Music Information Retrieval For Genre Classification

Soumya Ranjan Sahoo
Department of Computer Science
Saarland University
s8sosaho@stud.uni-saarland.de

Anindita Ghosh
Department of Computer Science
Saarland University
s8anghos@stud.uni-saarland.de

Abstract

Music Genre Classification is one of the many branches of Music Information Retrieval (MIR). However, music genre classification has been a challenging task in the field of MIR. Music genres are hard to systematically and consistently describe due to their inherently subjective nature. In this report, we revisit the problem of Music Genre Classification by using both the more traditional frame-level audio features and exploiting the temporal structure in audio spectrograms using deep convolutional and recurrent models. To this end, we implement k-nearest neighbors (k-NN), Gaussian Mixture Model (GMM), Multi-class SVM, Convolutional Neural Network (CNN), and Convolutional Recurrent Neural Network (CRNN) to classify the following four genres: Dark-Forest, Hi-Tech, Full-On, and Goa. We further extract 30 temporal features using a Long Short Term Memory (LSTM) based Auto-encoder from individual frames, and augment them with the frame-level audio features, which is a novel contribution in this work. With a very limited number of supervision signals, our best performing model achieves an F1 score of 0.84. Additionally, we discuss and compare the effectiveness of machine-learning versus deep-learning models for the problem of music genre classification.

Index Terms - Genre Classification, Information Retrieval, k-nearest Neighbors, Gaussian Mixture Model, Multi-class SVM, Convolutional Neural Network, Convolutional Recurrent Neural Network, Long Short Term Memory, Auto-encoder.

1 Introduction

Music information retrieval (MIR) encompasses most audio analysis tasks such as genre classification, song identification, chord recognition, sound event detection, mood detection, and feature extraction. Search algorithms, in particular, popularized by the likes of Spotify, Shazam Entertainment, Amazon music, etc, can efficiently search and identify a song given an audio sample, or any text input. Also, given the rapid growth in music tracks, the need for accurate meta-data required for database management and indexing for search and storage purposes climbs in proportion. Songs are analyzed based on their digital signatures for some factors, including tempo, acoustics, beats, energy, etc.
Machine Learning techniques have proved to be quite successful in extracting trends and patterns from the large pool of data. The same principles are applied in Music Analysis as well. To this end, we use a python based audio and music signal analysis library - Librosa from [10]. It includes the nuts and bolts to build a MIR system.

We implement a variety of classification algorithms admitting two different types of input - for machine learning and deep learning models respectively, with appropriate pre-processing. We experimented with k-NN, GMM, a Multi-class SVM with different kernels, a simple Feed-Forward Network neural network, with a CNN, and advanced CRNN classifiers. We further perform feature extraction from the beats of an individual frame by using an LSTM-AutoEncoder and augment these features to our original feature set. We find that feature augmentation together with a feature

.

selection strategy, sequential forward selection [12] in our case, improves the overall performance of the classifier.

Lastly, we investigate and report a formal comparison between feature drive machine learning models and the data-hungry deep learning models for the problem of music genre classification.

2 Related work

Music genre classification, as a branch of audio and speech processing, has been studied by many. A standard approach consists of two steps, feature extraction followed by classification. Most of the genre classification studies focus on finding the best set of temporal features, transformations, and filters that best represent the music. Machine learning techniques have been used for music genre classification for decides now post the feature extraction step.

In 2002, G. Tzanetakis and P. Cook [14] used both the mixture of Gaussian model and k-nearest neighbors along with three sets of carefully hand-extracted features representing timbral texture, rhythmic content, and pitch content. They achieved 61% accuracy. As a benchmark, human accuracy averages around 70% for this kind of genre classification work [4]. In the following years, this task used methods such as support vector machines as shown in [2, 11]. However, they still have limited performance. In recent years, music information retrieval applied deep learning and neural networks methods. More specifically, [9] studied music feature extraction using CNN. They used MFCC audio representation and trained a music pattern extractor to classify the music genre. In [13], the authors used a hierarchical attention network to perform the task in a large dataset of nearly half a million song lyrics, obtaining an accuracy of more than 45%. There are also recent works on music genre classification using Bidirectional LSTM [1].

