AI Music Genre Classification

Jess Winterborne
Mathematics
University of Bristol
Bristol, United Kingdom
bf19082@bristol.ac.uk

Matthew Ediz Beadman
Engineering Mathematics
University of Bristol)
Bristol, United Kingdom
ld18821@bristol.ac,uk

Joe Hulejczuk
Mathematics
University of Bristol
Bristol, United Kingdom
wk19601@bristol.ac.uk

***Abstract*—In this report we built and compared different models for a music classification task based upon the GTZAN dataset. The highest accuracy was 76% using a Convolutional Neural Network. Comparisons with Wav2Vec and simple neural networks along with future suggestions are discussed. We also explored where our models fell short and possible reasons why.**

I. Intro

A. Motivation

Machine learning is a field of study that has seen rapid development in recent times and exists in most of our daily lives already - music classification algorithms are an example of this and come in many different forms. The subject of this report is the building and comparison of models that can accurately define the genre of a piece of music given the audio in a .wav file. This can be a difficult task, even for humans and especially when categorising obscure genres. We were interested to see how well our own algorithms could classify genres and if they would come into the same pitfalls and confusions as humans do.

Music genre classification is a large area of research within machine learning audio processing. It is also an ever changing and growing task as we now have access to tens thousands of genres and counting.
Although this problem has been tackled by tech giants like Spotify and SoundCoud, their models and methods are well kept industrial secrets and are not readily available to the public. There are many applications within the respective services, including similar song recommendations and genre specific playlists. However a well-designed accessible model with good results would be applicable for use by professional musicians, DJ's and also the common music enjoyer.

B. Datasets

Our initial motivation was to test on fringe genres of music - especially electronic music where there is often a lot of debate between which songs fit into which genres. This proved difficult, mostly due to the lack of relevant data sets online as well as blurred lines within human categorisation.

We researched lots of relevant data sets for this project, one being the million song dataset [1]. This could have been a great option as there is obviously an abundance of data, with each song having a myriad of tags and features we could use. However this proved to be a hindrance since there was simply too much data to download and run through models in a reasonable amount of time. Additionally, there was no actual audio files in the data set – it only contained the chosen features for each song. We wanted to work only with audio files as it would allow us to in future test our models on songs outside the data set.

After weighing up our options, we decided the GTZAN data set [2] would be the most ideal for our goals and time frame. It is an infamous data set when it comes to audio processing and was used in one of the original and most well-known papers on genre classification by Tzanetakis and Cook [3]. It is also the most used public dataset for AI applications in genre classification [4].
It includes 10 genres, with a hundred 30-second songs in each. The shorter length allows us to use a decently sized set of data while not taking up too much time in data processing and training models. The data set also includes features for each audio file and mel spectrograms. We decided however to process the data ourselves to extract features, as this would allow us to use songs of our own choice within the models.

C. Project Definition

The goal of this project is to compare different models used to classify genres. We will be evaluating accuracy and run time, as well as comparing the models to a simple artificial neural network (ANN). The inputs to our system will be the signal from the .wav files, or features extracted directly from these signals. Our hypothesis based on the literature below, is that a convolutional neural network (CNN) will perform well on our data and that due to noisy data, regularisation would help to improve our model - especially since a common problem within CNNs is overfitting.

II. Literature Review

Music Genre classification is an area of machine learning that has seen lots of research, particularly since the emergence of more advanced methods. One of the early examples of this was in 2002 by Cook and Tzanetakis [3], who created the GTZAN dataset and introduced 3 sets of features - timbral structure, rhythmic content and pitch content - using Gaussian's mixture model and k-nearest neighbour classifiers.

They achieved 62% accuracy. Changsheng Xu et al. [5] used support vector machines, a statistical learning method useful for pattern recognition (which are effective in high dimensions but have many parameters that need to be set correctly to achieve the best classification).

Increasingly we have seen the use of neural networks, due to their ability to recognise patterns. Tao [6] discussed the use of a restricted Boltzmann machine, whilst Yu et al. [7] discussed the use of a bidirectional recurrent neural network, which connect two hidden layers of opposite direction, not just in one direction like a standard RNN, (achieving 90% accuracy on the GTZAN dataset). Recurrent neural networks are suitable for temporal data as they offer time-related context to the model, however they can struggle to learn long-term patterns due to gradient vanishing and exploding. Rafi et al. [8] achieved 96% accuracy on the GTZAN dataset (with only 23 epochs) by using an Independent Recurrent Neural Network (IndRNN). This approach can solve the problem of gradient vanishing and exploding, using long short-term memory to learn long-term dependencies.

