Music Genre Classification with Machine Learning

Lorena Yan Ryan Wang Tianhao Wu

Abstract

This paper addresses the task of music genre classification by comparing and contrasting different machine learning methods. As the baseline, the first method of this paper utilizes a softmax regression approach to genre classification that achieves a 24.24% test accuracy. Deeper inspection into the results reveals the need for techniques that can capture more fine-grained details while being more perceptive of global patterns. The second method involves a single CNN model that achieves 57.5% test accuracy. An improvement to this approach ensembles several CNN models using majority vote, which resulted in a 63.6% test accuracy. Manual error analysis reveals that both these CNN-based approaches struggle to classify spectrograms that are not representative of their genres. In these cases, CNN-based models fail to extract useful information from the spectrograms, which is the key limitation of this method. The third approach is to train Wav2Vec pretrained models with an ensemble-based learning method. Specifically, two models, a weaker model with smaller number of parameters for capturing biases and a main model for capturing more robust features, are combined together. This approach obtains a test accuracy of 74.8%. Although Wav2Vec2-based model still has difficulties in identifying music genres with high instrumentation and melody variations (such as country music), this attention-based model still achieves a relatively higher accuracy compared to the baseline and the CNN-based method. Github repo for this project can be found here: `https://github.com/Lorenayannnnn/csci467_music_genre_classification`.

1. Introduction / Problem Statement

With the digitization of modern society, more and more people have begun using internet services like Spotify or Apple Music to listen to their favorite songs. For these companies, being able to understand the musical tastes of its users has thus become a critical skill in order to keep their subscribers hooked to the service (by, for example, recommending songs that falls in a music category that the user likes).

Because of this demand, it thus becomes crucial to design a system that is capable of categorizing music based on its audio contents. This way, predicting musical preference turns into something as simple as looping through a user's playlist, categorizing each song in the list, and returning the categories with the most amount of songs inside the playlist.

Specifically, the system of categorizing music based on its audio contents should have the following inputs and outputs:

Input: Audio content. Note that this does not always mean the raw audio file. Audio content can take a wide variety of forms. In this project, 3 different representations of audio are taken as inputs for our model: raw audio, mel spectrogram (a graphical representation of a recording), and other relevant meta data of the audio such as tempo (the speed at which the audio is played)

Output: Predicted musical category

We aim to address this problem via utilizing various machine learning tools and comparing and contrasting their performances.

Brief summary of final results:

Baseline: 24.24% test accuracy with softmax regression.

CNN: 57.5% test accuracy with a single CNN model, and 63.6% with ensembling.

Wav2Vec2-based Model: test accuracy of 74.8%.

2. Related Work

As one of the most fundamental music information retrieval tasks, music genre classification has been studied with different modal data, which mainly includes visual representation of the audio, such as spectrogram that illustrates spectrum of frequencies of a signal as it varies with time, and lyrics (Li et al., 2022). Existing works can be mainly classified into 2 categories: (1) model based on RNN and attention and (2) CNN-based model.

2.1. Bidirectional-RNN(BRNN) and Attention

This method has been applied together to process input spectrograms of the audios (Yu et al., 2020). In this approach, spectrograms generated at different temporal steps of the audio via short-time Fourier Transformation are treated as input sequences of spectrum vectors. The sequences are passed into BRNN and attention-based encoder and the resulting hidden representations are passed to the classification head for producing genre prediction. One of the challenges that they have faced is the bad performance when the bidirectional RNN and attention layers are stacked together in a linear fashion for processing the inputs. In this case, the performance of the attention layer highly depends on the distributions learned and produced by the BRNN. Therefore, another model that arranges BRNN and attention layer in parallel is proposed and obtains a better performance of 90% accuracy. Inspired by the attention mechanism, we decide to adopt Wav2Vec2, a transformer-based model that is pretrained for processing human speech audio, to learn extracting underlying features of input music audios (Baevski et al., 2020). Similar to the existing approach, we also break down the input audio as a sequence of vectors by converting audios from analog to digital forms. Nonetheless, the Wav2Vec2 model applies both 1D CNN and 12 attention-based encoder layers stacked together for producing latent representations of the inputs. Ideally, our model should be able to learn to learn a more robust latent space of the audio features if there is enough data with high quality for training.

2.2. Convolutional Neural Network (CNN)

CNNs have been widely used to process visual inputs. In the context of music genre classification, a model based only on CNN with different dimensions and pooling layers has been experimented(Pelchat & Gelowitz, 2020) and has obtained a test accuracy of 85%. Another model architecture named bottom-up broadcast neural network was proposed in 2020 (Liu et al., 2019). With CNN-based broadcast module for capturing low-level features, the model obtained an accuracy of 93.9% on the GTZAN dataset, indicating that capturing both low and high level features is important for genre prediction. This was the inspiration for why transformer-based models with multiple layers were considered and analyzed in this study.

The main challenge they faced is that the number of annotated music recordings per genre class is often limited. Hence, it's not easy to train a robust CNN model from few labelled data.

