COMP47650 – Deep Learning Project

Music genre classification

Student Zhihang Zhang

Student ID: 20748939

Lecturer: Eoghan Cunningham

UCD School of Computer Science
University College Dublin

May 3, 2024

Table of Contents

Abstract

Music genre classification is one of the ongoing problems in the field of machine learning relating to audio because musical genres can be hard to identify due to their complexity and diversity. This report will focus on addressing challenges relating to music genre classification problems using deep learning models. While this classification problem has been widely addressed by deep learning models like CNN or traditional models like SVM, KNN, etc, this paper used MLP, CNN, and the pretrained model Wav2Vec2 to compare their performance in terms of genre classification. Each model has a distinct architecture; they are trained on different types of data from the GTZAN dataset and compared based on the results.

Chapter 1: **Introduction**

Music genre is defined as the style of music, which is complex because this style can be based on different factors like instruments, timing, etc. Music genres are also used to inspire the musician to develop their new style, and each music genre has its own structure based on different features. So it can be difficult to classify musical genres.[1]

Music genre classification is the task that can automatically identify the genre of given music, which can be useful for applications like music recommendation, and personalization of music services. This tasks can be done using machine learning models that learn extracted features from signals using digital signal processing techniques to predict the music genre. This genre classification task can typically be done using either traditional machine learning models like Knn, SVM, Bayes-NB, Decision tree, or deep learning models like MLP or CNN depending on the format of the extracted feature by the preprocessing technique.[2][3]

GTZAN dataset of music genre classification consists of 1,000 audio tracks, each 30 seconds long, each genre has 100 tracks. The categories of genres are: blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, and rock. Given that the GTZAN dataset of music genre classification provides tabular data of csv files, image spectrograms, and original audio files for all samples,[4][5], I have chosen to use Multilayer perception, Convolutional neural network, and pretrained model "Wav2Vec2" to train each of the different types of data.

MLP can be used to train on tabular data, CSV file contains features of the audio files, in which 57 columns represent the mean and variance computed over multiple features extracted from the audio file for each example, So the MLP can take those 57 features into the input layer and pass through each of the hidden layers and apply an activation function until the output layer produces probability by computed logits and minimizes the loss by backpropagation. [6]

Apart from the tabular data of csv file that can be used for training, each audio file can also be converted to visual representation such as Mel spectrogram image, so CNN can be used to classify image data since it learns spatial hierarchies of features by capturing low-level features like edge and colour in the initial layers and more complex features like textures and patterns in the deeper layers by setting filter size, each convolutional layers followed by pooling layers among to reduce spatial dimensions as well as applying activation function to introduce non-linearity to output in every convolutional layer. Finally, densely connected layers are fed by the flattened matrix to classify images based on the result of high-level features from previous layers.

One of the pretrained models that can be fine-tuned for music classification is Wav2Vec2, The paper written by Alexei et al[7] proposes this powerful framework for self-supervised learning of representations. This model first uses its feature encoder to normalize the raw waveform input to zero mean and unit variance as latent speech representations, followed by feeding the output of the feature extractor into encoder of context network based on transformer architecture and using a convolutional layer acting as embedding, which further processes the latent speech representations to capture better contextual information across the entire sequence. "Its quantization module discretizes the output of the feature encoder into a finite set of speech representations, which can lead to good results in prior work because it learns discrete units in the first step before learning contextualized representations".[7] This model pretrained on 16kHz sampled speech audio, and its pertaining phase involves a masking task that masks some of the time steps in the latent speech representations and "identifies the correct quantized latent audio representation in a set of distractors for each masked time step." [7] This state of the art pretrained model using transfer

learning beneficial from transformer architecture might be useful in terms of genre classification since it has been pretrained to learn the representations of unlabeled audio data by its self-supervised mechanism. So the audio files can be converted into raw waveform input for training this model on downstream task such as audio classification in this case.