3 Dataset and Features

3.1 Data Collection

We use a small collection of 45 songs from 4 different sub-genres of Trance; Hi-Tech, Dark-Forest, Full-On, and Goa. All the tracks are 44100Hz Stereo audio files in .wav format. We further segment these 45 audio files into chunks of 701 files, each with a duration of 30 seconds. We hypothesize that sub-genres of a major genre makes the problem of classification relatively more challenging as they share plenty of commonalities.

3.2 Feature Extraction

3.2.1 Audio Features

In signal processing, features can be extracted either on the temporal domain or the frequency domain. Various statistical descriptors like mean, median, standard deviation give us useful features. However, we must extract the characteristics that can best describe the audio signal. We, therefore, decide on the following informative audio features and extract them using Librosa.

- **Zero Crossing Rate:** Zero-crossing rate is defined as the number of sign changes of a signal in a certain period of time [8, 14]. Sign change is defined as the transition of the signal between negative and positive values. It usually has higher values for highly percussive sounds like those in metal and rock.
- **Spectral Centroid:** Spectral centroid is a feature used on a frequency domain and indicates the point of the center of gravity of the frequencies in the frequency bin [8, 14].
- **Spectral Bandwidth:** Spectral bandwidth gives weighted average amplitude difference between frequency magnitude and brightness [8, 14]. It is an indication of the frequency range in the frame.
- **Spectral Rolloff:** Spectral rolloff is the normalized frequency at which the sum of the low frequency power values of the sound reaches a certain rate in the total power spectrum [8, 14]. Briefly, it can be defined as the frequency value corresponding to a certain ratio of the distribution in the spectrum.

- **Chroma STFT:** Chroma features are an interesting and powerful representation for music audio in which the entire spectrum is projected onto 12 bins representing the 12 distinct semitones (or chroma) of the musical octave. Chroma STFT computes a chromagram from a waveform or power spectrogram [5].
- **RMS:** Computes the root-mean-square (RMS) value for each frame, either from the audio samples or from a spectrogram.
- **Mel Frequency Coefficient of Cepstrum-MFCC:** The Mel Frequency Cepstral Coefficients (MFCCs) are a small set of features that describe the overall shape of a spectral envelope [14]. It can be considered as the feature of timbre. The purpose of the MFCC is to adapt the cepstral coefficients to the human hearing system. A human can hear the frequencies below 1 kHz as a linear scale and the frequencies above as a logarithmic scale. Because MFCC being a linear scale is one of the commonly used features in speech and speaker recognition systems. Steps of MFCC are Frame Blocking, Windowing, Fast Fourier Transform, Mel Frequency Wrapping, and Spectrum, respectively. We have used 20 coefficients for MFCC.
- **Tempo:** The tempo of a piece of music is the speed of the underlying beat. The Tempo is measured in BPM, or beats per minute.
- **Beat:** Music is periodic (rhythmic) and its speed is measured by beats, a unit of time in music around which musical events (such as notes played by instruments) are organized. Beats provide a framework for describing relative locations within a piece of music.

We further reduce the matrix representation of the MFCC feature by taking the mean vector and reduce the vector representation of all other features by taking the mean of the vector. We then perform feature scaling such that each coefficient dimension has zero mean and unit variance to avoid the classifier from getting biased towards a certain feature with a broad range of values. Thus after pre-processing our input dataset is of shape (700,27), with 20 MFCCs.

3.2.2 Latent Feature Generation

As a novel contribution to this work, we decide to use the latent representation of the beats here. The latent representation is a compressed representation of the features so that it becomes easier to analyze. It contains all the important information needed to represent the original data point. Further, we decide to use an LSTM-Autoencoder for representation learning on the beat's feature vector. Beats are temporal sequences and follow a certain pattern. These patterns vary significantly across different genres of music, and therefore we hypothesize the beat sequences to be a very strong signal for genre classification problems. Also, the latent features are based on non-linear transformations as opposed to linear ones with traditional dimension reduction techniques like Principal Component Analysis (PCA).