Comparing our final approach with the state-of-art model, they use much more sophisticated broadcast module that allows to capture more features. Also, they use three different datasets to compare and contrast the results.

3. Dataset and Evaluation

3.1. Dataset

We use GTZAN, a popular music genre classification dataset from kaggle, to develop our model. It contains a collection of 1000 audio files, each of length 30 seconds, that are evenly categorized into 10 music genres. It also provides the converted mel spectrograms and two CSV files that characterize discrete feature values of the audio files.

Table 1. Num of Samples in Train, Dev, Test Set

GENRE	TRAIN	DEV	TEST
POP	70	20	10
METAL	70	20	10
DISCO	70	20	10
BLUES	70	20	10
REGGAE	70	20	10
CLASSICAL	70	20	10
ROCK	70	20	10
HIPHOP	70	20	10
COUNTRY	70	20	10
JAZZ	70	20	9
TOTAL	700	200	99

3.2. Data Split

The dataset is split into training (70%), development (20%), and tests (10%) sets. Table 1 shows number of samples of each genre in train, dev, and test dataset.

3.3. Evaluation

Because data samples are evenly distributed among different genres, we decide to use accuracy as the main metric to evaluate our model on the dataset.

3.4. Note

We use the same dataset and evaluation process for the final report, so no changes is made.

4. Methods

4.1. Baseline

Our baseline model aims to categorize music based on their spectrograms using softmax regression.

4.1.1. DATA PREPROCESSING

The model receives spectrogram images as input. A spectrogram is a visual representation of the spectrum of frequencies of a signal (in this case, an input audio file) as it varies with time. Spectrograms include information about the audio's frequency (ie pitch), rhythm, and dynamics (loud/soft). See figure 1 (ignore the green box for now).

For our baseline, we propose a softmax regression approach to classification. Since all audio files have the same time length (30s), and since the range of frequencies are unified,

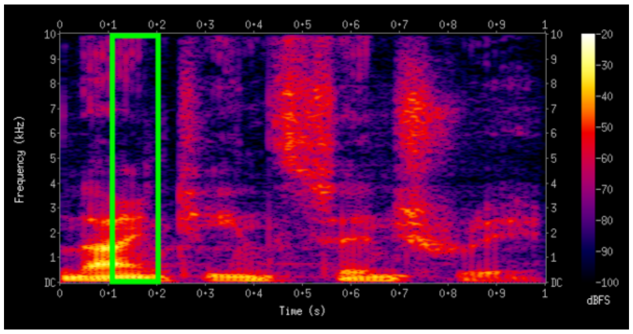

Figure 1. Spectrogram of a single audio file for the first 10 seconds. Time, frequency, and decibels are on the x, y, and z-axis respectively

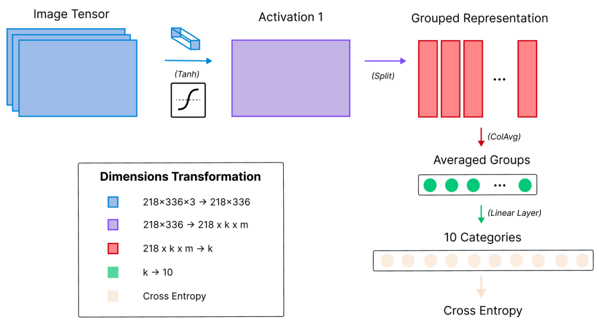

Figure 2. Structure of Baseline Model

this means that all spectrograms have the same pixel dimensions. Specifically for this study, the spectrograms have dimensions 218 (y-axis) x 336 (x-axis) x 3 (rgb). Thus, we will feed spectrograms into the model in the form of tensors of dimensions 218x336x3, where each entry will be a value from 0 to 255. Thus, if we have N images, the input training dataset will have dimensions N x 218 x 336 x 3.

We normalize the input with dimensions N x 218 x 336 x 3 as follows:

$$\text{Normalized}_{[i,j,k,l]} = \frac{\text{Input}_{[i,j,k,l]} - mean(\text{Input}_{[:,j,k,l]})}{std(\text{Input}_{[:,j,k,l]})}$$

with i being the sample index, j and k being the row and col index for the ith sample, and l being the index that chooses between R, G, or B values in the pixel located at (j, k) in sample i.

In essence, we are normalizing the RGB values of a given pixel position across all the training examples to have a mean of 0 and a standard deviation of 1. We then replace all NaN values (likely due to stdev = 0) with 0.

4.1.2. Model Structure

See Figure 2. Our baseline is composed of two layers. The first layer encodes RGB values into a single value, and outputs *Activation 1* after applying Tanh. The second layer takes in values from *Activation 1* and runs them through a fully connected layer, which is then run through softmax to achieve prediction results. Specifically, each layer is as follows:

Initially, every forward pass receives spectrograms in the form of tensors with dimension 218x336x3. For the first layer, the model applies a kernel-like transformation to each pixel in order to encode its three input RGB values into a single value. Specifically, we use a 3x1 linear layer that is dot producted with the RGB values of each pixel in the input tensor. The resulting output is a tensor with dimension 218x336, where each output value corresponds to the RGB encoding of a particular pixel. Note that the same four parameters in this 3x1 linear layer are used to encode all pixels. Note also that we can view this 3x1 linear layer as attempting to learn an encoding of RGB values that offers the "best" single-value information for genre prediction.