Chapter 2: **Related Work**

In the paper introduced by Nitin et al[8], the author pointed out that both CNN and Long Short-Term Memory networks can be used for music genre classification, but CNN is more suitable over LSTM for this task since it is better at "analyzing the temporal features of audio data, detecting local patterns in data, and handling high-dimensional data," while LSTM is good at working with sequential data for tasks like generating new music or predicting future notes in a sequence. So they used CNN model to train on the same GTZAN Music Genre dataset for the genre classification task, the results shows their training, validation and testing accuracies are obtained as 97.43%, 78.64% and 80.5% using the MFCC feature vector respectively. [8]

In the paper introduced by Ahmet et al[3], the author used both traditional machine learning KNN, NB, DT,RF SVM, and deep learning model CNN for music genre classification and compared the performance across those models. The results show that the deep learning model does not show significant performance over other models, and SVM has achieved better performance of over 70% than the CNN of over 60%.

In the paper introduced by Dhevan et al[9], the author trained CNN, MLP of deep learning approaches, and LR, Knn, SVM, and RF of transitional approaches on a 3-second and 30-second input feature set of the GTZAN dataset for genre classification. The result shows that all models using a 3-second input feature set have better performance than those using a 30-second input feature set. In terms of the 30-second input feature set, random forests reached the highest accuracy of 74.5%, followed by support vector machines of 73.5%, KNN of 68.5%, and multilayer perceptrons of 67.5%. CNN trained by spectrograms reaches 66.5%. The result suggests that the amount of data and the different nature of data used for training can directly impact the performance, and number of training epochs for convergence.

The audio course from hugging face [10] shows that the pretrained audio models can be used for audio classification. It used a pretrained model called DistilHuBERT to train on GTZAN dataset, its best evaluation accuracy has reached 83% for just 10 epochs with 899 examples of training data, and it still has room for improvement by training for more epochs, using regularization techniques. This great performance motivated me to test other pretrained models, like Wav2Vec2 in this project.

My work will apply deep learning models based on different architectures on datasets for classification; each model has its own advantages for handling different types of data, which can give a more comprehensive analysis across different deep learning techniques.

Chapter 3: **Experimental Setup**

Some code in the project is referred to pytorch documentation and their official Github [11], and the source also specifically comments in the code.

Mlp setup: This model with a simple architecture will be used as the baseline.

1. Dataset: This model uses the dataset of csv file that contains, for each song (30 seconds long), a mean and variance computed over multiple features that were extracted from an audio file. The dataset is split into test, train, and validation, with 80% for training and validating and 20% for testing.

2. Preprocessing: Read csv dataset using pandas library, leave out "label", "filename", "length" and select the rest of the 57 useful feature columns from dataframe and convert them into numpy array. As well as the labels are converted into numpy array. Since the label is nominal feature, one-hot encoding is used to convert nominal feature into an integer number representing 10 genre classes. The feature array is normalized using L2 normalization, which can help reduce outliers and improve convergence.

3. The architecture of feedforward neural networks: The input layer consists of 57 units, corresponding to 57 input features. There are two hidden layers: the first one has 512 units, followed by batch normalization, ReLU activation, and output of 128 units. Batch normalization normalizes the inputs to each layer within a mini-batch, improving the speed and stability of training, and ReLU activation is applied to introduce non-linearity. The second hidden layer has 128 units, followed by batch normalization, ReLu activation, and dropout, where a random subset of units is set to zero with a probability of 0.5 to prevent overfitting. The output layer consists of 10 units, followed by softmax activation to generate the probabilities of 10 genres. Crossentropyloss is used as a loss function since this loss function is suitable for multi-class classification because it measures the dissimilarity between output and ground truth, and the softmax function is also integrated.

4. Hyper-parameter tuning: Apart from Relu activation, batch normalization, and dropout mentioned, the optimization algorithm used is Adam. This optimization algorithm updates passed parameters based on computed gradients to improve performance; the learning rate of the optimizer used is 0.0001, and the regularization technique "weight decay" is used with a setting 0.00001 that penalizes large weights in the model to prevent overfitting. The batch size is set to 16 for the number of training examples in one iteration.

CNN setup:

1. Dataset: This model uses the dataset of Mel spectrograms png files, The dataset is split into test, train, and validation, with 80% for training and validating and 20% for testing.