Given the huge success of Recurrent Neural Networks (RNN) for temporal data that evolves, we use LSTM neural network cells [7] in our work, which are a more optimized sub-type of the more general RNN. They are capable of learning the complex dynamics within the temporal ordering of input sequences as well as use internal memory to remember or use information across long input sequences. The Beats sequence problem gets more challenging as the length of the input sequence can vary. Machine learning algorithms, and neural networks, in particular, are designed to work with fixed length inputs. An Autoencoder is a neural network model that seeks to learn a compressed representation of input using self-supervised learning. They are typically trained as part of a broader model that attempts to recreate the input.

An LSTM Autoencoder is an implementation of an autoencoder for sequence data using an Encoder-Decoder LSTM architecture. One of the early and widely cited applications of the LSTM Autoencoder was in the 2015 paper [15], to both reconstruct sequences of frames of video as well as to predict frames of video, both of which are described as an unsupervised learning task. Seeking this example as the motivation for our problem, we try to use the same architecture for learning a fixed vector representation that also takes into account the temporal ordering of the observations. For a given set of sequences, an LSTM encoder is configured to read the input sequence, encode it, decode it, and recreate it. The performance of the model is evaluated based on the model's ability to recreate the input sequence. Once the model achieves a desired level of performance recreating the sequence, we remove the decoder part, leaving just the encoder model. We then use the encoder to encode input

sequences to a fixed-length vector. We use this encoded vector as a representation of the original beat vector for our classification models. In our work, we use a fixed latent vector size of 30 for encoding the beat vectors, which are typical of length ranging from 60 till 90. Therefore we compress the beat vector by almost a factor of 2 to 3, and we do it in a self - supervised learning paradigm, which doesn't take into account the labels of respective sequences. We further standardize and append these latent vectors to our input dataset, making it of shape (700,57), with 20 MFCCs, and 30 beats.

4 Classification Models

Before any training experiments, we split the dataset into train and test sets with a 75/25 ratio, and seed them throughout the whole project. For the task of genre classification, we use both machine learning models; K-Nearest Neighbor (k-NN), Gaussian Mixture Model(GMM), and Support Vector Machines (SVM), and neural models; Convolutional Neural Networks (CNN) in this work.

4.1 Classical Machine Learning Models

The choice of the machine learning models; k-NN, GMM, and SVM is majorly motivated by the different learning approaches they use for classification. While SVM tries to learn a hyperplane separating the data points, k-NN attempts to approximate the underlying distribution of the data in a non-parametric fashion. GMM contrasts the above classifier by learning the Gaussian parameters for individual classes and then using the Mahalanobis distance as the metric for classification.

- **K-Nearest Neighbor (k-NN):** The k-NN algorithm is among the simplest of all machine learning algorithms. KNN classifier is a type of instance-based learning technique and predicts the class of a new test data based on the closest training examples in the feature space. It is a lazy learning model, with local approximation.
- **Gaussian Mixture Model:** The Gaussian Mixture Model is an intuitive approach where the model consists of several Gaussian components, which can be seen to represent an underlying set of acoustic classes. The spectral shape of the acoustic class is parameterized by the mean vector and the covariance matrix using the features of a specific class. It is essentially a density estimation model, but with the smart initialization of the cluster centers; the initial position for each cluster center is at the center of each class, we tweak a density-based clustering algorithm into a classifier.
- **Support Vector Machines:** The objective of the support vector machine algorithm is to find a hyperplane in N-dimensional space(N - the number of features) that distinctly classify the data points. Hyperplanes are decision boundaries that help classify the data points. SVM uses kernel trick to solve non-linear problems in the original feature space without computing the coordinates of the data in a higher-dimensional space.

We use our processed dataset of shape (700,57), from the above section, to train and test these 3 models. For training each of these models, we use a sequential forward feature selection algorithm for selecting the optimal set of features [12]. We select this strategy for its algorithmic simplicity in terms of computation time and interpretability.

4.2 Neural Network Models

Deep learning models; Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN), and their Ensemble - CRNN [3] and Parallel CNN-RNN [6] architectures, have achieved near state-of-the-art performances for this classical problem. We already use LSTM cells for sequential beat feature extraction and encode it in the feature set for training the above machine learning models. Deep learning models are data-hungry and need tons of labeled data for learning. Also, ensemble models like CRNN and parallel CNN-RNN, can get incredibly complex, both to model and interpret. Therefore given the limited amount of labeled training examples that we have, and intending to design a relatively simple to interpret model, we implement a CNN in our work. We later compare its performance with a simpler Feed Forward Neural Network (FFNN), and a more complex CRNN model.