The output of the 3x1 linear layer is then passed through the tanh function to obtain values for *Activation 1*, which will be a tensor of shape 218x336.

Next, given this 218x336-dimension *Activation 1* input, we then split the tensor values into k columns by grouping the 336 indexes that make up the horizontal dimension into k groups. Let m = 336/k represent the number of previous columns that each grouped column has. Graphically, an example of such a column group is illustrated in Figure 1. Through experimentation, we choose k = 112 and m = 3 for the grouping hyperparameters as it yields the best results. This is expected, since having a higher k allows the baseline model to learn more fine-grained patterns across the columns of the spectrogram.

For each column i, we take the mean of the pixels that are located within this column, and output the mean as the ith index of the output. Note that the output after this transformation will have a shape of k (one output for each of the k columns). Call these output values *Summation Output*.

Finally, we pass this k-lengthed *Summation Output* through a linear layer, which then is then passed through the Cross Entropy Loss with Pytorch's Adam optimizer to run back propagation.

4.2. Method 1: CNN

CNN is one of the most standard methods to process/classify multi-dimensional data such as images. In this method, spectrograms from the GTZAN dataset are fed as input images into the CNN during the training process. Ideally, the model should learn distinct features and patterns for the spectrograms of each music genre.

4.2.1. Data Preprocessing

The original spectrograms contain a margin that is white (with RGB value 255, 255, 255). This could dramatically influence the ability for kernals to learn meaningful patterns from the actual graph, since they are always convolved with a thick margin of white pixels that each have the maximum possible RGB values. Therefore, we crop the margins of these spectrograms to only keep parts that have useful information about the songs. Because of the cropping, this also means that the CNN model needs extra paddings in its convolution layers.

4.2.2. Model Structure

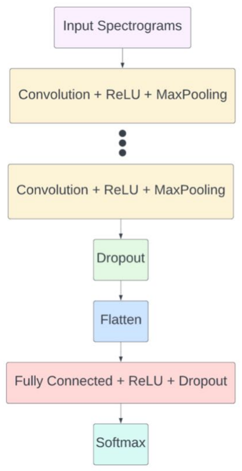
Input Spectrograms
Convolution + ReLU + MaxPooling
Convolution + ReLU + MaxPooling
Dropout
Flatten
Fully Connected + ReLU + Dropout
Softmax

Figure 3. Structure of CNN Model

The figure above outlines the architecture of the CNN model. Cropped spectrograms serve as the inputs to the model. Then the inputs are passed through several layers of convolution, ReLU, and MaxPooling. One dropout layer is added to reduce overfitting. Afterwards, data is flattened and fed into fully connected layers with ReLU and dropout. Finally, a softmax layer is applied to make predictions.

4.2.3. Hyperparameters

The following hyperpameters are considered when selecting the final model.

- Batch size: Smaller batch sizes can help improve the generalization performance of the model, but in some cases, larger batch sizes can lead to better accuracy.
- Learning rate: A high learning rate can cause the training process to converge faster. However, it could also lead to more suboptimal solutions. On the contrary, a slow learning rate takes more time but may lead to better results.
- Kernel: Larger kernel sizes are better for detecting larger, more complex features, while smaller kernel sizes are better for detecting smaller, more local features.
- Padding: With proper padding, the pixels around the edges of the image are used more frequently in the convolution operation, which can improve the overall performance.
- Stride: Stride affects the receptive field of each neuron in the network. Large strides may help the network capture more global features in the input image.
- Dropout probabilities: Dropout reduces overfitting.
- Number of neurons in hidden layer: Determine the best trade-off between model complexity and performance.
- Number of Conv+ReLU+Pool: Same as above.

4.2.4. Ensembling

For the final report, an ensemble of six models is proposed to improve the accuracy of genre prediction for a given test example. The ensemble consists of the two best-performing models from a set of CNN models with dev accuracy of 65.5% and 70.5%, respectively, and four new models with different architecture choices. Despite the new models having dev accuracies around 60%, they were included in the ensemble to increase the diversity of the models. The training process for the new models followed the same procedure as the previous models. The final prediction is determined by a majority vote of the six models, where a consensus is reached if a genre receives at least three votes. If there is no consensus, the prediction of the model with the best dev accuracy is used.

Ensembling is expected to address the limitations of the single CNN model approach used in the previous midterm report, particularly in cases where a model is ineffective in predicting certain genres due to potential spurious correlations. By using an ensemble of multiple CNN models that capture distinct features of different genres, the collective wisdom of the models can be leveraged to produce more accurate and robust predictions.

