Improvement of Music Genre Classifiers under Transformed Audio Data

Reese Madsen, Reza Alishahi, Kevin Smith, Quang Vu

October 2023

1 Introduction

How well does transformed audio data perform on music genre classifiers? Dynamic range compression is applied to the data and the accuracy is calculated. Will the model performance improve or worsen? Compression brings the loudest and quietest parts of the song closer together so there is more of a uniform gain throughout the audio signal. Our results show that testing on the compressed data improves the accuracy on all models. Testing on light compression yields 9.6% improvement in accuracy score for SVM, 7.8% improvement for SLA, and 5.3% improvement for PLA. Testing on heavy compression yields 9.6% improvement in accuracy score for SVM, 1.6% improvement for SLA, and 5.3% improvement for PLA. Thus we have shown that accuracy on music genre classifiers can be improved by applying dynamic range compression to the data.

2 Literature survey

With the advent of online music platforms, there has been an exponential increase in the daily generation of music data. These platforms are compelled to use recommender and search systems that effectively categorize, tag, and recommend music tailored to user preferences, notably based on their favorite genre. The enormity of the data makes manual genre labeling daunting and impractical. Hence, in recent decades, there's been a concerted effort among researchers to devise automated genre classification methods, taking advantage of various emerging machine learning and deep learning algorithms. Studies carried out on machine learning based music genre classifiers can be divided into two broad categories, namely methods based on traditional machine learning algorithms, and methods that use deep learning algorithms.

Traditional machine learning models can be effective classifiers for music genre classification particularly for small-size datasets, but they oftentimes need extensive feature engineering and considerable domain expertise in order to demonstrate good performance. In one of the earliest studies Tzanekis and cook [1] extracted short time Fourier transform (STFT) and MFCC at the frame level from music audio files, then calculated mean and variance of these features across different frames to form temporal features, and combined them with beat and pitch features to form the final feature vector. They trained Gaussian mixture model (GMM) and KNN classifiers on this feature vector and were able to achieve 61% genre classification accuracy with the GMM classifier. They later published their curated dataset as GTZAN which became the benchmark that most researchers use for music genre classification tasks. One of the most widely used classifiers for music genre classification is support vector machine (SVM) [3, 4, 6, 5, 8]. Genre classification accuracy rates achieved by studies that used SVM classifiers ranges from 72% [3] to 90% [8] depending on data quality, combination of feature sets used, and model parameters. Fu et al. [8] extracted various features at low-level (MFCC, ASE, OSC), temporal features (TMFCC, TASE, TOSC), and mid-level features (Beat, Chord) from GTZAN dataset. Then they used multiple feature combination methods either at feature level (feature concatenation, kernel average, and multiple kernel learning "MKL"), or at decision level (majority voting, sum rule, and stacked generalization "SG") to come up with best combined feature vector based on which SVM learners could be trained. They were able to achieve the highest accuracy rate (90%) with SG and MKL fusion methods. There are also other studies that have used less common classifiers such as AdaBoost, nearest centroid (NC), and sparsed representation classifier(SRC) [2, 7, 9]. Among these studies Panagakis and Kotropoulos [9] were able to achieve very high accuracy of almost 94% on the GTZAN dataset. They came up with very high dimensional crotical representation (CR) features inspired by human auditory system based on which sparsity representation classifiers (SRC) were trained. However, the performance of this model was not tested with other commonly used features and their combinations.