- **Convolutional Neural Network (CNN):** In audio processing, we can get a spectrogram by applying Short-time Fast Fourier Transform (SFFT) on a piece of music. A spectrogram can

be thought of as a visual representation of audio across frequency and time dimension. Spectrograms of a song are kind of like an image, each with their distinct patterns. Since CNNs for a long time have been proven to be the undisputed architecture for image classification problems, we use CNNs to process images like audio spectrograms. CNN's assume features that are in different levels of hierarchy and can be extracted by convolutional kernels. CNN learns hierarchical features to achieve a given task during supervised training by learning appropriate kernels/filters and using a set of convolution operations. For example, learned features from a CNN that is trained for genre classification exhibit low-level features (e.g., onset) to high-level features (e.g., percussive instrument patterns).

For training the CNN, we extract the spectral features of the audio by computing the Mel spectrogram using Librosa. We use a kernel size of 3×3 throughout and experiment with the number of kernels in different layers of the network. We use the ReLU activation function for the hidden layers and softmax function for the output layer. We use categorical cross-entropy function as our loss function. To prevent overfitting, we further use early-stopping and dropout as our regularization strategies.

5 Results

As mentioned in the previous section, we use selective features for the different classifiers. For KNN, the algorithm suggests the following best features; chroma STFT, root mean square, zero-crossing rate, tempo, mfccs, and beats. We also use a grid search for the optimum value of K starting from 5 to 25 and see that $K = 5$ gives the best result. With the selected features we obtain a prediction accuracy of 76% and an F1 score of 0.76. The confusion matrix is shown in Table 1.

For GMM, the algorithm selects the following features; chroma STFT, tempo, spectral centroid, zero-crossing rate and mfccs. We experiment with all 4 types of the covariance matrix and as shown in Table 2. As seen in Fig. 1, the best accuracy is obtained using the spherical and diagonal covariance types with F1 scores of 0.4.

For SVM, the algorithm selects spectral centroid, spectral roll-off, chroma STFT, mfccs, and beats as the features. We further do a grid search to obtain the optimal values for the regularization parameter, C, and the gamma parameter for RBF kernel. Using the optimal values; gamma as 0.1 and C as 1.5, we get the best result using the RBF kernel with an accuracy of 84.09% as shown in Table 3. The confusion matrix using RBF Kernel is shown in Table 4.

For the neural networks, we experiment with a variety of networks using the Mel-spectrogram feature vectors as discussed above and as demonstrated in the jupyter notebook; a simple Feed-Forward Network (FFN)/ Multi-layered Perceptron (MLP), a relatively complex CNN, and a very complex CRNN. The results are shown in Table 5. As expected, the vanilla FNN underfits the data with an accuracy of only 35% (Fig: 3a), the complex CRNN on the other hand seems to overfit due to its model complexity with a marginal improvement over the FNN with 32% accuracy (Fig: 3b). The CNN model turns out to be the best model with an accuracy of 76%. It finds the optimal balance between the model complexity and model performance, and hence we report the CNN model for evaluation. The confusion matrix for CNN classifier with early-stopping and dropout is shown in Table 6. We also train a simple FFN/MLP with the same 57 feature dataset that we train the machine learning models with, and report an accuracy of 70.5%. We have shown the CNN's training learning curve with 75:25 split (Fig: 2a) and a 80:20 split (Fig: 2b). The accuracy of the CNN further improves to 80% when using the 80:20 train-test split, as shown in Table 7.

Analysing the confusion matrices from our experiments, we further infer that in general the F1 score of the genre Hi-Tech is relatively low when compared to other genres, that is Hi-Tech is comparatively challenging to classify. The genres Goa and Dark-Forest have the best F1 scores of the four genres.