W. Zhang et al. [9] proposed a CNN architecture using residual blocks to prevent vanishing gradient issue and performance degredation, achieving 87.4% accuracy on the GTZAN dataset. M. Athulya and S. Sindhu [10] extracted feature values from spectrograms as input for a 2D CNN with 94% accuracy. CNNs on the whole are extremely efficient and can automatically detect important/representative features without a person's input. The use of a CNN is particularly appropriate when using spectrograms due to its compatibility with image classification.

However CNNs are also prone to overfitting (which we found to be the case). Hinton et al. [11] used dropout layers on the fully connected layers of their model as a form of regularisation, which we will also apply in our model to try and prevent overfitting.

III. Data

A. Understanding the Data

Within the GTZAN dataset, we received 1000 .wav files, as well as a large table of extracted audio data and images of mel spectrograms. This is all useful information to use in our models, however we decided it would be more useful to create our own data from the .wav files as we could generate more information and implement the model on our own songs not in the dataset.
Spectrograms and mel frequency cepstral coefficients (MFCCs) are most commonly used in speech recognition tasks [12]. They're also seeing increasing usage as features in music information retrieval tasks [13], which is why we decided they would be suitable for this problem.
A spectrogram is like a 3D plot of an audio signal. They're created by taking the short time Fourier transform (STFT) of the audio signal over each time window. Time is along the x-axis, frequency along the y-axis and colour represents the amplitude of each frequency at the corresponding time. The amplitude is then changed to a log scale in order to convert to decibels.
The only difference between a spectrogram and a mel-spectrogram, is that the frequency is further converted into a mel scale by a logarithmic transform. The mel scale is not hugely different - it just scales the frequencies in a way that reflects how humans hear frequencies. You can see the difference between the two by comparing Fig. 1 and Fig. 2.

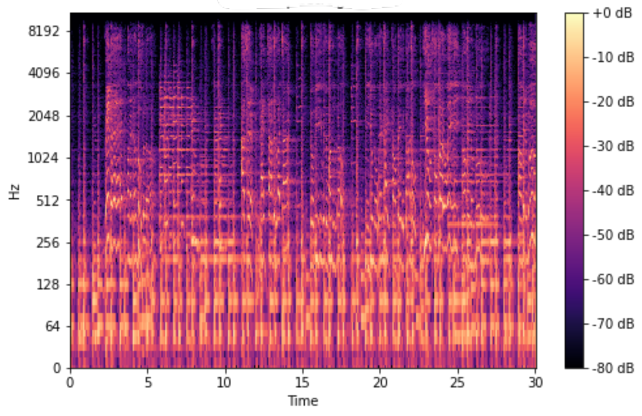

Fig. 1. Examle of a blues spectrogram

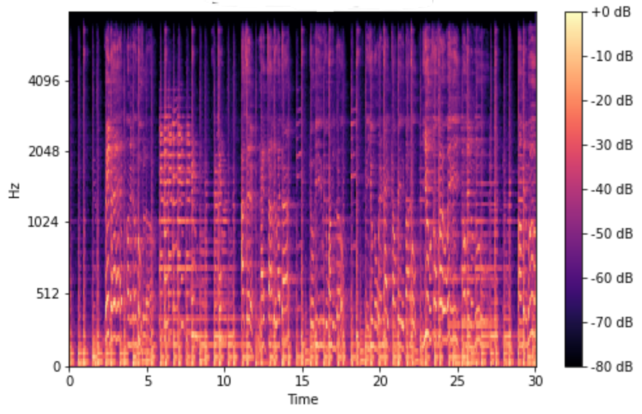

Fig. 2. Example of a blues mel spectrogram

MFCCs are computed from mel spectrograms using the discrete cosine transform. This creates a visual representation over the mel frequencies, rather than time.
We used 13 MFCCs as this is most commonly used in audio recognition and is one of most optimal numbers according to the literature [14]. This means MFCCs contain less data than mel spectrograms - with 13 bands compared to 128. This could have big implications when considering computation times.