In the last recent years adoption of deep learning based methods for music genre classification has gained traction among researchers. Sarkar and Saha [10] used empirical mode decomposition (EMD) for the decomposition of audio signals, then based on this transformed feature vector trained multi-layer neural networks which achieved a test accuracy of 97.7% for the architecture with two hidden layers. Choi et al. [11] adopted a transfer learning approach, in which they trained a convolutional neural network (CNN) on the source task and then used this pre-trained CNN as a feature extractor on the target task that that fed into the SVM classifier. This model was able to achieve a genre classification accuracy of 89.8% on GTZAN used in the target task. Yu et al. [13] developed an attention-based model with bidirectional recurrent neural networks (BRNN) with parallel as well as serial attention architecture using linear as well as CNN-based attention models. These models were trained on STFT spectrogram features. Researchers in this study conducted music classification experiments with Extended Ballroom and GTZAN datasets and the parallel attention architecture was able to reach an accuracy of 92.7%. Deepak and Prasad [12] developed a hybrid LSTM-SVM model with pre-trained LSTM used as a feature extractor, the output of which is fed into the SVM classifier. This hybrid model was trained on MFCC features and was able to achieve an impressive accuracy of 98.2% on GTZAN dataset.

3 Audio Transformations

3.1 Dynamic Range Compression

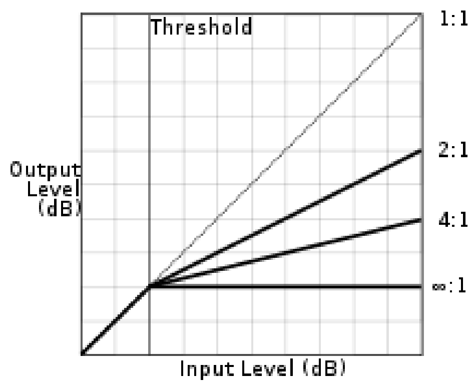
Compression is a technique in audio engineering that reduces the difference between the loudest and quietest part of the song. This is done to add dynamics and movement to an audio signal as well as balancing and keeping the signal consistent in level. Compression is used in several aspects of creating a song including vocals, instruments, and on the overall master track. After applying this effect, the signal sounds more controlled. The inputs of compression include threshold and ratio. The threshold is the level at which the compression begins to occur. The ratio is the amount of gain reduction that is applied when the signal passes threshold. The ratio is defined as follows:

$$R = \frac{x - T}{y - T} \text{ for } x > T \tag{1}$$

where x is the input gain, y is the output gain, and T is the threshold. Thus we can write the output gain as the following piece wise function:

$$y = \begin{cases} x & \text{if } x \leq T, \\ T + (x - T)/R & \text{if } x > T, \end{cases} \tag{2}$$

If the ratio is 2:1 then for every 2 dB the signal exceeds the threshold the compressor will reduce the gain by 1 dB. We add heavy and light compression to all the audio data. For the heavy version we set the threshold to 20db and the ratio to 10:1. For the light version we set the threshold to 20 db and the ratio to 2:1. A graph of the dynamic range compressor can be represented as below.



4 Data

4.1 Dataset

We use the GTZAN data set and its transformations. The original data set contains 1000 30 second song snippets split into 10 genres including hip hop, blues, classical, metal, rock, pop, reggae, country, disco, and jazz. The task is to correctly predict the genre for each song. We transform the data by applying audio engineering techniques used in producing and song writing. Heavy compression, light compression, highpass equalization, and lowpass equalization are applied to each wav file and we obtain 4 new data sets. The transformed data is used for testing to see how the model accuracy changes in the presence of these audio engineering effects.

4.2 Features

4.2.1 MFCC

One of the main features we are going to extract is MFCCs. Mel frequency cepstral coefficients (MFCCs) are one of the most widely used features in music genre classification tasks. MFCCs can be used for audio similarity features and timbral description for music. Timbre is the character of a sound and is represented by overtones on a frequency spectrum. A guitar and a piano sound different even if they place the same frequency say 440Hz. MFCCs will be able to differentiate the timbre from different musical instruments for instance the guitar and piano even if they are playing the same note. This is useful for music genre classification because the overall timbre will be different for a hip hop song vs. a jazz song. There are several steps in computing MFCC:

1. Take the discrete Fourier transform of a signal.
2. Take the log-amplitude of the power spectrum.
3. Change to mel scale.
4. Take the discrete cosine transform and compute power.
5. MFCCs are the amplitude of the resulting spectrum (cepstrum).