4.3. Method 2: Wav2Vec2-based Model

Inspired by the previous approach of applying attention and treating audios as sequences of input vectors extracted at different temporal steps (Yu et al., 2020), we think that utilizing a relatively more sophisticated, attention-based model can result in high performance with better capability of extracting low and high level features within the audios. Wav2Vec2, one of the current state-of-the-art models for automatic speech recognition, is selected for our task (Baevski et al., 2020). Specifically, wav2vec2-base is selected, which is pretrained on 53k hours of unlabeled audio data. With its ability of processing audio signals, a pretrained Wav2Vec2 model is used as a feature extractor for processing input audio segments. After a series of experiments, the following data preprocessing method and model structure are adopted.

4.3.1. Data preprocessing

Due to the limited amount of data entries in the GTZAN dataset (only around 1000 raw audios), we decided to do data augmentation and increase number of inputs by splitting each audio file into shorter segments. Specifically, given that each audio is 30-seconds long, every input is split into 18 segments with each segment having 50% overlap with the previous and the latter one.

Torchaudio is applied to load raw audio and convert the audio from continuous to digital form (an array of float). Due to the sample rate (number of samples per second taken when converting audio to discrete signals) required by Wev2Vec2's feature extractor, the audio array is down-sampled from 22050 to 16000. Last but not least, the down-sampled audio arrays are normalized to standard normal distribution in order to help the neural network learn features on a similar scale.

4.3.2. Model Structure

The model architecture is illustrated on the left side of figure 4. First, the raw audio segments are converted to digitized format and down-sampled. Then, the audio array data are passed into the convolution layers and transformer encoders of the pretrained Wav2Vec2 model. After hidden states are extracted (with a dimension of sequence length of 49 and hidden dimensions of 768), all hidden states at each token position are pooled into one representation via averaging all the outputs. Then, the pooled result is passed into the classifier head, which consists of a linear projector layer, a dropout layer, tanh, and a final classification layer that maps from hidden dimension to number of labels, which is 10 in this case. At last, the logits are used to make predictions and to calculate cross entropy loss for backpropagation.

To further improve the model's performance, ensemble learning is applied here: as illustrated on the right side of figure 4, predictions of a biased model and a main model are combined via weighted average:

$$\textit{Ensemble Prediction} = \alpha * \textit{Pred}_{\text{main}} + (1 - \alpha) * \textit{Pred}_{\text{biased}}$$

Here, both the biased and the main model has the same structure as described previously. Nonetheless, the pretrained wav2vec2 model that the biased model uses as the encoder is loaded from pretrained *wav2vec2-base*, which has fewer number of parameters than *wav2vec-large* that is used in the main model. Here, it is assumed that a smaller model is much more likely to overfit to the data be biased. By combining the outputs of the biased and the main model, we expect the main model to capture the unbiased portion of the underlying features and thus is able to produce more robust predictions during inference.

4.3.3. Hyperparameters

The following hyperpameters are considered when running experiments.

- Part of the pretrained model that will be frozen: Given the large scale of the model, part of the pretrained model should be frozen, because with limited amount of data entries, it will be very likely for the model to overfit to the training set and have bad performance on the dev and test set. Different settings are trialed: freeze the entire model, freeze everything except the last or the last two encoder layers. **Freezing everything except the last or the last two encoder layers** is chosen with higher accuracy.
- Method for processing last hidden layer: as illustrated in figure 4, hidden states at different token positions are pooled into one representation, which will be passed into the classifier head. 4 pooling methods have been used: last (only use the hidden states of the last token), average (average over all hidden states), sum (sum over all hidden states), and max (choose the maximum hidden value at each position). Based on experiment result, **averaging** pooling method is chosen.
- Learning rate: 1e-3 is chosen at last for faster learning due to limited amount of accessible computation power and time.
- Structure of the classifier head: Two different structures are experimented with: (1) A two-linear-layer structure that first maps the input data (of of length 768) to a hidden dimension of size 768, and then from this hidden layer to a layer of size 10 (each corresponding to the ten different labels) and (2) One linear layer followed with tanh, dropout layer, and a final linear layer for producing logits for each genre category. Based on the performance, the third approach is selected.
- α value for prediction ensemble: based on experiment results, 0.1 is chosen for α.

All hyperparameters are chosen based on the model's performance (accuracy value) on the dev set. For more details,