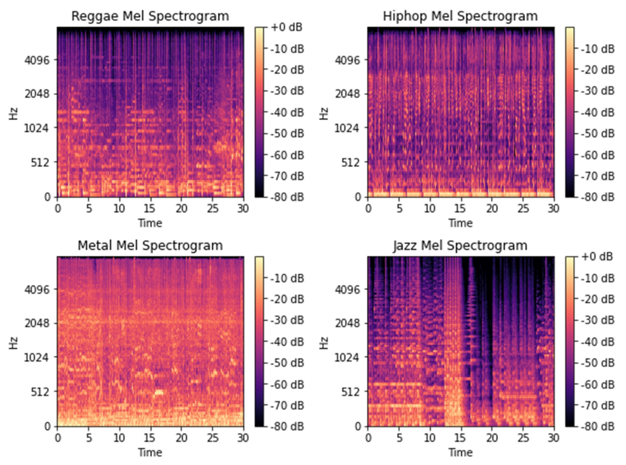

Fig. 3. Comparison of mel spectrgrams of different genres

B. Data Processing

We used the Librosa library [15] throughout the data processing stage as it has many extremely useful features for handling audio data. In this section we will be comparing the performance of the model taking mel spectrograms and MFCCs as input, in order to differentiate their respective performance and computation times. Thus, we built functions in order to extract each.
The mel spectrogram data arrays were size 128 x 1293 and the MFCC data arrays were size 13 x 1293. Due to the MFCCs having much less data (and thus an anticipated shorter computation time), we also decided to build another function to extract the MFCCs in 3 second segments – meaning we could increase the number of data points tenfold.

Within each function we also cleaned the data, as a there were a few discrepancies between some of the sizes of our arrays. This was assumed to be from errors and distortions based on research done on the dataset itself [4]. For the sake of ease, we decided to discard these audio signals from the models which reduced the number of samples from 1000 to 945. Only a minimal amount of other data cleaning was needed since all the songs were of the same length.

After we extracted the data, we normalised the values to range from 0 to 1. Lastly we did a stratified split of the data into training and testing sets, with an 80:20 split respectively. This gave us 755 training samples and 189 testing samples, with all 10 genres in each split. It's important to note that the CNNs needed a third input axis for the colour channel. This would have value 3 for a usual RGB image, however we used value 1 to represent a single colour channel.

IV. Experimentation and Analysis

A. Mel Spectrograms and MFCCs

As explained previously, we initially tested mel spectrograms and MFCCs on convolutional neural networks. We will be analysing how well they perform via accuracy, computation time, and also comparing performance to a simple ANN.
The structure of the ANN used for comparison was an input flattening layer with the input size adjusted for each set of data. This was followed by three dense layers with 512, 256 and 64 neurons respectively. Lastly, we had an output layer with 10 neurons for the 10 different genres, and used a Softmax activation function, because we needed multi-class classification. We also added dropout layers after noticing overfitting. We then used Adam optimisation with a learning rate of 0.0001, sparse categorical cross entropy for loss, and between 25 to 50 epochs depending on the data used. The same optimiser and loss function is used throughout this subsection, as well as Softmax activation for output.

1) Mel Spectrogram: Feeding the data through the ANN we achieved a train accuracy of 95.10% and a test accuracy of 53.54%. There was still some overfitting of the data so we assumed the same would happen for the CNN.
The CNN structure comprised of two 2D convolutional layers, with 16 and 32 filters respectively and each followed by max pooling layers. We then had a flattening layer, followed by a dense layer of 64 neurons, a drop out layer to regularise, and lastly an output layer with 10 neurons. We noticed the model did not learn much more after about 10-15 epochs and each epoch took around 50 seconds. This means the model took around 10 minutes to train. There was noticeably better accuracy using a CNN as expected, with a train accuracy of 94.57%, and a test accuracy of 68.78%.