Here is the reason for the log amplitude. Let $y(n) = x(n) * h(n)$ be a signal where $y(n)$ is the convolution of two signals. To separate $y(n)$ we need to use the log amplitude after we take the discrete Fourier transform:

$$log[F(y(n))] = log[F(x(n) * h(n))] = log[F(x(n)) \cdot F(h(n))] = log[F(x(n))] + log[F(h(n))] \tag{3}$$

Thus we have separated the signal $y(n)$. Now mel scale is meant to mimic the human auditory experience. Here is the conversion from hertz to mels:

$$m = 2595\log(1 + \frac{f}{700}) \tag{4}$$

where f has hertz units. This conversion is meant to resemble the fact that humans hear things logarithmically. The discrete cosine transform is a type of Fourier transform. The benefit over a normal discrete Fourier transform is that it only gets real valued coefficients, reduces the dimensionality, and is computationally inexpensive. Finally, the cepstrum is a play on letters of the word spectrum. We use cepstrum to resemble the spectrum of a spectrum since we take the discrete Fourier transform and then the discrete cosine transform.

4.2.2 STFT

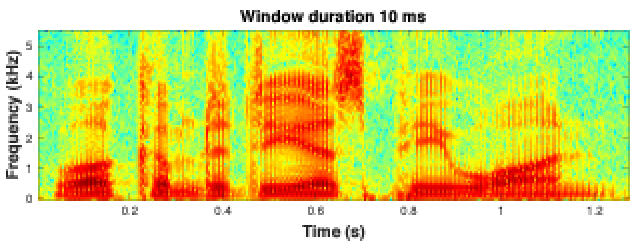
Fourier transforms average the frequency components of an audio signal over time. However to examine music we need a way to look at the frequency components over time without averaging. This is the motivation behind the short-time Fourier transform. It is called short-time because we look at the a short duration of the total signal and compute the Fourier transform. The original signal is x and we define a window function w to represent that we wish to take a short time. A common window function is a rectangular window where the short time is extracted without any changes. Let ω be the frequency variable. The STFT is given by

$$X(n, \omega) = \sum_{m=-\infty}^{\infty} w[n-m]x[m]e^{-j\omega m} \tag{5}$$

where n is discrete and ω is continuous. In practice only a finite number of frequencies are used so we define N to be the number of discrete frequency channels. Define N_w to be the length of the window function. The frequencies become $\omega_k = 2\pi k/N$ where k is the frequency index. The duration of the window is finite with values n from 0 to $N_w - 1$. The updated equation is

$$X(n, k) = \sum_{m=n-(N_w-1)}^{n} w[n-m]x[m]e^{-j\omega_k m} = \sum_{m=n-(N_w-1)}^{n} w[n-m]x[m]e^{-j2\pi mk/N} \tag{6}$$

Now the STFT is a function of time and frequency and will give us a spectrogram as shown below.



5 Classifiers

5.1 SVM

SVMs classify N-dimensional data using optimal hyperplanes based on support vectors. The aim is to maximize the margin between different classes. The support vectors come from the data points closest to the hyperplane. For the GTZAN dataset, the goal is to classify songs based on MFCC means and variances for SVM. For our SVM model we use a RBF kernel with a C value of 10 and Γ = scale. RBF is radial basis function given by

$$\exp(-\Gamma||x - x^{'}||^2) \tag{7}$$

where $\Gamma > 0$. A high C value aims to classify the data accurately i.e. overfitting. The Γ parameter determines how much weight an entry of the data has with respect to the SVM model. Here $\Gamma = 1/(n \cdot \sigma)$ where n is the number of features, and σ is the standard deviation of the data.