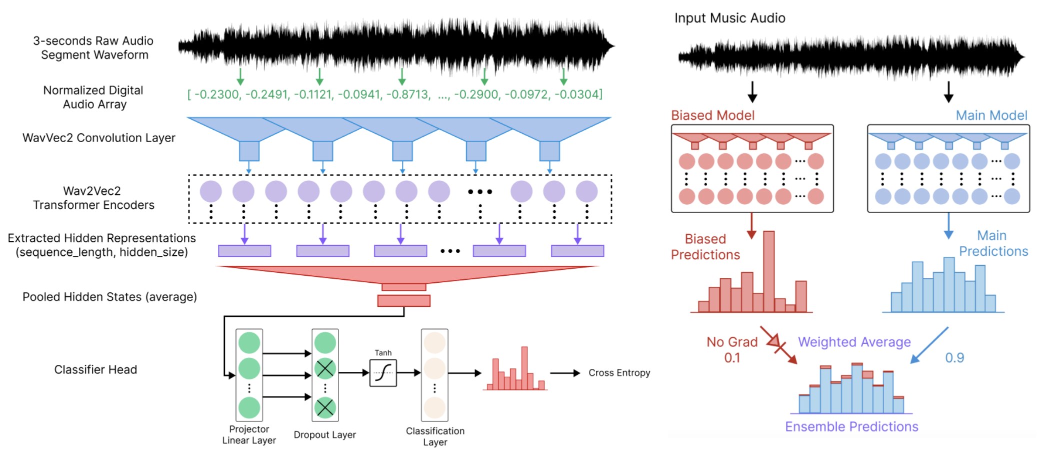

Figure 4. Visual illustration of internal architecture of Wav2Vec2-based model (left) and how predictions of a biased and a main model are combined together (right). During training, predictions of the two models are weighted (0.1 for biased and 0.9 for main) and averaged for updating the model. During inference, only predictions of the main model are used.

Table 2. Classification Accuracy for Baseline model

	TRAIN	DEV	TEST
BASELINE	0.59714	0.28500	0.24242

please refer to the table 5 in the experiment section for accuracy values.

5. Experiments

5.1. Baseline

Based on the method from section 4.1 along with the train/dev/test split highlighted in 3.2, we trained a model using a learning rate of 0.001, with 500 epochs, and a batch size of 32. The results are listed in table 2.

5.2. Method 1: CNN

Experiments are run on different choices of hyperparameters. We use early stopping to prevent overfitting. Table 3 records the results to select the hyperparameters of the CNN model. The last row shows the best performance on the dev set with an accuracy of 70.5%. Using this model, we get 57.58% test accuracy.

The six models for ensembling are shown in Table 4. Using the procedure described in previous section, we get 63.6 % test accuracy. Compared with midterm report, the new model performs slightly better. Nonetheless, the improvement is not significant. This result is not surprising because it's likely that different models make the same mistakes even if trained with different hyperparameters. Majority voting prevents some minor errors that can be easily detected and fixed. However, it cannot handle very difficult cases where the test examples' spectrograms are "outliers".

5.3. Method 2: Wav2Vec2-based Model

Table 5 and 6 displays accuracy of wav2vec2-based models and ensemble-based models with different hyperparameters. In general, it can be concluded from the table that:

- **Freeze all layers of the pretrained wav2vec2 model except the last 2 encoder layers** can result in better performance: because Wav2Vec2 model is pretrained for speech recognition instead of music audio processing. Therefore, freezing the entire model can result in producing bad hidden representations of the music audio array. On the other hand, having too many layers unfrozen may cause the model to overfit to the data, which is reflected in row 8 of table 5.
- With more parameters, Wav2Vec2-Large based model is able to achieve better performance compared to Wav2Vec2-base model. Therefore, it is assumed that Wav2Vec2-base model is weaker than Wav2Vec2-Large based model. Correspondingly, the former is used as the biased model and the latter is used as the main model for ensemble-based learning method.
- Ensemble-based learning can result in better performance, which can be seen from the result presented in table 6. Ensemble-based method does help us to leverage features learned from multiple models.
- Choosing not to freeze the biased model leads to slightly better performance. One possible explanation is that if the biased model is weaker or more biased, then the biased model is more effective in capturing underlying biases, while the main model can focus on cap-

Table 3. Classification Accuracy of CNN model on dev set

BATCH	LR	KERNEL	PAD	STRIDE	DROPOUT	HN	LAYERS	TRAIN ACC(%)	DEV ACC(%)
64	0.001	3x3	1	1	0.4 0.1	128	2	94.13447783	**57.5**
32	0.001	3x3	1	1	0.4 0.1	128	2	95.42203147	54
128	0.001	3x3	1	1	0.4 0.1	128	2	87.12446352	56.5
64	**0.002**	3x3	1	1	0.4 0.1	128	2	94.70672389	47.5
64	**0.0005**	3x3	1	1	0.4 0.1	128	2	98.14020029	**59**
64	**0.0001**	3x3	1	1	0.4 0.1	128	2	91.84549356	54
64	0.0005	**5x5**	1	1	0.4 0.1	128	2	98.85550787	**60**
64	0.0005	**10x10**	1	1	0.4 0.1	128	2	91.70243205	58
64	0.0005	5x5	**2**	1	0.4 0.1	128	2	97.28183119	57.5
64	0.0005	5x5	1	**2**	0.4 0.1	128	2	90.84406295	**65**
64	0.0005	5x5	1	2	**0.5 0.2**	128	2	80.40057225	58.5
64	0.0005	5x5	1	2	**0.3 0.1**	128	2	91.13018598	59.5
64	0.0005	5x5	1	2	0.4 0.1	**64**	2	97.71101574	58
64	0.0005	5x5	1	2	0.4 0.1	128	**3**	84.97854077	**65.5**
64	0.0005	5x5	3	2	0.4 0.1	128	**4**	83.83404864	**70.5**
								TEST ACCURACY	**57.5**