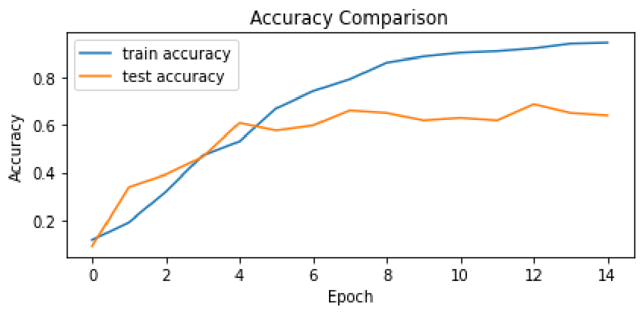

Fig. 4. Test and train accuracy plotted against epochs using mel spectrogram data in a CNN

We can analyse how well the model performed on the different genres by looking at the plotted confusion matrix below in Fig. 5. We found it particularly interesting to see which genres the model mixed up. Country, jazz and rock were often misclassified as blues. This is explainable, due to the influence blues music had on many genres, including the aforementioned. Blues and jazz also have distinctively similar use of piano style. Pop was commonly misclassified as disco, and one might argue that these are similar genres with a definite overlap during certain periods when disco was very popular. One more interesting observation is the consistent

misclassification of rock. Rock has had many influences across an array of genres including blues, country and jazz.
What we find significant here is that the model is mixing up genres similar to how a human would, so must be learning significant and unique characteristics about the music.
Its also interesting to note that the Wav2Vec model below had similar misclassification like disco with pop, and struggles with rock. This suggests that the two models are picking out similar distinguishing factors.

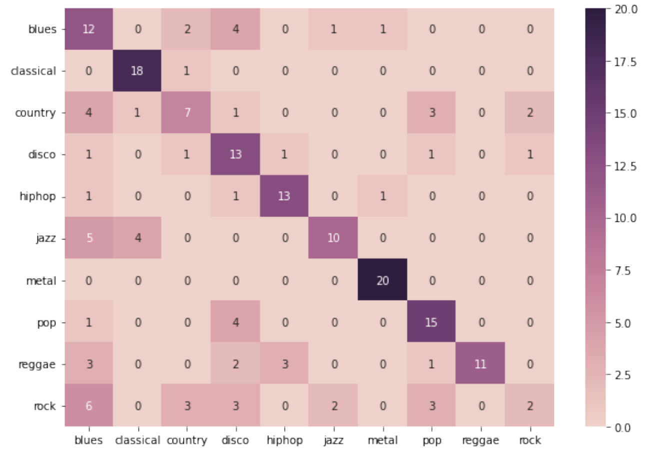

Fig. 5. Confusion matrix for CNN with mel spectrogram data

2) Full - Segment MFCCs: Using the ANN, we achieved a train accuracy of 92.98% and a test accuracy of 50.79%. This is extremely close to the results achieved from mel spectrograms, which is understandable as they contain similar data.
Our CNN had a structure comprising of 3 convolutional layers, each with 32 filters and followed by max pooling and batch normalisation layers. We then flattened the output and fed it into a dense layer of 64 neurons before adding a dropout layer for regularisation. Lastly we had the output layer with 10 neurons.
We achieved a train accuracy of 97.19% and a test accuracy of 54.30%. This is lower than the mel spectrograms, and this can be explained because MFCCs contains less data. Learning did not really improve after about 15-20 epochs and each epoch took about 11 seconds. So the model only took 3-4 minutes to train. Therefore, even though we achieved a slightly lower accuracy than the mel spectrograms, the computation time was significantly lower.

3) 10 - Segment MFCCs: Using 10 segment MFCC data as an input, the ANN achieved a train accuracy of 86.29%, and a test accuracy of 63.51%. As expected this is higher than the full segment MFCCs, but it is also higher than the mel spectrograms which is promising.
We then fed the data through the same CNN as above, and achieved a train accuracy of 83.39% and test accuracy of 75.77%. This is notably higher probably due to the fact that we now have 10x as many data points. Also computation time was still significantly better than the mel spectrograms, with around 13s per epoch. We ran for 50 epochs, but learning did not increase much after around 30.

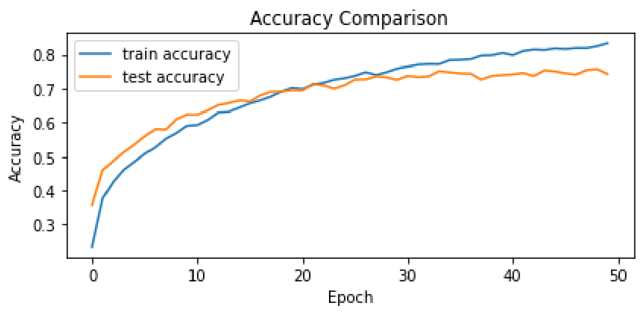

Fig. 6. Test and train accuracy plotted against epochs using 3 second segment MFCC data in a CNN

Once again we can compare performance between the genres and compare against the mel spectrograms. Visually, we can notice how much better this model is performing by comparing the confusion matrices.