5.2 Deep attention-based classifiers

5.2.1 RNN background

Recurrent neural networks (RNNs) have well-suited architecture for the analysis of sequential data. These models get inputs at time t (x_t) and hidden states at the previous time step (h_{t-1}) as arguments and apply relevant activation function (ϕ) to update the hidden state in the current time step (h_t) as shown in eq.(8).

$$h_t = \phi(h_{t-1}, x_t) = \phi(Wx_t + Uh_{t-1} + b) \tag{8}$$

However, due to their long chain of gradients, RNNs suffer from the vanishing gradient problem in the training process. Improved versions of RNNs were later developed that rectify this problem. Long Short-term Memory (LSTM) and Gated Recurrent Unit (GRU) networks are descendants of such networks that use gating mechanisms to address the vanishing gradient problem. Among the two GRU is more recent and computationally more efficient because it uses two gates instead of three in the case of LSTM while this does not have significant adverse impact on its performance when compared to LSTM.

5.2.2 Overall approach

The music genre classification task can be broken down into two major components namely the encoder and the decoder. In the encoder part, a model needs to be trained on the input features (in this case STFT features) and encode them into vector representations, then the decoder component decodes these vectors into class/genre labels. This approach can be summarized in eq.(9)

$$\begin{aligned} h &= \text{Encoder}(X) \\ \hat{y} &= \text{Decoder}(h) \end{aligned} \tag{9}$$

Deep attention-based neural networks in this project are implementations of the three high performing models in the study undertaken by Yu et al.[13]

5.2.3 GRU structure

In this project, bidirectional GRU was used as the encoder backbone of all three deep attention-based models due to its relative computational efficiency compared to LSTM as well as encoding capabilities on sequential features such as STFT. In GRU architecture, the hidden state at current time h_t is a function of the candidate state $\widetilde{h_t}$ and the hidden state at the previous time step h_{t-1} by modulating the update gate u_t as shown in eq.(10). In this equation, $\odot$ stands for Hadamard product operation.

$$h_t = u_t \odot \widetilde{h_t} + (1 - u_t) \odot h_{t-1} \tag{10}$$

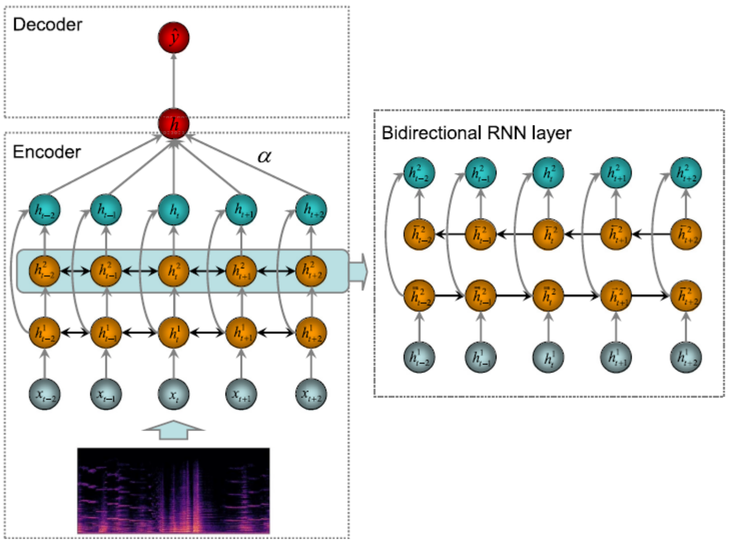
Candidate state $\widetilde{h_t}$ and update gate each are obtained using the input x_t, reset gate r_t, and hidden state at the previous time step h_{t-1} eqs.(11, 12). The reset gate r_t is obtained by combining x_t and h_{t-1} eq.(13) and acts like a gate that controls to what degree h_{t-1} effects $\widetilde{h_t}$ and as a result h_t. In these equations, W_h, U_h, W_u, U_u, W_r, and U_r are all weight matrices and parameters that are learned during the training process.

$$\widetilde{h_t} = \tanh(W_h x_t + U_h(r_t \odot h_{t-1})) \tag{11}$$

$$u_t = \sigma(W_u x_t + U_u h_{t-1}) \tag{12}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{13}$$

Figure below shows the architecture of bidirectional RNN used in this project. In this encoder block two BRNN layers are stacked together. A BRNN is a combination of two RNNs one which moves forward and another one that moves backward the direction of sequential data. The final output of such network is the concatenation of forward and backward passes in other words $h_t = h_t^1 \oplus h_t^2$. In extracted STFT features there are 128 timesteps, thus the output of this network is going to be of length 256.