Table 4. Classification Accuracy of CNN models on dev set (ensembling)

BATCH	LR	KERNEL	PAD	STRIDE	DROPOUT	HN	LAYERS	TRAIN ACC(%)	DEV ACC(%)
64	0.0005	5x5	3	2	0.4 0.1	128	4	83.83404864	70.5
64	0.0005	5x5	1	2	0.4 0.1	128	3	84.97854077	65.5
64	0.0005	5x5	1	2	0.1 0.1	128	3	72.24606581	60.5
64	0.0005	5x5	1	2	0.1 0.1	128	2	76.5379113	59
64	0.0005	5x5	1	2	0.25 0.1	128	2	81.85550787	64.5
64	0.0005	10x10	1	2	0.25 0.2	32	2	69.67095851	57
								TEST ACCURACY	**63.6**

turing more robust features. (However, due to time and resource limitation, a freezed version of Wav2Vec2-based model is used as a biased model for the SOTA model in our project).

- Using a model that is already finetuned on our dataset instead of finetuning the Wav2Vec2-large model from the beginning will give better performance, which is expected because the model is already adapted to the new task of music classification to some extent and thus may already have some prior knowledge.

5.4. Comments

Based on the results shown previously, Wav2Vec2-based model stands out as the best method with 74.8% test accuracy. Both the baseline and CNN-based methods struggle to distinguish certain atypical examples mainly because the their prediction only depends on spectrograms. If the spectrograms are not representative of their respective genre, these two methods fail to extract useful information. In contrast, Wav2Vec2 has already been trained on a large amount of audio data, which can make the model to be more effective at handling basic features of audio at the first place. In addition, Wav2Vec2 model also take advantage of CNN. Specifically, Wav2Vec2 model combines CNN for first capturing low-level audio features such as pitch and timbre and transformer architecture for capturing high-level features, which allows Wav2Vec2 to have greater capacity at capturing nuances of raw audio of different music genres, especially when the dataset has small number of examples with high degree of music diversity.

6. Discussion

6.1. Baseline

The model achieved a 21.5% dev accuracy. Upon further inspection, the specific distribution of accuracy for each category in the dev set is as follows:

The baseline model achieved an 80% accuracy when categorizing Metal music in the dev set. Metal music is known for its ”distorted guitars” and ”emphatic beats and loudness”. Visually inspecting the spectrograms of Metal Music reveals a dense graph of bright pixels that usually extends the entire graph. See Figure 5 for a side-by-side comparison between ”Metal” and ”Rock” spectrograms:

We note potential problems for why the baseline model is not able to achieve high accuracies in other categories such as Classical or Rock:

Table 5. Classification Accuracies for Wav2Vec2-based Model on Dev and Test Set

Model	Frozen Layers	LR	DEV ACC(%)	TEST ACC(%)
Baseline	-	-	28.5	24.2
Wav2Vec2-base				
	Entire Wav2Vec2	1e-3	53.2	53.8
	All w/o last encoder layer	1e-3	61.5	60.8
	All w/o last 3 encoder layers	1e-3	42.4	43.3
	All w/o last 2 encoder layers	1e-3	63.8	62.5
Wav2Vec2-large	All w/o last encoder layers	1e-3	68.8	68.2
	All w/o last 2 encoder layers	1e-3	70.8	68.9
	All w/o last 2 encoder layers	1e-4	**71.9**	**72.5**

Performance (accuracy) of models with different hyperparameters on dev and test set. "Frozen layers" refers to the layers that will be freezed inside the pretrained Wav2Vec2 model. Regarding details of pooling method for processing last hidden states and model structure, please refer to the method section 4.3 for more details.

Table 6. Classification Accuracies for Wav2Vec2-ensemble-based Model on Dev and Test Set

Biased Model	Frozen Layers(Main)	α	DEV ACC(%)	TEST ACC(%)
wav2vec2-base	All w/o last 2 encoder layers	0.1	70.6	70.3
wav2vec2-base(No grad)	All w/o last 2 encoder layers	0.1	69.7	69.3
wav2vec2-base(No grad)	(Finetuned)All w/o last 2 encoder layers	0.1	**73.8**	**74.8**

Performance (accuracy) of ensembled-based models with different hyperparameters on dev and test set. A learning rate of 1e-3 is used for all experiments presented in this table. "Finetune" in the frozen layers column indicates that the main model is initially loaded from finetune checkpoint of a wav2vec2-large based model with best performance presented in table 5.

Table 7. Accuracies Across Genres of Baseline, CNN-based, and Wav2Vec2-based model

Genre	Baseline(%)	CNN(%)	Wav2Vec2(%)
Pop	30.0	75.0	61.2
Metal	80.0	85.7	89.9
Disco	30	50.0	65.8
Blues	15	41.7	74.7
Reggae	30	85.7	74.9
Classical	10.0	69.2	96.7
Rock	15.0	30.0	58.6
Hiphop	20	61.5	86.0
Country	20	57.1	56.0
Jazz	35	40.0	84.5

- Most frequency information is stored across the y-axis. However, the baseline model crudely takes the average of the values across the entire y-axis for a given column, thereby losing large portions of valuable frequency information that the spectrogram provides.