Table 1: Confusion matrix for KNN classifier ($K = 5$)

	Precision	Recall	F1-score	Support
Dark Forest	0.79	0.72	0.75	36
Hi Tech	0.69	0.67	0.68	43
Full On	0.69	0.95	0.80	43
Goa	0.90	0.70	0.79	54
Accuracy			0.76	176
Macro avg	0.77	0.76	0.76	176
Weighted avg	0.78	0.76	0.76	176

Table 2: Confusion matrices for GMM classifier with different covariance.

Covariance Type		Precision	Recall	F1-score	Support
Spherical	Dark Forest	0.57	0.57	0.57	42
	Hi Tech	0.29	0.19	0.23	43
	Full On	0.32	0.26	0.29	47
	Goa	0.39	0.61	0.48	44
	Accuracy			0.40	176
	Macro avg	0.39	0.41	0.39	176
	Weighted avg	0.39	0.40	0.39	176
Diagonal	Dark Forest	0.54	0.74	0.63	42
	Hi Tech	0.05	0.02	0.03	43
	Full On	0.39	0.26	0.31	47
	Goa	0.40	0.61	0.49	44
	Accuracy			0.40	176
	Macro avg	0.35	0.41	0.36	176
	Weighted avg	0.35	0.40	0.36	176
Tied	Dark Forest	0.43	0.69	0.53	42
	Hi Tech	0.00	0.00	0.00	43
	Full On	0.50	0.32	0.39	47
	Goa	0.15	0.14	0.14	44
	Accuracy			0.28	176
	Macro avg	0.27	0.29	0.27	176
	Weighted avg	0.27	0.28	0.27	176
Full	Dark Forest	0.59	0.57	0.58	42
	Hi Tech	0.14	0.14	0.14	43
	Full On	0.69	0.19	0.30	47
	Goa	0.28	0.50	0.35	44
	Accuracy			0.35	176
	Macro avg	0.42	0.35	0.34	176
	Weighted avg	0.43	0.35	0.34	176

Table 3: Prediction accuracy of SVM with different kernels

Kernel type	Accuracy (%)
Linear	75.56
Radial basis	84.09
Polynomial	72.15
Sigmoid	63.06

Table 4: Confusion matrix for SVM classifier with Radial basis kernel.

	Precision	Recall	F1-score	Support
Dark Forest	0.81	0.83	0.82	36
Hi Tech	0.89	0.74	0.81	43
Full On	0.75	0.95	0.84	43
Goa	0.94	0.83	0.88	54
Accuracy			0.84	176
Macro avg	0.85	0.84	0.84	176
Weighted avg	0.85	0.84	0.84	176

Table 5: Prediction accuracy of different neural networks

Kernel type	Accuracy (%)
FFN with dataset's 57 features	70.5
FFN with mel-spectrograms	35
CNN with mel-spectrograms	76
CRNN with mel-spectrograms	32

Table 6: Confusion matrix for CNN classifier (75:25 split) with early-stopping and dropout.

	Precision	Recall	F1-score	Support
Dark Forest	0.89	0.57	0.70	42
Hi Tech	0.75	0.70	0.72	43
Full On	0.69	0.91	0.79	47
Goa	0.79	0.84	0.81	44
Accuracy			0.76	176
Macro avg	0.78	0.76	0.76	176
Weighted avg	0.78	0.76	0.76	176

Table 7: Confusion matrix for CNN classifier (80:20 split) with early-stopping and dropout.

	Precision	Recall	F1-score	Support
Dark Forest	0.87	0.79	0.83	34
Hi Tech	0.66	0.71	0.68	35
Full On	0.83	0.78	0.81	37
Goa	0.84	0.89	0.86	35
Accuracy			0.79	141
Macro avg	0.80	0.79	0.80	141
Weighted avg	0.80	0.79	0.80	141