5.2.4 Attention mechanism

To improve the performance of the encoder, various attention mechanisms were employed. The role of attention is to take in all the temporal steps and assign various attention scores (weights) to those steps, in the next stage attention probabilities are calculated, and later on a classifier assigns the labels. These steps can be summarized in eq.(14) in which $e = \{e_1, e_2, ..., e_T\}$ is the attention score vector and f_{att} is the attention function applied to the sequential data. Next probability vector α_i is generated using the softmax function.

$$\begin{aligned} e_i &= f_{\text{att}}(h_i), i \in \{1, 2, ..., T\} \\ \alpha_i &= \text{Softmax}(e) = \frac{\exp(e_i)}{\sum_{j=1}^{T} \exp(e_j)} \end{aligned} \tag{14}$$

5.2.5 Serial attention model

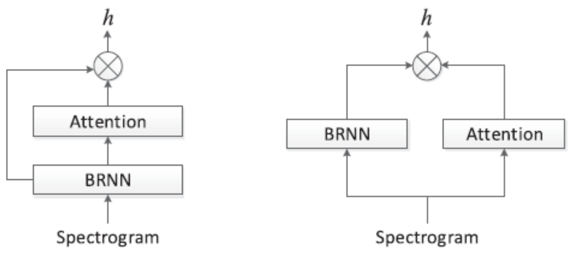
In the serial attention model (SAM) the scores are calculated by applying f_{att} to the output of the BRNN-GRU encoder. The architecture of this model is shown in the left schematic of Figure below. The attention function used for this model is the linear transformation function shown in eq.(15).

$$e_i = f_{\text{att}_s}(h_i) = s_i^T h_i, i \in \{1, 2, ..., T\} \tag{15}$$

5.2.6 Parallel attention model

This architecture is shown in the right schematic of Figure below. Two variants of this model were implemented in this project. The first one was parallel linear attention (PLA) implementation in which the attention function performs a linear transformation directly on the input features x_t instead of encodings of the BRNN network eq.(16)

$$e_i = f_{\text{att}_p}(x_i) = p_i^T x_i, i \in \{1, 2, ..., T\} \tag{16}$$



In the second variant of this model named PCNNA a convolutional neural network (CNN) was used as the attention function. The CNN attention has the architecture shown in the table below and is applied directly to input features which are 2D arrays of 513x128. As shown in this table the output size of CNN attention is a vector of size 256, which then is multiplied by the output of the BRNN encodings, and then goes through the softmax function to generate probabilities.

Type	Filter Shape	Stride and padding	Input shape
Conv	3x3x1x16	1x1, 1x1	1x513x128
BN & ReLU	-	-	16x513x128
MaxPooling	Pool 2x2	2x2, -	16x513x128
Conv	3x3x16x32	1x1, 1x1	16x256x64
BN & ReLU	-	-	32x256x64
MaxPooling	Pool 2x2	2x2, -	32x256x64
Conv	3x3x32x64	1x1, 1x1	32x256x64
BN & ReLU	-	-	32x128x32
MaxPooling	Pool 4x4	2x2, -	64x128x32
Conv	3x3x64x128	1x1, 1x1	64x128x32
BN & ReLU	-	-	64x64x16
MaxPooling	Pool 4x4	4x4, -	128x64x16
Conv	3x3x128x64	1x1, 1x1	128x64x16
BN & ReLU	-	-	64x16x4
MaxPooling	Pool 4x4	-	64x16x4
Flatten	-	-	64x4x1
Dense	Output 1xT	-	1x256

5.2.7 Classifier decoder

After generating attention probabilities, these weights α_i are multiplied by the output of the BRNN network h_i eq.(17). Next these outputs are passed through a softmax function which will give us the predicted genre label eq.(18).