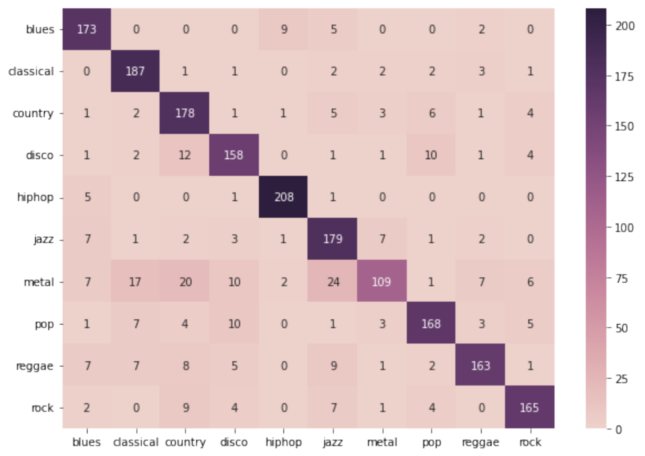

Fig. 7. Confusion matrix for CNN with 3 second segment MFCC data

B. Wav2Vec

Wav2Vec is a state of the art self-supervised speech recognition model developed by Facebook for the purposes of automatic speech recognition, and has achieved impressive results where it achieves a word-error-rate of below 2.5 according to Facebook [16].
Wav2Vec works by extracting complex acoustic features from raw audio data, and extracts many of these features - features extracted from 5 seconds of audio amount to a size of more than 60000, and these features are then transformed via a tokenizer to a string sentence although that section Wav2Vec is beyond the scope of the model being built [17]. The way that these features are extracted is a combination of convolutional neural networks and transformers. Audio clips are broken

into very short segments and operated on individually by the CNN's to output 'latent speech representations'. These latent speech representations are then operated on by a transformer to output a multi dimensional array that is representative of (once the model has been fine-tuned for the task at hand) important and contextual information about the raw audio file.

Wav2Vec is able to learn intrinsically meaningful and contextual representations of audio data with respect to speech recognition after being trained on a data set. To fine-tune a model means to adjust its parameters from an initiated checkpoint according to a slightly different task than it was originally trained for. Conveniently Facebook has many of these checkpoints readily available. The checkpoint we used for finetuning the model is the XLS-R checkpoint with 300 million parameters [18]. This was chosen due to its extensive training in a range of different languages which was hypothesised to make it more applicable to analysing music. The XLS-R model is pretrained on 436000 hours of unlabeled speech in 128 languages [19]. Through fine tuning, the model parameters can be adjusted such that it learns to find contextual representations of audio data with respect to music genres. Similar aims have been achieved before such as in the notebook by m3hrdadfi [20] where Wav2Vec was fine-tuned for the downstream task of emotional recognition in Greek speech with great success.
It should be noted that this model is very computationally expensive to fine-tune, and did not achieve spectacular results partly due to computational limitations, although the results are by no means poor. Due to limitations, we cut the sample sizes from 30 seconds to 10 seconds, as the cost of the model rises quadratically with respect to the length of the audio sample, and training was often cut short because Google Colab terminated GPU access - without which the training would be impossible.

In order for Wav2Vec to be applicable to a classification task we added a simple classification head, which takes output from the transformer layer and runs it through a two layer deep neural network to finally output a 10 length array representative of the corresponding strength of each genre in the audio. For prediction, we used a softmax function to get the index of it's largest value; this index is converted to the genre it corresponds to and is outputted as the prediction. Hyperparameters of the model were set according to experimentation and the literature, where the pooling mode was set to mean and an Adam optimizer implemented a dynamic learning rate initialised at 0.0001. The loss function we used was cross entropy loss, this stays consistent with previous models and is a tried and tested loss function for classification problems with many classes. The model was fine tuned for 6 epochs, and reached an accuracy rate of 74.5% with a cross entropy loss of 0.795. The loss curve of the model is showcased below:

As seen above the accuracy improvements are much more

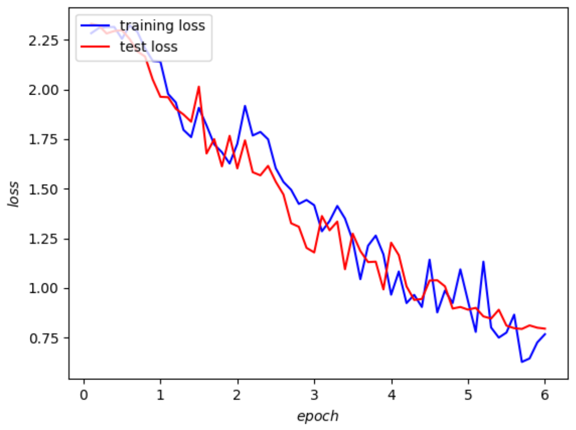

Fig. 8. Training and test loss against epoch

noisy than one would expect and jumps between values after each 10 steps sporadically. This may reflect the nature of transfer learning in this task as the model was never meant to analyse music, but to analyse speech in audio. The trainer may be struggling to find the correct weights for our task but once it has found them can move upwards from there. The results are particularly impressive for a model that was designed to recognise speech and not music, although unfortunately the model produces similar results as the convolutional MFCC model in respect to its weaknesses in classifying pop, rock and country music.

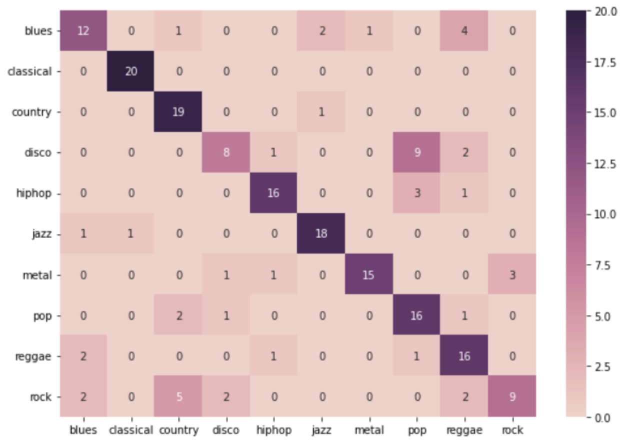

Fig. 9. Confusion matrix for the fine-tuned Wav2Vec model

Most of the technical difficulties in this model originated from the large amounts of memory that the model intrinsically uses which puts a strain on time and resources. The audio clips had to be reduced to 5 seconds for the GPU to have enough memory. Using the more complicated 1 billion and 2 billion parameter Wav2Vec2 XLS-R models proved too expensive for google colabs GPU to achieve. Due to the complexity of the model it takes upwards of half an hour to train each

epoch and this coupled with Google's GPU usage limits, has withheld training the model further and potentially seeing stronger results. The complete training of the model has taken almost four hours. Despite these limitations the model has achieved competitive results.
The model can be further improved for our specific task, one could split the audio into segments, much like the previous model, and operate Wav2Vec on each segment to then output an averaged vector to softmax as this would increase training data and let the model analyse full audio clips. Also with more power it would be feasible to analyse longer lengths of audio which is likely to improve the accuracy. Finally as mentioned before, using the XLS-R trained Wav2Vec with 2 billion parameters would be the optimal solution and this would undoubtedly make the model not only incredibly robust but applicable to a much larger scale of genre classification tasks.

V. Discussion

Although the models achieved their aims, it has not been able to achieve industry level standards, which has been due partly to computational limitations and a lack of experience implementing classification models. One way to improve the models metrics would be to incorporate both the MFCC model and the fine-tuned Wav2Vec into a single unified model. This could be achieved through flattening the outputs of the CNN and concatenating it with the flattened output of the transformer layer from Wav2Vec to then put through a classification head. An attempt to make a unified model was made, however due to programming issues it was never achieved. If we assume an upper bound for the accuracy model is such that it classifies a song correctly if it has been correctly classified by either model then the accuracy of the unified model would be in the range of 80%.
An extension of this approach would be to then use a feature selector, like FeatureWiz [21], on the concatenated array of flattened features to determine the best features to use for our model. Although this would not have a substantial increase in our metrics it would not be negligible, where it has been seen to increase accuracy by 2-3%.