- Rhythm and frequency progressions of music may be interpreted incorrectly, simply because the model is only able to learn pattern progressions across entire columns. Since columns are of fixed size (that are pretty large), this means that genres like classical music - where chord progressions and rhythm changes very fast - would not be picked up by the baseline model

Given these problems, there are a few solutions. Firstly, one root problem is that the current approach splits these input tensors into fixed splits, which can contribute to a lack of model plasticity given the wide range of musical styles. Moving forwards, the data should be processed in a "non-discrete/non-disjunct" manner.

Furthermore, analyzing the potential causes for inaccuracies in the baseline model, we see that we need to improve the model's ability to take into account variations of frequency within a given column. Thus, it may be better to split the spectrogram horizontally such that the entire image is split into grids. A CNN approach can be used to expand on this idea.

Since the current baseline model also lacks the ability to track the relative relations amongst progressions/patterns both within a column and across multiple columns, the usage of attention is also another reasonable improvement.

See CNN and Wav2Vec2-based methods for further expansion on these two improvements.

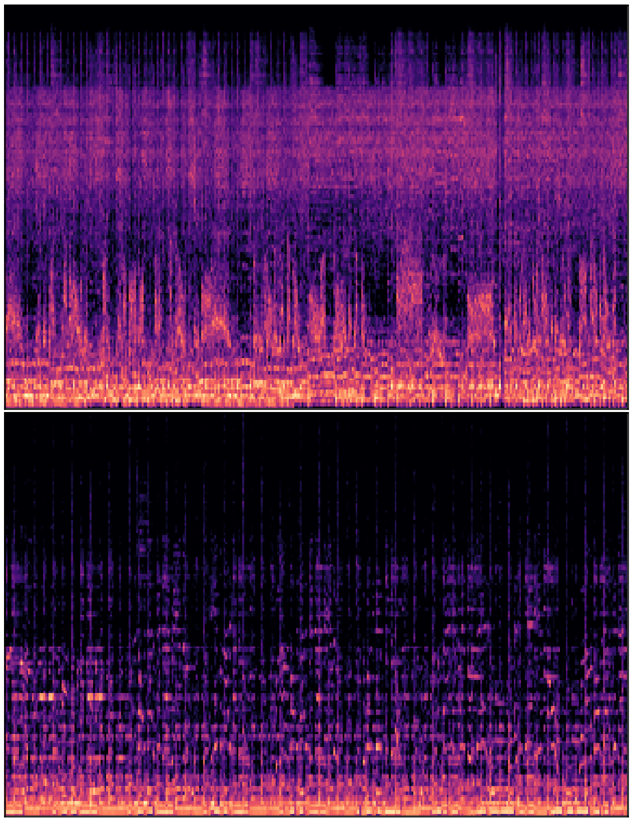

Figure 5. Spectrogram of Metal (Top) vs Rock (Bottom)

6.2. Method 1: CNN

From Table 7, we discover that the model performs well on metal and raggae, and badly on rock and jazz. We construct the confusion matrix to help understand the underlying reasons (see Table 8). For the rock pieces misclassified as blues, their rhythmic patterns are not as strong as other rock pieces. As a result, their spectrograms are asymmetric and more ”soft”, displaying some features common in other genres such as blues.

To address the limitations of the CNN model, we construct an ensembling model and analyze its performance using a confusion matrix (Table 9). The ensembling model shows improved accuracy in blues, country, disco, metal, and rock genres, but still doesn't perform as well as we expected. Upon a manual error analysis, we observe that almost all the same test examples are misclassified by both the CNN and ensembling models. Closer analysis reveals that these misclassified examples have spectrograms that are not representative of their respective genres, indicating a potential limitation of CNN in capturing certain types of audio features.

Therefore, we suggest the possibility of ensembling CNN with other models such as MLP that can capture different types of audio features. This could potentially enhance the accuracy and robustness of the ensembling model in predicting genre for audio samples with diverse features.

6.3. Method 2: Wav2Vec2-based Model

As can be seen from table 7, the best Wav2Vec2-based model with dev accuracy of 73.8% and test accuracy of 74.8% has best performance on Classical and worst performance on country. Based on manual observation of the spectrograms and listening of raw audios, classical music has a relatively consistent structure with a clear hierarchy of rhythm and melody. Compared to other genres like Hiphop (which is the genre on which the model had the best performance in the midterm report), classical music does not have elements of electronic manipulation, which makes the audio more representative of the original performance.

In contrast, country music may be less identifiable due to its high degree of variability in instrumentation and arrangement. Different country music pieces may feature different instrument combinations and performance style. Thus, it can be more difficult for the model to extract distinguishable features. The difficulty is also reflected in the performance result: although Wav2Vec2-large has a much higher number of parameters than Wav2Vec2-base, the accuracy for country music has only been increased by 0.1% since the midterm report, whereas accuracy of classical music has been increased by 19.1%.