$$h = \sum_{i=1}^{T} \alpha_i h_i \tag{17}$$

$$\begin{aligned} p(y|h) &= \text{Softmax}(W_c^T h + b_c) \\ \hat{y} &= \arg\,\max_y p(y|h) \end{aligned} \tag{18}$$

5.3 Data transformation and feature extraction

5.3.1 SVM classifiers

The chosen features, derived from the analysis of feature coefficients, were subsequently extracted from the transformed audio data set using the Librosa library. The mean values of Mel-frequency cepstral coefficients (MFCCs) (specifically, the 6th, 3rd, 4th, 1st, and 19th coefficients) were selected to capture crucial spectral characteristics. Additionally, the variance of certain MFCCs (5th, 4th, and 6th) was included to account for the dynamic nature of the audio signals. The Librosa library's functionality was leveraged to compute the variance of the chroma short-time Fourier transform (STFT), providing insights into tonal content and harmonic characteristics. The mean value of the root mean square (RMS) amplitude, chosen for its significance in indicating average loudness, was also extracted using Librosa. This comprehensive feature extraction process, facilitated by Librosa, ensures a focused and informative representation of essential audio characteristics for subsequent analysis or machine learning applications. In addition to feature extraction, both the original data and the previously transformed data underwent a preprocessing step using a Min-Max scaler. This scaling technique was employed to normalize the data and bring it within a consistent range, typically between 0 and 1. The normalization process is beneficial for several reasons. Firstly, it ensures that features with different scales contribute more uniformly to the analysis, preventing those with larger magnitudes from dominating the learning process. Secondly, normalization enhances the convergence speed and performance of certain machine learning algorithms, as it minimizes the impact of outliers and stabilizes the optimization process during model training. By applying the Min-Max scaler to both the original and transformed data, consistency is maintained, facilitating a more reliable and effective analysis or machine learning model training across datasets with potentially diverse feature scales.

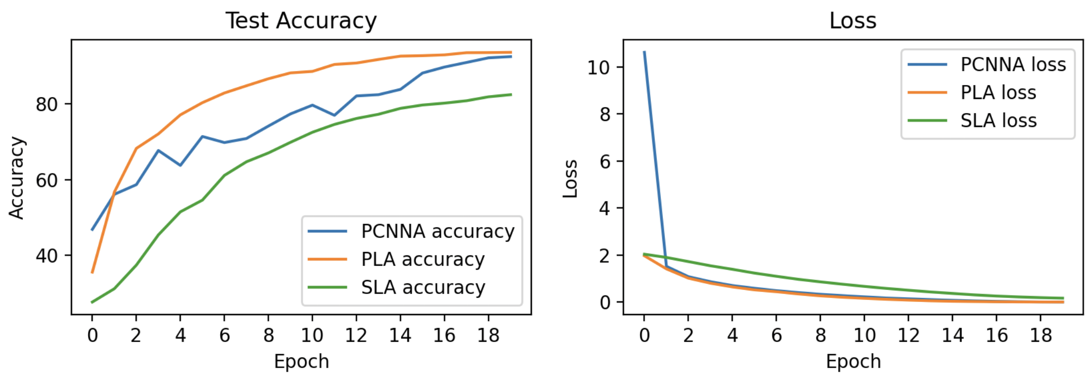
5.3.2 Deep attention-based classifiers

For deep attention-based classifiers, the STFT feature was chosen due to its accurate representation of amplitude variation of different frequency bins across time. The sequential time-dependent nature of this feature makes it suitable for training RNN-based encoders like GRUs which were used in this project. Due large number of parameters of deep attention-based classifiers and a relatively small number of labeled data points, the GTZAN dataset was augmented by dividing the 30s clips into 3s audio clips that had 1.5s (50%) overlap. In this manner for each 30s clip, 19 clips each 3s long were generated, this adds up to about 18981 data samples. Training the deep attention-based classifiers on such a large dataset boosts the performance of these classifiers considerably. Next STFT features were extracted, Librosa python package was used for this task. This package provides a comprehensive toolbox for audio file processing and feature extraction. We used a frame size of 1024 and a hop size of 520 for STFT feature extraction, this process created 2D array features that had 513 rows and 128 columns for each 3s audio file. The number of rows in each feature sample stands for the number of frequency bins present in the extracted feature also the number of columns corresponds to the number of time frames. For labels, we mapped genres (blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, rock) to numbers from 0 to 9. All these features were concatenated together to form an array with shape of (18981, 513, 128) and were compressed and stored in HDF5 file format for smoother loading during the training process.

