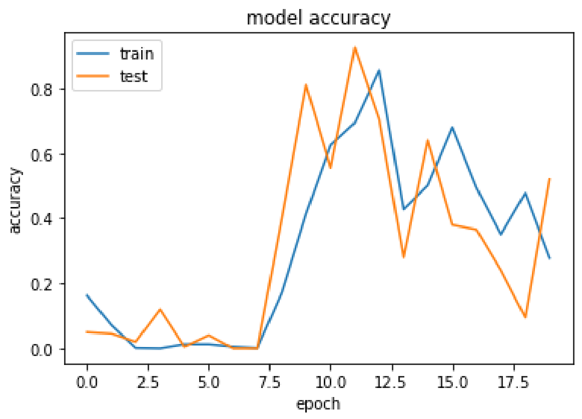


*Figure 16: Bayes by BackProp using GTZAN*

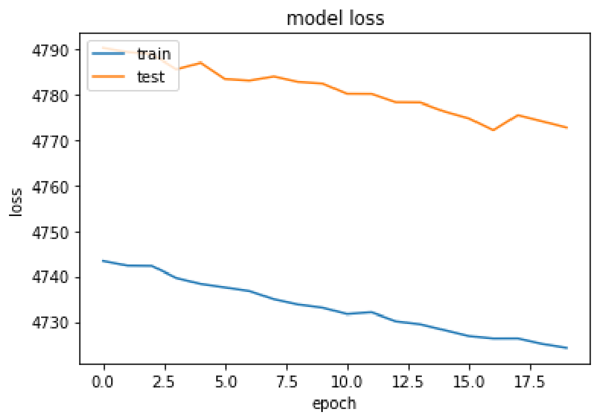


*Figure 17: Loss plot by BackProp on GTZAN*

When compared with traditional networks, we expected this model to perform better given the uncertainty induction.

## 4. CONCLUSION

During the inception of this project, it was our understanding that inducing uncertainty into the model would help in better prediction. Training models on both datasets gave us inconsistent results and extremely low accuracies within the range of 17 – 35%. Considering the performance of other models on the same datasets, we suspect that this approach (Bayes by BackPropagation) might not be suited for analysis of such data. Tweaking CNNs and using timeseries data to train a RNN would be some of the better approaches to tackle this problem.

In future, we have our sights set on using Bayes by BackProp on Spectogram data as it again brings it down to an image classification problem and we are curious to see how it turns out. We would like to further investigate the features of audio clips and see how other features impact the genre and accordingly build the models.

## 5. REFERENCES

[1] Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*.

[2] Zhang, S., Gu, H., & Li, R. (2019). MUSIC GENRE CLASSIFICATION: NEAR-REALTIME VS SEQUENTIAL APPROACH.

[3] A manual for Bayes by Backprop https://gluon.mxnet.io/chapter18_variational-methods-and-uncertainty/bayes-by-backprop.html

[4] Tzanetakis, G., & Cook, P. (2002). Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, *10*(5), 293-302.

[5] FMA dataset https://github.com/mdeff/fma

[6] PyTorch Library for Bayesian Neural Networks https://github.com/piEsposito/blitz-bayesian-deep-learning

[7] Project Code https://github.com/nirupam52/COSC-6368-project-