2. Preprocessing: [4]Each spectrogram image has a white board at the edge, so I cropped the image into a size of 217x315 (pixels) to remove the white borders before training the model for data enhancement[9]. The original image has 4 channels in 432(width) * 288 (height) pixels, so each image is converted to the image in RGB format, resized to 224 * 224, converted to a PyTorch tensor value in (C, H, W), and normalized to tensors of the range [-1, 1] for three channels.

3. The architecture of cnn networks: There are five convolutional layers and two feed-forward layers; each convolutional layer has a 3*3 kernel size of the filter matrix that slides over the input data and 1 stride moving one pixel at a time, with 1 padding adding around the input matrix, from the start convolving three-channel input image with 8 filters in the first layer to output 128 input channels in the final layer, so 3 channels with 8 filters- > 8 channels with 16 filters-> 16 channels with 32 filters -> 32 channels with 64 filters-> 64 channels with 128 filters -> 128 channels. Max pooling with 2*2 is applied after each convolutional layers to reduce feature map by taking the maximum value in each 2x2 region. The output from the last convolutional layer is flattened having 128 channels $\times$ 7 width $\times$ 7 height to be fed into a feed-forward layer with 128 units, and the output from the previous feedforward layer is passed to the final fully connected layer, which has 10 units for classification.

4. Hyper-parameter tuning: Batch normalization is applied after each convolutional layer , ReLU activation is applied after each convolutional layers and full connected layers, softmax function is used for the final full connected layer. The optimization algorithm used is Adam with setting learning rate of 0.0001, and weight decay of 0.00001. The batch size is 16.

Wav2Vec2 setup:

1. Dataset: This model uses the dataset of original audio files in wav format, the dataset is split into test, train, and validation, with 80% for training and validating and 20% for testing.

2. Preprocessing: Convert the audio file into waveform tensor in shape of [channels, samples], each of waveform tensor is resampled to 16000 to match up the audio sample format of training data of pretraining phase in Wav2Vec2, so this pretrained model can direct take raw waveform tensor as input. Normalize amplitude values to be within [-1, 1] for each raw waveform, and truncate waveform to make sure the same length. Mapping each class name to a unique integer for labels.

3. The architecture of cnn networks: This network consists of pretrained model "wav2vec2 base" and the forward feed network, the pretrained model takes the raw waveform as input and output feature vectors from last hidden status that can be used for downstream tasks like classification in this case. Since the output is a sequence of vectors (one per time step), so pooling method is used that takes the mean value from each feature dimension, this aggregates these to form a single representation of the entire input, then uses the pooled representation as features to train a forward feed network. The forward feed network has three layers, its input layer takes 768 units because the size of the hidden status is 768[12] in pretrained model, the hidden layer has 768 units, and the output layer has 10 for classification.

4. Hyper-parameter tuning: Batch normalization and ReLU activation are applied after hidden layer, softmax function is used for the final output layer. The optimization algorithm used is Adam with setting learning rate of 0.00005, and weight decay of 0.00001. The batch size is 8.

Chapter 4: **Results**

The evaluation metric I have chosen here for comparison is accuracy, this evaluation metric can be calculated by dividing the correctly classified examples by the total of examples. This evaluation metric is perfect for balanced data. my dataset is perfectly balanced, each genre has 100 samples as shown in the figure below. 4.1.

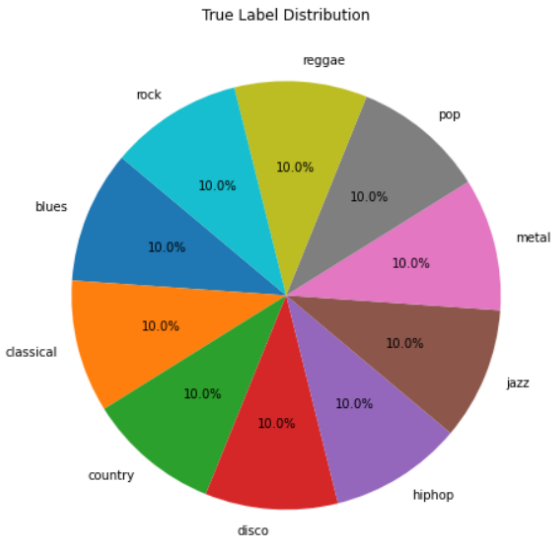

Figure 4.1: data balance check

MLP: The figure4.2 shows the loss over training for mlp, initially the loss decreased sharply for both training and validation, which indicates quick learning in the early epochs. As training progresses, the loss of validation and training both gradually decrease and start to be stuck at an epoch of 350, and the training line starts to diverge from the validation line, which indicates the overfitting issue, it can decrease a bit further by training with more epochs.