6 Training on the original data

SVM models were implemented using Scikitlearn library available in python. During training process various kernels were tried during the training process including linear, polynomial, RBF, and sigmoid with different hyperparameters. Eventually, RBF kernel with gamma scale and C=10 was chosen due to higher accuracy rate.

BRNN based deep attention neural networks were implemented using the Pytorch package available for Python. To accelerate the training process Google Colab IDE was used which gives access to GPU training. The HDF5 compressed features were loaded to Colab notebooks, this data then was scaled using the mean-max scaler tool available in Scikitlearn. Then training process was carried out on V100 GPUs on a cluster with high ram (52GB). During the training process, various hyperparameters were tried, and eventually batch size of 64, a linearly decaying learning rate with an initial value of 0.002, and 20 epochs were chosen for all models. Epochs were chosen to be 20 for all models even though some models might have demonstrated better performance for more epochs, in the interest of time this value was chosen. Overall testing accuracy and training loss, f1 score, and recall were measured over each epoch. Classification accuracy values were also tracked for each genre in the GTZAN dataset. For obtaining all of these values K-Fold (in our case 10 folds) cross-validation was carried out and all of these metrics were averaged across all 10 folds. As seen in Figure below all three deep attention-based models converge relatively well and demonstrate great performance on the original dataset.



7 Results: Testing on original and transformed data

Our results include 3 tables: validation set of the original data on 4 models, validation set of the 2:1 compressed data on 4 models, and validation set of the 10:1 compressed data on 4 models. Also, we include the results of the validation sets of compressed data for each music genre. As expected Deep attention-based models demonstrate superior performance both on the original dataset as well as the transformed datasets. Among these models, BRNN with parallel attention mechanism (PLA) shows the best performance and SVM shows the lowest in terms of all metrics. The BRNN attention-based models also demonstrate a well-balanced accuracy across different genres.

Validation Set			
Model	Recall	F1 Score	Accuracy
SVM	0.62	0.57	0.62
SLA	0.825	0.824	0.825
PLA	0.94	0.94	0.94
PCNNA	0.92	0.92	0.92

2:1 Compression			
Model	Recall	F1 Score	Accuracy
SVM	0.68	0.68	0.68
SLA	0.89	0.89	0.89
PLA	0.99	0.99	0.99
PCNNA	0.98	0.98	0.98

10:1 Compression			
Model	Recall	F1 Score	Accuracy
SVM	0.63	0.63	0.63
SLA	0.94	0.94	0.94
PLA	0.99	0.99	0.99
PCNNA	0.97	0.97	0.97

Our results show that testing on the compressed data improves the accuracy on all models. Testing on light compression(2:1) yields 9.6% improvement in accuracy score for SVM, 7.8% improvement for SLA, 5.3% improvement for PLA, and 6.3% improvement for PCNNA. Testing on heavy compression(10:1) yields 9.6% improvement in accuracy score for SVM, 1.6% improvement for SLA, 5.3% improvement for PLA, and 5.2% improvement for PCNNA. Thus SVM improved the most under the presence of compression. Light compression improves accuracy more than the heavy compression. The heavy compression is rather drastic and changes the audio more than the light compression. PLA and PCNNA provide the highest accuracy despite a greater improvement of SVM under the compressed data.