7. Conclusion

Overall, the project has the following findings:

- The single CNN model achieves a test accuracy of 57.5%, while the ensembling model improves the accuracy to 63.6%. However, the ensembling approach still fails to accurately predict the ”confusing” examples. This indicates a limitation of the CNN models, as they struggle to extract useful information from spectrograms that deviate from typical examples of their genre. To further enhance the performance of the existing model, we suggest two possible approaches. First, we could implement a more sophisticated CNN architecture that can better capture diverse audio features and handle more complex audio data. Alternatively, we could ensemble CNN with other machine learning models that can complement its strengths and overcome its limitations. This could potentially improve the accuracy and robustness of the model in predicting genre, even for ”confusing” examples with atypical spectrograms.

- Wav2Vec2 based model achieves 73.8% accuracy on the dev set and 74.8% accuracy on the test set, with

Table 8. Confusion Matrix of CNN

	PREDICTED									
	BLUES	CLASSICAL	COUNTRY	DISCO	HIPHOP	JAZZ	METAL	POP	REGGAE	ROCK
BLUES	5	0	1	0	0	2	0	0	1	1
CLASSICAL	0	9	0	0	0	1	0	0	0	0
COUNTRY	2	0	4	2	1	1	0	0	0	0
DISCO	0	0	0	6	3	0	1	0	0	0
HIPHOP	0	0	0	1	8	1	0	0	0	0
JAZZ	0	4	0	0	0	4	0	0	0	1
METAL	1	0	0	0	1	0	6	0	0	2
POP	1	0	0	1	0	0	0	6	0	2
REGGAE	0	0	0	1	0	1	0	1	6	1
ROCK	3	0	2	1	0	0	0	1	0	3

Table 9. Confusion Matrix of CNN (ensembling)

	PREDICTED									
	BLUES	CLASSICAL	COUNTRY	DISCO	HIPHOP	JAZZ	METAL	POP	REGGAE	ROCK
BLUES	6	1	1	0	1	1	0	0	0	0
CLASSICAL	0	9	0	0	0	1	0	0	0	0
COUNTRY	1	0	6	0	1	0	0	0	0	2
DISCO	0	0	0	8	1	0	1	0	0	0
HIPHOP	0	0	0	2	7	0	0	0	1	0
JAZZ	0	3	1	0	0	3	0	0	1	1
METAL	0	0	0	0	0	0	9	0	0	1
POP	0	0	0	0	1	0	1	6	0	2
REGGAE	0	0	1	2	0	0	0	1	5	1
ROCK	1	0	2	2	0	0	0	1	0	4

Wav2Vec2-large finetuned on the GTZAN dataset as the main model and Wav2Vec2-base as the biased model. Adopting ensemble-based learning method helps us to increase accuracy by approximately 10% compared to midterm performance. Based on the result, it can be concluded that a model pretrained for human speech recognition is able to be adapted for music audio classification tasks with assistance of ensemble-based learning method. To further improve model's performance, a larger dataset with better quality should be obtained, as Wav2Vec2 model is too large such that having only 1000 samples is not sufficient for training. Also, the effect of adjusting α for ensemble learning should be further explored, as keeping a proper balance between the biased and the main model can be critical to the model's performance.

Learning from this project:

- It is important to systematically organize codes into functions and files into certain hierarchies to enhance productivity when running experiments.
- One should closely examine the dataset, such as distribution of data samples by labels, and spend more time on experimenting with different data-preprocessing methods like data augmentation. Having a dataset with good quality can greatly improve the model's performance, especially when the dataset is small.
- Performance results should be logged properly to avoid wasting time on loading model checkpoints and rerun experiments.
- Tools such as parameter parsers and sh scripts should be used for changing hyperparameters of the model in a clearer and more logical way.
- Before model implementation, one should conduct enough literature review and have better intuition on examining the effectiveness of his or her own methods.

References

Baevski, A., Zhou, H., Mohamed, A., and Auli, M. wav2vec 2.0: A framework for self-supervised learning of speech representations, 2020.

Li, Y., Zhang, Z., Ding, H., and Chang, L. Music genre classification based on fusing audio and lyric information. *Multimedia Tools and Applications*, Dec 2022. ISSN 1573-7721. doi: 10.1007/

s11042-022-14252-6. URL https://doi.org/10.1007/s11042-022-14252-6.

Liu, C., Feng, L., Liu, G., Wang, H., and Liu, S. Bottom-up broadcast neural network for music genre classification, 2019. URL https://arxiv.org/abs/1901.08928.

Pelchat, N. and Gelowitz, C. M. Neural network music genre classification. *Canadian Journal of Electrical and Computer Engineering*, 43(3):170–173, 2020. doi: 10.1109/CJECE.2020.2970144.

Yu, Y., Luo, S., Liu, S., Qiao, H., Liu, Y., and Feng, L. Deep attention based music genre classification. *Neurocomputing*, 372:84–91, 2020. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2019.09.054. URL https://www.sciencedirect.com/science/article/pii/S0925231219313220.