Figure 4.2: Loss over Training

The figure4.3 The figure4.3 shows the accuracy over training for mlp, Just like the pattern discovered in the figure of loss over training, the accuracy of both initially improved greatly, then validation started to be stuck at an epoch of 100 and around 0.5 accuracy with high fluctuation. However, training accuracy kept improving over the epochs, which again suggests overfitting and potentially a bit high learning rate.

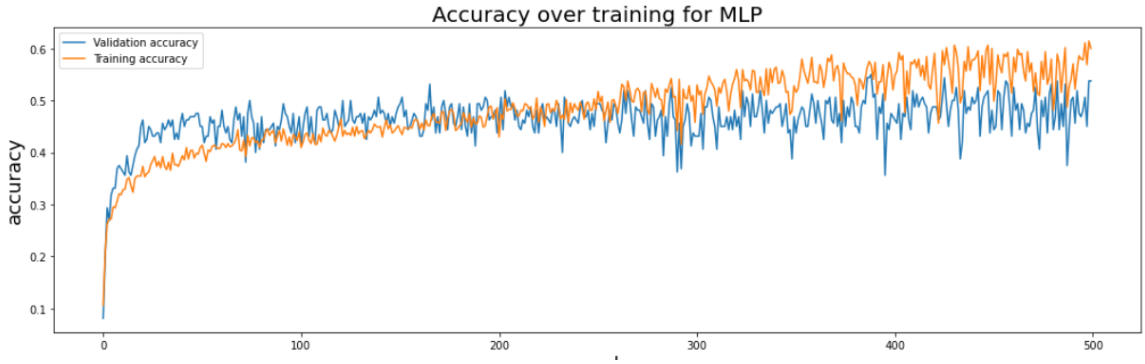

Figure 4.3: Accuracy over Training

This model as the baseline takes more epochs to learn on tabular data for convergence using the same parameter as others. The above result shows that there is more room for improving performance and generalization by adjusting hypsometers.

CNN:

The figure4.4 shows the loss over training for CNN, The training loss decreases steadily as the epochs progress. This indicates that the model is learning well from the training data, approaching 0 loss at the end of the epoch; however, the validation loss initially decreases perfectly, but it is stuck at the epoch of nearing 10 and forms a clear diverging gap between training losses, which indicates severe overfitting.

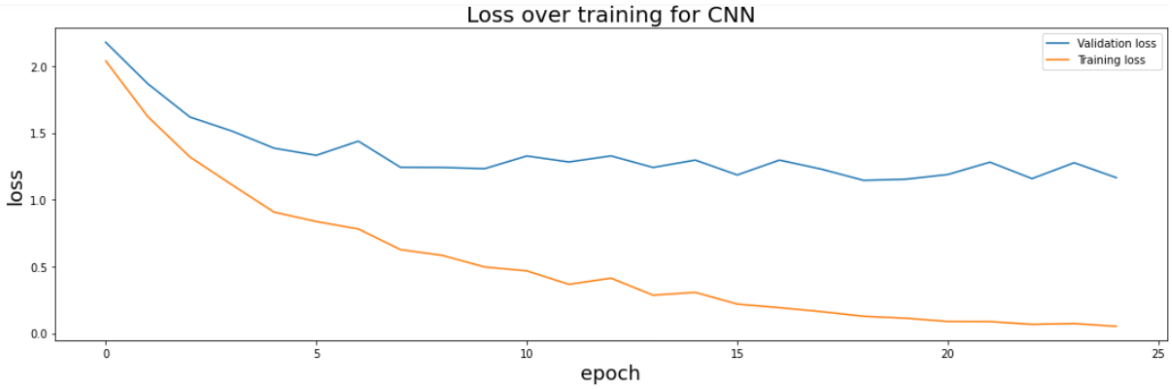

Figure 4.4: Loss over Training

The figure4.5 shows the accuracy over training for CNN, same as the issue and pattern found in the loss over training: training accuracy increases perfectly as epochs progress, but the validation accuracy is stopped at the same level, indicating overfitting.