8 Conclusion

We conclude that applying light compression to the audio data improves music genre classifier accuracy. Heavy compression also improves the accuracy, but not as much as the light compression. The change from the original to light is noticeable, but not drastic. The dynamics are controlled making the audio signal less random which in turn improves the classifier performance. In my experience applying compression to the master channel of a song adds noticeable dynamics when run in parallel with the original signal. These compression settings not only make music sound more interesting but also improves music genre classifier performance which is impressive.

2:1 Compression				
Genre	SVM	SLA	PLA	PCNNA
Blues	.55	.84	.94	0.98
Classical	.97	.93	.98	0.99
Country	.62	.76	.92	0.98
Disco	.65	.76	.91	0.98
Hiphop	.35	.84	.94	0.98
Jazz	.9	.86	.95	0.99
Metal	.85	.91	.96	0.99
Pop	.67	.83	.94	0.98
Reggae	.69	.80	.95	0.98
Rock	.6	.71	.89	0.95

10:1 Compression				
Genre	SVM	SLA	PLA	PCNNA
Blues	.44	.84	0.99	0.97
Classical	1.0	.93	1.0	0.99
Country	.58	.76	1.0	0.97
Disco	.59	.76	0.99	0.96
Hiphop	.23	.84	0.97	0.96
Jazz	.9	.86	1.0	0.99
Metal	.83	.91	0.99	0.98
Pop	.5	.83	0.99	0.97
Reggae	.69	.80	0.98	0.97
Rock	.62	.71	0.99	0.96

References

[1] 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. ISSN: 1063-6676.

[2] James Bergstra et al. “Aggregate features and a da b oost for music classification”. In: *Machine learning* 65 (2006), pp. 473–484. ISSN: 0885-6125.

[3] Tao Li and Mitsunori Ogihara. “Toward intelligent music information retrieval”. In: *IEEE Transactions on Multimedia* 8.3 (2006), pp. 564–574. ISSN: 1520-9210.

[4] Michael Mandel and Dan Ellis. “Song-level features and svms for music classification”. In: *In Proceedings of the 6th International Conference on Music Information Retrieval, ISMIR.* Vol. 5. 2006.

[5] Ioannis Panagakis, Emmanouil Benetos, and Constantine Kotropoulos. “Music genre classification: A multilinear approach”. In: *ISMIR.* 2008, pp. 583–588.

[6] George Tzanetakis. “Marsyas-0.2: a case study in implementing music information retrieval systems”. In: *Intelligent Music Information Systems: Tools and Methodologies.* IGI Global, 2008, pp. 31–49.

[7] Chang-Hsing Lee et al. “Automatic music genre classification based on modulation spectral analysis of spectral and cepstral features”. In: *IEEE Transactions on Multimedia* 11.4 (2009), pp. 670–682. ISSN: 1520-9210.

[8] Zhouyu Fu et al. “On feature combination for music classification”. In: *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshop, SSPR&SPR 2010, Cesme, Izmir, Turkey, August 18-20, 2010. Proceedings.* Springer, 2010, pp. 453–462. ISBN: 3642149790.

[9] Yannis Panagakis and Constantine Kotropoulos. “Music genre classification via topology preserving non-negative tensor factorization and sparse representations”. In: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing.* IEEE, 2010, pp. 249–252. ISBN: 1424442958.

[10] Rajib Sarkar and Sanjoy Kumar Saha. “Music genre classification using EMD and pitch based feature”. In: *2015 Eighth International Conference on Advances in Pattern Recognition (ICAPR).* IEEE, 2015, pp. 1–6. ISBN: 1479974587.

[11] Keunwoo Choi et al. “Transfer learning for music classification and regression tasks”. In: *Computer Vision and Pattern Recognition* (2017).

[12] S Deepak and B G Prasad. “Music Classification based on Genre using LSTM”. In: *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA).* IEEE, 2020, pp. 985–991. ISBN: 1728153743.

[13] Yang Yu et al. “Deep attention based music genre classification”. In: *Neurocomputing* 372 (2020), pp. 84–91. ISSN: 0925-2312.