6 Conclusion and discussion

It is really interesting to see how well the traditional feature-based machine learning models, especially SVM and k-NN perform when compared to the more advanced and complex neural networks. Our hypothesis for the comparatively poor performance of the neural models is credited to the scarcity of training samples for better generalization. We experiment with this by increasing the training samples by 5%, and we see the immediate improvement in CNN's learning and model performance. Figure 2b learns better and converges well when compared to Figure 2a. As discussed in the result section, we also prove that a more complex model like the CRNN, and a very simple model like the FFN, suffer from over-fitting and under-fitting respectively which can be see , and thus the poor performance. Our discussions on the neural models are very much theoretically grounded. We also infer that with good feature engineering, very simple models like the k-NN can perform better. In our experiments, they perform as par with the neural models. SVM with the RBF kernel comes out as the best classifier in our experiments. As there is a certain degree of collinearity between the features for SVM, using a parametric model, like the RBF kernel performs better due to its strong regularization. The improvement in performance using an RBF kernel is an indication that the data just might not be linearly separable. Our experiments also suggest that an appropriate hyperparameter search for the kernel is very important for getting the maximum out of any non-linear SVM kernel. The same applies to every other machine learning and deep learning model in general.

As we append the latent features of the beat vectors using the LSTM Autoencoders to our feature set, we see that the performance of the classifiers doesn't improve much. In fact with the GMM classifier, the performance drops by 3%, which is due to the additional 30 dimensions, and traditional maximum-likelihood estimation for GMMs shows disappointing performance in high-dimensional settings [16]. Theoretically larger feature sets would need larger training samples to generalize better and our experiments sufficiently adhere to this.

The majority of our results from our experiments are theoretically grounded and we provide succinct explanations throughout. As the future steps, we would like to investigate better feature engineering approaches for more robust machine learning experiments. In the deep learning arena, we would further like to experiment with the neural models with sufficiently more training samples, and we hypothesize the models to converge better and improve the classification performance. We would also like to look into transfer learning approaches for this problem and experimenting with the latest neural architectures.

References

[1] Raul de Araújo Lima et al. "Brazilian Lyrics-Based Music Genre Classification Using a BLSTM Network". In: *ArXiv* abs/2003.05377 (2020).

[2] Changsheng Xu et al. "Musical genre classification using support vector machines". In: *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03).* Vol. 5. 2003, pp. V–429.

[3] Keunwoo Choi et al. "Convolutional recurrent neural networks for music classification". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2017), pp. 2392–2396.

[4] Mingwen Dong. "Convolutional Neural Network Achieves Human-level Accuracy in Music Genre Classification". In: *ArXiv* abs/1802.09697 (2018).

[5] Dan Ellis. "Chroma feature analysis and synthesis". In: *Resources of Laboratory for the Recognition and Organization of Speech and Audio-LabROSA* (2007).

[6] Lin Feng, Sheng-lan Liu, and Jianing Yao. "Music Genre Classification with Paralleling Recurrent Convolutional Neural Network". In: *ArXiv* abs/1712.08370 (2017).

[7] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9 (1997), pp. 1735–1780.

[8] Ali Karatana and Oktay Yildiz. "Music genre classification with machine learning techniques". In: *2017 25th Signal Processing and Communications Applications Conference (SIU)*. IEEE. 2017, pp. 1–4.

[9] T. L. Li, Antoni B. Chan, and A. Chun. "Automatic Musical Pattern Feature Extraction Using Convolutional Neural Network". In: 2010.

[10] Brian McFee et al. *librosa/librosa: 0.8.0*. Version 0.8.0. July 2020. DOI: 10.5281/zenodo.3955228. URL: https://doi.org/10.5281/zenodo.3955228.

[11] Achmad Benny Mutiara, Rina Refianti, and NRA Mukarromah. "Musical genre classification using support vector machines and audio features". In: *TELKOMNIKA Indonesian Journal of Electrical Engineering* 14.3 (2016), pp. 1024–1034.

[12] Sebastian Raschka. "MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack". In: *The Journal of Open Source Software* 3.24 (Apr. 2018). DOI: 10.21105/joss.00638. URL: http://joss.theoj.org/papers/10.21105/joss.00638.

[13] Alexandros Tsaptsinos. "Lyrics-based music genre classification using a hierarchical attention network". In: *arXiv preprint arXiv:1707.04678* (2017).

[14] George Tzanetakis and Perry Cook. "Musical genre classification of audio signals". In: *IEEE Transactions on speech and audio processing* 10.5 (2002), pp. 293–302.