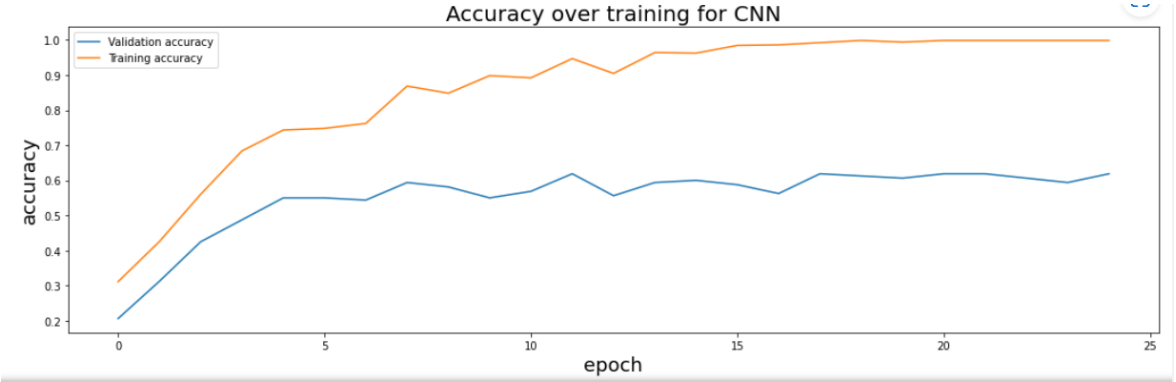

Figure 4.5: Accuracy over Training

The CNN model is way more complex than MLP in terms of its structure of using a convolution layer to extract features from images and a feed-forward network for classification, which leads to better and quicker performance compared to MLP. However, there is clear room for improvement if overfitting can be addressed by further tuning parameters since I have used the regularization techniques of drop, normalization, and weight decay.

wav2vec2:

The figure4.6 and 4.7 shows the loss and accuracy over training and for wav2vec2. The loss of both validation and training slowly decreases with high variance. The training loss shows an extreme decrease at the epoch of 9 and keeps decreasing as the epoch progresses. The validation follows its pattern but struggles to decrease as much as the training loss and forms a bigger gap, indicating a clear generation problem. The accuracy shows the same trend that the model generalizes badly on new data.

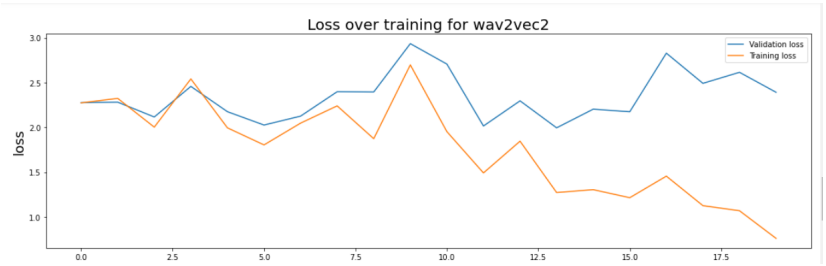

Figure 4.6: Loss over Training

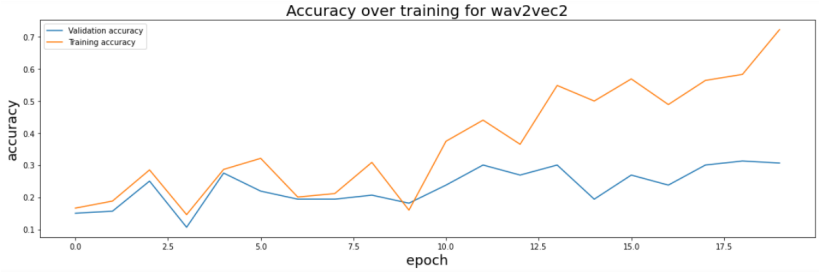

Figure 4.7: Accuracy over Training

wav2vec2 is expected to have the best performance, since it can benefit from the transfer learning technique as it is designed to learn useful representations from raw audio data before being fine-tuned on labeled data. Its performance can be improved by better model configuration and hyperparameters since the figure 4.6 shows effective learning by training data, and i have only used a feed-forward network with a hidden layer to fine-tune the model for classification, and lSTM can possibly be used before feed-forward layer to improve the sequential output data generated by pretrained wav2vec2 for enhancement. As well, the quantity of training data can also help.