Another interesting addition could be to add the song name, and song artist as features. This would likely not have a noticeable impact for the data set we are using, due to its small size, but on larger data sets with multiple songs from the same artist it isn't impossible for the model to 'learn' an artists preferred genre and weight the song accordingly.
Looking at both models it is fair to say that pop music is a common difficulty which, due to the nature of pop being whatever is popular at the time, is not surprising. Possibly removing pop as a category all together and placing pop tunes into whatever genre suits the song best, apart from pop, could improve accuracy metrics.
It has been seen that both models fall short on classifying disco music. This may be a symptom of the limited data set. Because disco is electronic it is quite possible that the same models trained on a larger data set inclusive of electronic music would be more accurate.

References

[1] Available at: http://millionsongdataset.com/ [Accessed February 2022].
[2] A. Olteanu, "GTZAN Dataset - Music Genre Classification", 2001. Retrieved Feb 2022 from https://www.kaggle.com/andradaolteanu/gtzan-dataset-music-genre-classification.
[3] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," in IEEE Transactions on Speech and Audio Processing, vol. 10, no. 5, pp. 293-302, July 2002, doi: 10.1109/TSA.2002.800560.
[4] B. L. Sturm, "The State of the Art Ten Years After a State of the Art: Future Research in Music Information Retrieval," Journal of New Music Research, vol. 43, no. 2, pp. 147–172, Apr. 2014, doi: 10.1080/09298215.2014.894533.
[5] Changsheng Xu, N. C. Maddage, Xi Shao, Fang Cao and Qi Tian, "Musical genre classification using support vector machines," 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03)., 2003, pp. V-429, doi: 10.1109/ICASSP.2003.1199998.
[6] Feng, Tao. "Deep learning for music genre classification." private document (2014).
[7] Yang Yu, Sen Luo, Shenglan Liu, Hong Qiao, Yang Liu, Lin Feng, Deep attention based music genre classification, Neurocomputing, Volume 372, 2020, Pages 84-91.
[8] Rafi, Quazi Ghulam, et al. "Comparative Analysis of Three Improved Deep Learning Architectures for Music Genre Classification." International Journal of Information Technology and Computer Science 13.2 (2021): 1-14.
[9] Zhang, Weibin, et al. "Improved Music Genre Classification with Convolutional Neural Networks." Interspeech. 2016.
[10] K M, Athulya and S, Sindhu, Deep Learning Based Music Genre Classification Using Spectrogram (July 10, 2021). Proceedings of the International Conference on IoT Based Control Networks Intelligent Systems - ICICNIS 2021, Available at SSRN: https://ssrn.com/abstract=3883911 or http://dx.doi.org/10.2139/ssrn.3883911
[11] Geoffrey E. Hinton et al. "Improving neural networks by preventing co-adaptation of feature detectors." https://doi.org/10.48550/arXiv.1207.0580
[12] T. Ganchev, N. Fakotakis, and K. George, "Comparative evaluation of various MFCC implementations on the speaker verification task," Proceedings of the SPECOM, vol. 1, Jan. 2005.
[13] M. Müller, "Information Retrieval for Music and Motion," 2007. doi: 10.1007/978-3-540-74048-3.
[14] L. Grama and C. Rusu, "Choosing an accurate number of mel frequency cepstral coefficients for audio classification purpose," Proceedings of the 10th International Symposium on Image and Signal Processing and Analysis, 2017, pp. 225-230, doi: 10.1109/ISPA.2017.8073600.
[15] B. McFee, *et al*. librosa/librosa: 0.9.1. [online] Zenodo, 2022. Available at: https://zenodo.org/record/6097378 [Accessed February 2022].
[16] Available at: https://ai.facebook.com/blog/wav2vec-state-of-the-art-speech-recognition-through-self-supervision/ [Accessed February 2022].
[17] Available at: https://arxiv.org/pdf/2006.11477.pdf [Accessed February 2022].
[18] Available at: https://huggingface.co/facebook/wav2vec2-xls-r-300m
[19] Available at: https://huggingface.co/facebook/wav2vec2-xls-r-300m
[20] Available at: https://colab.research.google.com/github/m3hrdadfi/soxan /blob/main/notebooks/Emotion_recognition_in_Greek_speech_using _Wav2Vec2.ipynbscrollTo=sp37lZOV2042
[21] Available at: https://pypi.org/project/featurewiz/