[15] H. Yang et al. "Unsupervised Extraction of Video Highlights via Robust Recurrent Auto-Encoders". In: *2015 IEEE International Conference on Computer Vision (ICCV)* (2015), pp. 4633–4641.

[16] Yang Zhao, Abhishek K. Shrivastava, and Kwok Leung Tsui. "Regularized Gaussian Mixture Model for High-Dimensional Clustering". In: *IEEE Transactions on Cybernetics* 49 (2019), pp. 3677–3688.

Appendix

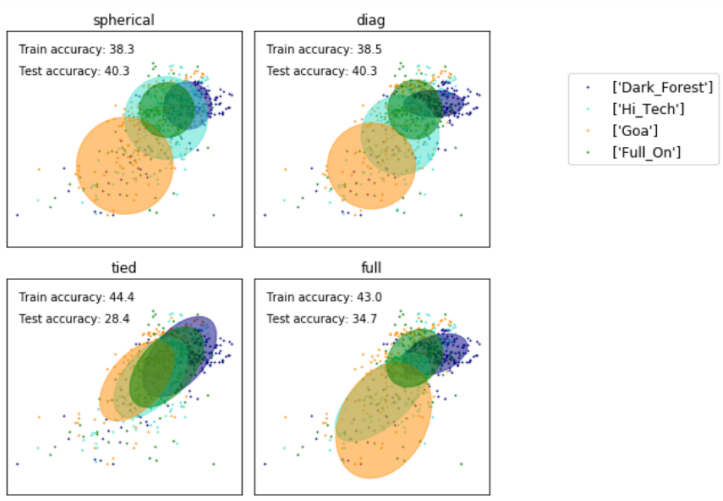
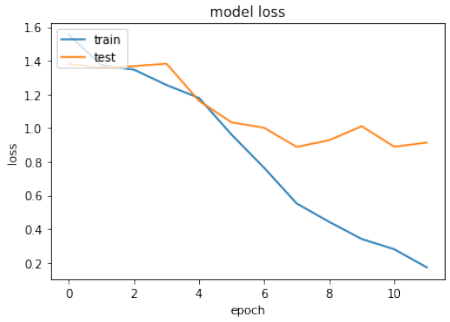
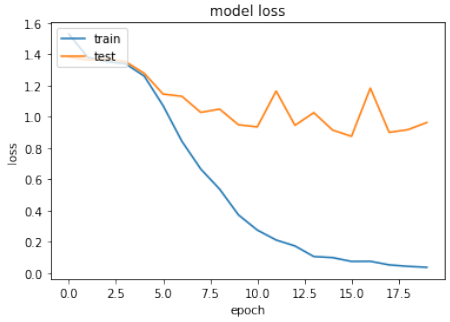


Figure 1: Plot showing the results of GMM clustering when the cluster centers are initialized using individual class centers

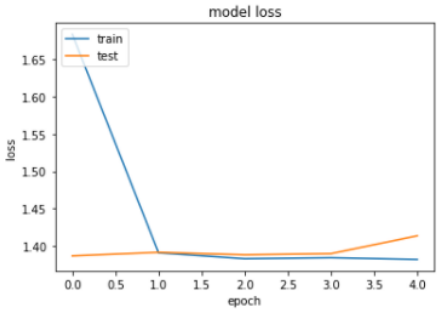


(a) CNN's training learning curve with 75:25 split (Accuracy:76%)

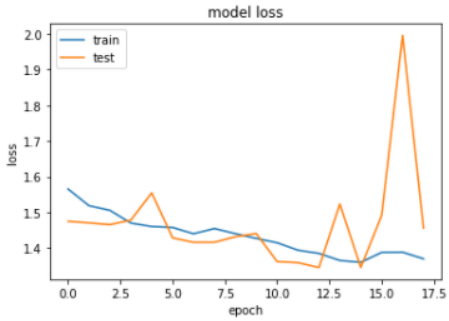


(b) CNN's training learning curve with 80:20 split (Accuracy:80%)

Figure 2: CNN's training learning curves



(a) Under-fitting : FFN learning curve (Accuracy:35%)



(b) Over-fitting : CRNN learning curve (Accuracy:32%)

Figure 3: Analysis of model complexity vs model performance from learning curves