Comparison in terms of accuracy:

The table below 4.1 shows the mean accuracy of five iterations over a test set across three models. CNN has the best performance in terms of both accuracy and speed of convergence, showing its more complex architecture for better learning. MLP, as the baseline model ranks second with much slower convergence and accuracy because of its simpler architecture. wav2vec2 is expected to have the best performance since it combined pretrain model to catch better-adjusted parameters using transfer learning but achieved the worst performance in this project. However, this does not mean the failure of the model; the loss of accuracy over training clearly shows its room for improvement if the overfitting can be addressed, but it emphasizes the importance and difficulty of fine-tuning the pretrained model for the desired task.

	model	accuracy
0	MLP	0.42± 0.0117
1	CNN	0.63± 0.0129
2	Wav2Vec2	0.27± 0.0484

Table 4.1: Mean accuracy of five iterations over test set across three models

Chapter 5: **Conclusion and Future Work**

By analyzing the results of each model as discussed in the result section, we can draw a general conclusion that a simple neural network like mlp training with a small number of tabular dataset can have very slow learning and convergence, and it can lead to much worse performance without optimal hyperparameter settings compared to the other complex models. CNN is a more complex model and achieved the best performance in this project as it uses convolution layers to extract features from images and is combined with a feedforward network for classification, which is beneficial to learning speed and performance, particularly for image classification. wav2vec2 pretrained model is the most complex, as its original architecture combined both transformers and CNN to capture the features from an audio file. It achieved the worst performance, emphasizing the proper handling of the fine-tuning phase, including the output of upstream and hyparameter adjusting for downstream work, which is crucial depending on your task.

Bibliography

1. Gate, M. What Are The Different Genres Of Music? https://www.musicgateway.com/blog/spotify/what-are-the-different-genres-of-music (2019).
2. Shog'ulom, K. Discover the Magic of Music Genre Classification with Machine Learning. https://medium.com/@shogulomkurganov73/discover-the-magic-of-music-genre-classification-with-machine-learning-cba2b24febd6 (2023).
3. Elbir, A., Bilal Çam, H., Emre Iyican, M., Öztürk, B. & Aydin, N. *Music Genre Classification and Recommendation by Using Machine Learning Techniques* in *2018 Innovations in Intelligent Systems and Applications Conference (ASYU)* (2018), 1–5. https://ieeexplore.ieee.org/document/8554016.
4. ANDRADA, o. *GTZAN Dataset - Music Genre Classification* 2020. https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification.
5. Tzanetakis, G., Essl, G. & Cook, P. *Automatic Musical Genre Classification Of Audio Signals* 2001. http://ismir2001.ismir.net/pdf/tzanetakis.pdf.
6. Sejal, J. Multilayer Perceptrons in Machine Learning: A Comprehensive Guide. https://www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning (2024).
7. Baevski, A., Zhou, H., Mohamed, A. & Auli, M. *wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations* 2020. arXiv: 2006.11477 [cs.CL].
8. Choudhury, N., Deka, D., Sarmah, S. & Sarma, P. *Music Genre Classification Using Convolutional Neural Network* in *2023 4th International Conference on Computing and Communication Systems (I3CS)* (2023), 1–5. https://ieeexplore.ieee.org/document/10127554.
9. Lau, D. & Ajoodha, R. Music Genre Classification: A Comparative Study between Deep-Learning and Traditional Machine Learning Approaches, 239–247 (Jan. 2022).
10. Face, H. *The Hugging Face Course, 2022* https://huggingface.co/course. [Online; accessed <today>]. 2022.
11. Paszke, A. *et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library* 2019. arXiv: 1912.01703 [cs.LG]. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.
12. Huggingface. *Transformers* https://huggingface.co/docs/transformers/en/model_doc/wav2vec2.